

Federal Reserve Image Export Standards Document

Comments or questions should be directed to:

Jerry Giaccai
National Image Services
Federal Reserve Bank of Boston
617 973-3774
jerry.giaccai@bos.frb.org

-Or-

Dave Mollon
National Image Services
Federal Reserve Bank of Boston
617 973-3141
dave.mollon@bos.frb.org

1. Summary

This is the specification for the Common Output Format (COF) for both the export of images and associated financial data from multi-vendor Federal Reserve archives to financial institutions and for the import of such items from financial institutions into the Federal Reserve archives.

All check images exported from image systems located at Federal Reserve Offices will be in a newly developed standard format called the Common Output Format or COF. The COF has been designed to standardize the export and import of image data regardless of what type of hardware/software was used to capture and store the images. By adapting their software to understand the COF, vendors can be assured that data coming from any Fed office on media will be useable when received. Financial institutions that use Fed Image Export Services and receive CD-ROM or magnetic tape media will need to work with their software vendors to ensure that their software can understand COF.

In COF, the Images File supports multiple types of image compression (T.6, JPEG, ABIC gray, and ABIC binary) as well as multiple image file formats (TIFF and IOCA). On CD-ROM media, identification and financial information about each check is carried in the Data File, along with the offset of the item's corresponding images in the Images File. Index files have also been included on CD-ROM media to make access to images more efficient. COF interchanges on tape media incorporate Images Files of the same format, but include neither Data Files nor Index Files.

COF is not a Federal Reserve proprietary format and can be used for delivery of image data outside the Federal Reserve System if desired.

2. Contents

1. SUMMARY	1
2. CONTENTS	2
3. INTRODUCTION	3
4. DESIGN CRITERIA	4
4.1 MIGRATION TO NETWORK USAGE	4
4.2 EFFICIENCIES	5
4.3 USABLE WITHOUT IMPORT	5
4.4 CROSS-VOLUME SEARCHING	5
4.5 USE OF TOOLS	5
5. RELATIONSHIP TO ANSI X9.46 FINANCIAL IMAGE INTERCHANGE STANDARD	5
6. TERMINOLOGY	6
7. COMPLIANCE	7
7.1 REQUIREMENTS ON WRITERS.....	7
7.2 REQUIREMENTS ON READERS	7
7.3 REQUIREMENTS ON MEDIA	7
7.4 LIKELY AREAS OF CHANGE.....	7
8. OVERALL STRUCTURE	8
8.1 FILE NAMING	8
8.2 OPTIONAL FILES	9
8.3 FILE SUITES	10
8.3.1 <i>Format of .ini Files</i>	10
8.3.2 <i>File Suite Header Structure</i>	11
8.3.3 <i>File Suite Trailer Structure</i>	11
8.4 MEDIA SETS.....	12
8.4.1 <i>Media Sources</i>	13
9. MEDIA HEADER FILE STRUCTURE	13
10. MEDIA TRAILER FILE STRUCTURE	14
11. DATA FILE STRUCTURE	15
11.1 DETAILED STRUCTURE OF THE DATA FILE	15
11.1.1 <i>Data File Fields</i>	15
11.1.2 <i>dBASE III Compatible DBF File Format</i>	17
12. INDEX FILES STRUCTURE	20
12.1 DETAILED STRUCTURE OF INDEX FILES.....	20
13. IMAGES FILE STRUCTURE	23
13.1 SUPPORTED IMAGE COMPRESSION TYPES	23
13.2 SUPPORTED IMAGE FILE TYPES.....	23
13.3 AGGREGATION METHOD.....	23
13.3.1 <i>Relation to X9.46 Approach</i>	23

13.4 DETAILED IMAGES FILE STRUCTURE	24
13.4.1 <i>ItemData in Images Files</i>	26
13.5 DETAILED TIFF FILE STRUCTURE	27
13.5.1 <i>General Notes on TIFF Tags</i>	27
13.5.2 <i>Group 4 Compressed Binary Files</i>	27
13.5.2.1 Notes Regarding the Group 4 Image Tags	28
13.5.3 <i>JPEG Files</i>	29
13.5.3.1 Notes Regarding the JPEG TIFF Tags	30
13.6 DETAILED IOCA FILE STRUCTURE	32
13.6.1 <i>ABIC Compressed Binary Files</i>	32
13.6.2 <i>ABIC Compressed Grayscale Files</i>	34
13.7 FINANCIAL DATA IN IMAGE FILES	36
14. PHYSICAL MEDIA TYPES	37
14.1 PHYSICAL LABELING REQUIREMENTS	37
14.2 MEDIA-SPECIFIC ISSUES: CD-ROM	37
14.3 MEDIA-SPECIFIC ISSUES: TAPE	38
14.3.1 <i>Tape Media Types</i>	38
14.3.2 <i>Physical Labeling Requirements</i>	38
14.3.3 <i>Media-Specific Issues: Tape</i>	38
14.3.4 <i>Tape Logical Formatting</i>	39
14.3.5 <i>Required Files and File Ordering</i>	44
14.3.6 <i>Media-Specific Issues: Tape, DLT™</i>	45
14.3.6.1 DLT1	45
14.3.6.2 DLT2	45
14.3.6.3 DLT3	45
14.3.6.4 DLT4	46
14.3.6.5 DLT5	46
14.3.7 <i>Media-Specific Issues: Tape, 3480-Compatible</i>	46
14.3.7.1 3480	46
14.3.7.2 3490	46
14.3.7.3 3490E	46
14.3.8 <i>Media-Specific Issues: Tape, 8mm Helical-Scan</i>	47
15. REFERENCES	47
16. REVISION HISTORY	48
17. ANNEX A (NORMATIVE). DBF FILE FORMAT	49

3. Introduction

A working group within the Federal Reserve has developed this image delivery standard. This Image Export Standards Working Group is made up of current image system implementers from multiple Fed districts that are working with archive products from a variety of vendors. Much careful deliberation has gone into this effort.

The goal of the image delivery standard is to provide a uniform Common Output Format (COF) for delivery of images out of the image archive systems of the various districts to the Fed's customer institutions. This goal falls far short of developing a general-purpose interchange vehicle for any-to-any interchange; such a need is already well served by the full-featured, robust and complicated ANSI X9.46 Financial Image Interchange Standard (FIIS).

The current situation is that different Federal Reserve districts use different vendors' products to deliver images on CD-ROM or magnetic tape to client institutions. These all currently have

different formats. The goal of the working group was to quickly produce a common format for image delivery on these media which did not differ too significantly from the current approaches of these vendors, yet showed leadership in moving the delivery philosophies of these products toward the approach envisioned in the FIIS.

Clearly, thinking will be changing radically over the next few years about how and where images will be archived and delivered and the FIIS will be part of that. Once the ability of the system to capture good images and safely archive them with a responsive retrieval system has been established in the public mind, and once networking technology makes access of remote archives a common occurrence, the complicated and robust FIIS architecture will really begin to shine. At that point it will not be necessary to perform bulk delivery of images to banks. People will not need image statements (to store and lose), since they know they can get instant on-line access to all their archived checks.

The bulk delivery of images on media is a simple, interim application, which calls for a simple, lightweight solution, which can be developed and deployed in a short time frame. The transition to use of the FIIS would best occur during the shift away from bulk delivery and toward networked, remote archives.

Accordingly, the COF does not use FIIS. It does, however take several steps in the direction of the philosophies embodied in FIIS, in order to smooth that later transition. For example:

1. COF supports multiple image compression types (T.6, JPEG and ABIC) and multiple embedded file formats (TIFF and IOCA) as does FIIS.
2. COF combines multiple images into a single storage-efficient file in a manner consistent with the approach taken in the FIIS -- by use of concatenation of separate image files rather than by use of a single multi-page TIFF file. The structure of the resulting Images File mirrors the structure and terminology of the corresponding X9.46 transaction set, without using the X.12 EDI syntax.
3. On CD-ROM media, COF keeps a copy of the financial data separately bundled (in a Data File) from the image data (in the Images File), just as FIIS keeps these items in separate transaction sets.
4. The fields of the Data File are those found in X9.37, just as they are in FIIS.

In summary, this working group believes that the Common Output Format meets its stated goals, can be deployed expeditiously to solve a currently pressing problem and also provides leadership toward migration to the next generation in check image systems and eventual wide implementation of the FIIS by the financial system.

4. Design Criteria

The criteria discussed in the sections below affected design decisions in creating the Common Output Format.

4. Migration to Network Usage

It is expected that the eventual architecture for export of images from the Federal Reserve archive will use network interchange rather than media.

The most efficient such network architecture would allow queries requesting images from a bank by its customers to be relayed to the Federal Reserve archive. Only then (for the tiny percentage of images actually needed), would the images of an item be relayed to a bank for transmittal to a customer. This presumes a post-Image Statement Print environment.

In this document, reference is made to files on media. It is recognized that before network

transfer can occur, an additional transport specification will be required which permits identification and request of individual items and specifies the bundling of multiple items into a unitized transfer. It seems clear that the ANSI X9.46 standard would be the most appropriate choice for any network transfer (see *Section 5. Relationship to ANSI X9.46*, below).

4.2 Efficiencies

1. The COF should be reasonably time efficient to write and to read.
2. The COF should be reasonably efficient in its use of the available storage space on the delivery medium.
3. For random access media (CD-ROM), the COF should support quick search and retrieval times so it can be used as the sole storage copy by the end recipient.

4.3 Usable Without Import

In a sense, the COF format is intended to permit a copy of a portion of the Federal Reserve archive to be distributed outside the walls of the Fed (at least for CD-ROM media). In that sense, it is less an interchange than a means for a financial institution to access its own items without networked retrieval from a centralized Fed archive. Tape, on the other hand, is intended as an interchange means where import into a local imaging system occurs at the receiving institution.

The COF therefore should represent an open archival format, which can be retained for future reference, and searched against directly. It should not require bulk import at the financial institution onto other media in order to be useful (while not precluding such importing).

For ease of cross-volume searching, a financial institution **may** choose to import the Data Files (as opposed to the Images Files) into a larger, primary database. The large Images Files, however, may be retained directly on the originally delivered media in the user's archive as the primary image storage medium without additional expense.

Given a well-identified date range, however, individual media may be searched directly without any importing process. This supports the familiar microfilm reference model and minimizes costs for smaller institutions.

4.4 Cross-Volume Searching

The COF should enable use of search capabilities across multiple COF volumes when the end recipient uses them as the sole storage copies.

4.5 Use of Tools

The COF should support the use of readily available tools for searching and viewing. Ideally, the option should exist of placing some format verification tools on the delivery media.

5. Relationship to ANSI X9.46 Financial Image Interchange Standard

The Common Output Format is not intended to replace ANSI X9.46, which the Federal Reserve endorses. X9.46 Financial Image Interchange Standard (FIIS) is a general-purpose check image standard for interchanges between any two financial institutions. It incorporates a very robust and complex framework for queries to and for acknowledgments and retrievals from remote archives. FIIS supports a variety of applications including forward presentment, return item processing and check safekeeping, among others. While the FIIS may be used for bulk delivery of images, its complex architecture anticipates a much richer set of relationships between financial institutions and provides mechanisms to support them. This complexity begins to pay off in a multi-party, remote archiving environment.

The Common Output Format is a much simpler means of delivering media from the Federal Reserve to its client financial institutions. These media are directly usable as a local archive copy at those institutions. A FIIS interchange is very much a sequential data stream, the use of which carries the expectation that a processing center at the receiving institution will parse the stream, manipulate the data and store it into a local, full-featured image system and database. The COF, through its inclusion of immediately usable DBF and index files, requires only a PC-class database; this database can immediately support informal reference searches against the COF media without an initial digestion process.

With the FIIS, there can be no naive users. To take on FIIS, a financial institution must make a major corporate commitment, purchase an in-house image management system, develop software, and create business and technical agreements with potential interchange partners. The COF, on the other hand, by virtue of its simplicity, supports casual use, is easily implemented and still provides a migration path toward a full FIIS implementation as the financial system moves toward a remote archiving approach.

6. Terminology

1. **Item.** A representation of a single physical object, such as a check. A single item corresponds to a row in a Data File.
2. **Data Files.** These are the DBF format database files which contain all the financial, processing and other non-image data for one or more items.
3. **Index Files.** These are a set of NDX format database index files associated with a Data File which permit faster searching of that Data File.
4. **Images Files.** These are the files which contain all the image data (as well as all the financial, processing and other non-image data) for one or more items.
5. **Data Element.** This is an attribute or field having a distinct value for each of the items in a Data File. It corresponds to a column within a Data File or database table.
6. **Field.** Synonym for Data Element.
7. **File Suite.** A collection of related files for a group of items. A mechanism used to limit the size of Images Files.
8. **File Suite Header.** A file containing descriptive information about the files in a File Suite.
9. **File Suite Trailer.** A file containing verification information about the files in a File Suite.
10. **Media Header.** A file containing descriptive information about the File Suites on a piece of media.
11. **Media Trailer.** A file containing verification and descriptive information about the File Suites on a piece of media.
12. **Media Set.** Multiple pieces of media serving the function of a single piece of media, which would otherwise be hindered by a size limitation.
13. **Writer.** A program that creates a set of COF files.
14. **Reader.** A program that reads and interprets a set of COF files. This may constitute a portion of a financial institution's archive program or it may be a bulk import utility ancillary to the financial institution's archives program. Another example of a reader would be a PC-based program which interprets the COF format, permits selection of items by search criteria and which then invokes an integral or separate viewing routine to display the images of an item.
15. **Viewer.** A program which parses a COF Images File and decompresses and displays an individual (TIFF or IOCA) image. Alternatively, a COF **reader** might parse the Images File and then pass the data for an individual (TIFF or IOCA) image to the **viewer** through a memory buffer or hard disk file. It may be appropriate to allow multiple viewers (each appropriate to a different image file format and image compression algorithm) to be invoked from within a COF reader.

7. Compliance

The description "COF version X.Y compliant" is to be applied to

- COF writing programs ("writers")
- COF reading programs ("readers") and
- COF media

only as described in this section.

A version number is to always be used in such a description to avoid confusion.

A version number comprises two parts:

- a major version number (the number "X" in version X.Y) and
- a minor version number (the number "Y" in version X.Y).

Minor version number changes are those deemed unlikely to cause difficulties for reading programs having a lower minor version number but the same major version number, providing such reading programs are written to anticipate likely areas of change (see below).

Changes in the major version number are used to indicate fundamental changes in the format likely to cause difficulties for reading programs having a lower major version number.

7.1 Requirements on Writers

Writers which are listed as "COF version X.Y compliant" shall have a default behavior of creating "COF version X.Y compliant" media. They may also create COF media having lower version numbers (both major and minor) in response to a specific request from a financial institution for such media.

While a specific installation of a writing system may not be capable of writing all media types, the writer application and operating system shall be capable of creating any of the media types listed in the COF version by simple installation and configuration of the appropriate tape drive or CD-ROM writer and their associated hardware. The writer application shall be capable of creating both the CD-ROM and tape variants of the COF logical format.

7.2 Requirements on Readers

Readers which are listed as "COF version X.Y compliant" shall properly interpret a piece of COF-compliant media having the major version number X and a minor version number less than or equal to Y.

Proper design of a reader (in anticipation of likely areas of change, see below) should make it possible for it to fully read and partially interpret a piece of COF-compliant media having the major version number X and a minor version number greater than Y.

7.3 Requirements on Media

Media described as "COF version X.Y compliant" shall meet all requirements of the X.Y version of this standard, including media type and physical labeling.

In the absence of a media preference among the allowed media types by the generating Federal Reserve or by the receiving financial institution, the recommended media types are CD-ROM and/or DLT™ 20 Gigabyte tape.

1.4 Likely Areas of Change

Likely areas of change in future revisions of the COF include:

- added or removed database fields in the Data Files
- added fields in the Images Files, such as new View parameters
- added parameters in the various header and trailer files
- added image compression types or file formats
- added media types, with regard to physical dimensions, capacity and encoding methods
- dropped media types

8. Overall Structure

A COF-compliant piece of media contains several kinds of files:

- one Media Header File,
- one or multiple File Suites,
- optional files (see restrictions in *Section 8.2 Optional Files*), and
- one Media Trailer File

Regardless of the media type, the logical formatting of the data is identical with the following exceptions:

1. The File Suites of COF-compliant tapes do not contain Data Files or their associated Index Files.
2. Tape media have a single file which is the concatenation of the individual files in the interchange in appropriate order. (see *Section 14.3 Media-Specific Issues: Tape*)

8.1 File Naming

On operating systems where upper-case and lower-case file names are distinguished, the required filenames shall use upper-case.

All required files shall be located in the root directory of a CD-ROM.

The **Media Header File** will be named HEADER.TXT.

The **Media Trailer File** will be named TRAILER.TXT.

Each **Data File** will be named COFxxxx.DBF, where xxxxx is an ASCII string representing a 5-digit decimal number having leading zeroes. This number may be between 00001 and 99999, inclusive. The first Data File on a **set** of media will be named COF00001.DBF (where the string "COF" uses the ASCII letter "Oh", not the numeral zero). Note that Data Files do not appear on tape media.

Each **Images File** will be named COFxxxx.IMG, where xxxxx is an ASCII string representing a 5-digit decimal number having leading zeroes. This number may be between 00001 and 99999, inclusive. The first Images File on a **set** of media will be named COF00001.IMG (where the string "COF" uses the ASCII letter "Oh", not the numeral zero).

A Data File and its corresponding Images File will have file names whose numerical components are equal, for example COF00001.DBF and COF00001.IMG. Note that Data Files do not appear on tape media.

For a discussion of the naming of the other files in a **File Suite**, please see *Section 8.3 File Suites*.

The set of **Index Files** corresponding to any one Data File are named as described in *Section 12*.

Index Files Structure. Note that Index Files do not appear on tape media.

The numbering of the files in File Suites continues across the various pieces of media in a set of media. For example, if the last Data File on the first piece of media is COF00006.DBF (contained in the sixth File Suite), the first Data File on the second piece of media in the media set will be COF00007.DBF (contained in the seventh File Suite).

For naming of optional files, see the next section.

The file name of each of the various COF files is replicated at the top of that file, enclosed in double square brackets, in all upper-case characters, as per the following example:

```
[FileInfo]  
FileName=COF00002.DBF
```

The FileInfo information, while present in files of both the CD-ROM and tape variants of COF, is to assist in the parsing of tape interchanges. These consist of single files that are formed by the concatenation of the various individual files present on a CD-ROM.

8.2 Optional Files

No optional files will be placed on media generated by systems within the Federal Reserve.

It is recognized that commercial uses of the Common Output Format may wish to introduce extensions to it, for instance to transmit additional fields describing items. The restrictions in this section govern what extensions may be introduced as optional files for commercial uses while still remaining COF compliant.

Optional, non-standard files may be included, but a writer is cautioned that a reader may be likely to ignore such files.

Optional files may have any name which does not conflict with the other file names, subject to the requirements of this section. Any optional files shall have file names that are unique across the Media Set of which they are a part.

An example of a useful optional file might be a database program for searching or verifying the correct format of the Data Files. Note that Data Files do not appear on tape media.

Another example would be a program which accepts an offset into the Images File from the database program and extracts a standard individual image file to the local hard disk.

Yet another useful optional file might be one or more viewing programs for viewing standard individual image files which have been extracted from the Images File.

Optional files containing item data are **required** to use the DBF file format and any associated index files must be in the NDX index file format (if index files are used). This ensures that receiving applications from other vendors will be able to use the information. If **any** optional files are placed on the media, a file called OPTIONAL.TXT must be included on that same piece of media which will describe in English prose all optional files placed on that piece of media. This will describe the uses of any such optional item data fields placed in any optional DBF or NDX file. The naming of such files shall mirror the naming of the other, required files in the same File Suite, i.e. their names will incorporate the same 5 digit number string used in the required files of that File Suite.

If any optional executable programs have been placed on the media, the environment required by

those programs will be described in the OPTIONAL.TXT file of that piece of media along with basic instructions for invoking the program. It is recommended that such optional executable programs be placed on the first of or on all of the pieces of media in a media set.

8.3 File Suites

A single piece of media may carry multiple File Suites of Images Files, Data Files and their associated Index Files (for CD-ROM media) and Header and Trailer files. For example, a piece of media containing three File Suites would have the following files in addition to the Media Header and Media Trailer files.

Header Files	Data Files *	Images Files	Index Files *	Trailer Files
HDR00001.TXT	COF00001.DBF	COF00001.IMG	several	TRL00001.TXT
HDR00002.TXT	COF00002.DBF	COF00002.IMG	several	TRL00002.TXT
HDR00003.TXT	COF00003.DBF	COF00003.IMG	several	TRL00003.TXT

* Not present on tape media.

On CD-ROM media, each Data File has associated with it a set of Index Files with related numbering. See *Section 12. Index Files Structure*.

The Media Header File identifies all the File Suites that are present on a given piece of media. See *Section 9. Media Header File Structure*.

The multiple File Suite mechanism allows the building of the Images Files before the target media has been chosen. Choosing an Images File size which is large enough to be media-efficient but not too large allows the multiple File Suites from a day's work to be placed on, say, either nine CD-ROMs or two 8mm tapes, without re-running the formatting step.

It is recommended that a single File Suite not be larger than can fit on a CD-ROM, even when using other, larger media. It is recommended that vendors of COF writing systems parameterize the size under which File Suites are kept in order to adjust its value to suit different applications.

It is recommended that File Suites not be made excessively small, resulting in too many File Suites on a single CD-ROM. Some operating systems have difficulty dealing with large numbers of files (100 to 1000) in a single directory.

8.3.1 Format of .ini Files

Several of the files included in the COF use the format of Window 3.1 .ini files, for which simple and flexible Windows 3.1 and Windows 95 programming functions exist (GetPrivateProfileInt, GetPrivateProfileString).

The information in this section is largely drawn from the winini.wri file found in the windows directory of a Windows 3.1 installation.

These files contain text lines which are terminated by <cr><lf> (carriage return, line feed) pairs.

The .ini files contain several sections, each of which consists of a group of related settings. The sections and settings are listed in the file in the following format:

```
[SectionName]  
KeyName=value
```

In this example, [SectionName] is the name of a section. The enclosing brackets ([]) are

required, and the left bracket must be in the leftmost column.

The ordering of sections and of keynames within sections does not matter in such files. Blank lines are permitted anywhere.

The KeyName=value statement defines the value of each setting. A keyname is the name of a setting. It can consist of any combination of letters and digits in uppercase or lowercase, and it must be followed immediately by an equal sign (=) with no spaces. The value can be an integer, a string, or a quoted string, depending on the setting.

Optional or non-applicable keynames may have the form
 KeyName=
or be absent entirely.

Note the capitalization of and the lack of spaces within the parameter names.

Comments may be included in initialization files. Each line of a comment must begin with a semicolon (;) in column one.

Numbers are decimal values unless a "0x" (zero, lower-case x) prefix is used, which indicates a hexadecimal number.

When reading either floating point values such as the COFVersion= or arrays of values such as FileSuiteCheckSums=, readers are cautioned to read these using the GetPrivateProfileString function rather than the GetPrivateProfileInt function and to parse them properly from the returned string.

8.3.2 File Suite Header Structure

The File Suite Header for a given File Suite will be named HDRxxxxx.TXT, with xxxxx corresponding to the five digit number used in the file names of the other files in the File Suite.

The File Suite Header File contains summary information relating to all of the other files within the File Suite. Its structure mimics that of Windows 3.1 ".ini" files. See *Section 8.3.1 Format of .ini Files* for the requirements of this format.

```
[FileInfo]
FileName=HDR00006.TXT
```

```
[FileSuiteInfo]

FileSuiteNumber=6
FileSuiteDescription=Any text up until the first carriage return.
; note this is an optional field
```

8.3.3 File Suite Trailer Structure

The File Suite Trailer for a given file will be named TRLxxxxx.TXT, with xxxxx corresponding to the number used in the file names of the other files in the File Suite.

The File Suite Trailer File contains summary information relating to all of the other files within the File Suite. Its structure mimics that of Windows 3.1 ".ini" files. See *Section 8.3.1 Format of .ini Files* for the requirements of this format.

```
[FileInfo]
```

FileName=TRL00006.TXT

[FileSuiteInfo]

FileSuiteNumber=6

[FileSuiteVerificationInfo]

; In the parameters that follow, byte counts are in decimal and
; checksums are in hexadecimal

; The next two parameters are absent, blank or zero on a tape

;

DataFileBytes=12345

DataFileChecksum=0xAB45

ImagesFileBytes=12345

ImagesFileChecksum=0x4574

; All of the following Index File related parameters are absent, blank or zero on a tape

;

PRDIndexFileBytes=12345

PRDIndexFileChecksum=0x2784

RTNIndexFileBytes=12345

RTNIndexFileChecksum=0x4578

ACTIndexFileBytes=12345

ACTIndexFileChecksum=0xB578

CKNIndexFileBytes=12345

CKNIndexFileChecksum=0xA784

PRCIndexFileBytes=12345

PRCIndexFileChecksum=0x2784

CKAIndexFileBytes=12345

CKAIndexFileChecksum=0x2784

ISNIndexFileBytes=12345

ISNIndexFileChecksum=0x5784

The checksums are created by summing the contents of all of the bytes in each described file into a single 32 bit number for that file, with carries out of the 32 bits thrown away.

Neither the checksums nor the file byte counts should contain commas.

8.4 Media Sets

Multiple pieces of media constituting a single delivery may be associated into a larger entity called a **Media Set**. Typically this mechanism would be used to place a large number of items for a given period (often a day) for a given customer onto multiple pieces of media, while still indicating their related nature.

This mechanism, along with the multiple File Suites mechanism (see *Section 8.3 File Suites*)

facilitates flexible targeting of a large day's work onto either a (large) tape or multiple (smaller) CD-ROMs which are associated as a Media Set.

Multiple tapes may also constitute a Media Set. This method of associating multiple tapes is simpler and more reliable than tape label based schemes for writing across multiple tapes.

Adhesive labels carry information ("Media Set Info:") which indicates an individual piece of media's relationship to its Media Set (for example, "2 of 4"). See *Section 14.1 Physical Labeling Requirements*.

The files placed on CD-ROM and tape media are identical, with the exception that tape media do not contain Data Files or Index Files; nor do they include parameters within their header or trailer files associated with the Data Files and Index Files.

The Media Header File also carries information about the set relationship in electronic form. See *Section 9. Media Header File Structure*. The Media Trailer also contains information about the Media Set. See *Section 10. Media Trailer File Structure*.

While there exists a relationship among those media which are members of the same Media Set (they all come from the same period's work and are destined for the same bank), each piece of media stands alone in that each piece of media has a one-to-one correspondence between its contained images and the items in its Data Files. (Note that tape media do not include Data Files.) No Data File item refers to an image on another piece of media, nor is any individual image on a given piece of media referred to by a Data File on another piece of media.

8.4.1 Media Sources

In some applications, it may be necessary to generate Media Sets in parallel from more than one machine at the same site. For example, the required data throughput may exceed the write rate of the tape drive or CD-ROM drive. Each of these generators of Media Sets is called a **Media Source**. Each site may have multiple Media Sources, each distinguished by a site-unique ID called a MediaSourceID.

The MediaSourceID appears in the Media Header File of every piece of media in every Media Set. It may be an integer between 1 and 32767.

9. Media Header File Structure

The Media Header File contains summary information relating to all of the other files found on the media. Its structure mimics that of Windows 3.1 ".ini" files. See *Section 8.3.1 Format of .ini Files* for the requirements of this format.

The Media Header File is named HEADER.TXT.

```
[FileInfo]
FileName=HEADER.TXT
```

```
[General]
```

```
Date=19960430
From=FRB Boston
To=Bank of Erehwon
```

COFVersion=1.3
MediaSourceID=3
; unique ID for this media generator at this site
MediaSetNumberOfTheDayForThisMediaSource=1

[MediaFileSuitesInfo]

FirstFileSuiteNumber=7

[MediaSetInfo]

MediaSetMemberNumber=2
; this identifies which number this piece of media is within the media set.

The COFVersion may be treated as a floating point number. It will never contain a second decimal point.

10. Media Trailer File Structure

The Media Trailer File contains summary information relating to all of the other files found on the media. Its structure mimics that of Windows 3.1 ".ini" files. See *Section 8.3.1 Format of .ini Files* for the requirements of this format.

The Media Trailer File is named TRAILER.TXT.

[FileInfo]
FileName=TRAILER.TXT

[MediaVerificationInfo]

NumberOfFileSuites=3
; this identifies the number of file suites on this piece of media
; checksums are in hexadecimal
FileSuiteCheckSums=0x1234, 0x7E93, 0x3A6B

; byte counts are in decimal
FileSuiteByteCounts=4323479, 123485, 348765

[MediaSetInfo]
LastFileSuiteNumber=9
LastMemberInMediaSet=yes

[ImageTypeInfo]
; this section warns a reader what image types are present on the piece of media
ContainsLOCAGray=no
ContainsLOCABinary=yes
ContainsTIFFJPEG=yes
ContainsTIFFG4=no

Each FileSuiteChecksum is created by summing the checksums for each of the individual files in the File Suite (including the File Suite Header File and File Suite Trailer File) into a single 32 bit number for the entire File Suite, with carries out of the 32 bits thrown away. The checksums for the individual files are created as described in *Section 8.3.3 File Suite Trailer Structure*.

11. Data File Structure

Each Data File has the same structure. The Data File structure described in this section applies only to CD-ROM media.

Data files occur only on CD-ROM media, not on tape media. The financial information associated with an individual item is also placed, in the form of a flat-file row of text, into the Images File along with each item. This provides a second means to access the same data on a CD-ROM and the only means for a piece tape media. A separate Data File is not useful on a sequential medium like tape.

A COF Data File complies with the somewhat fluid industry-standard format for PC-based databases known as DBF (Data Base Format).

The DBF format was originally defined by Ashton Tate (now part of Borland International) for use by its dBASE family of products. Many vendors have built products using the so-called xBASE programming language and use variants of a DBF file as their native database file format.

Many database or database toolkit vendors have introduced proprietary extensions to the DBF file format and their claims about producing industry-standard output are to be taken with caution. In addition, such toolkits may not produce the appropriate version of DBF file required by this standard.

An ANSI committee has attempted to standardize both the xBASE language and the DBF file format, but has not completed its work. The closest we have found to a DBF format standard is the document attached as Appendix A. It documents the formats used in succeeding generations of the dBASE products.

11.1 Detailed Structure of the Data File

Each piece of CD-ROM media will contain at least one Data File. Each Data File will exist in a one-to-one correspondence with an Images File.

All Data Files will be compliant with the DBF file format used by the dBASE III and dBASE III PLUS products originally produced by Ashton Tate. DBF files shall be formatted as documented in this section.

NOTE TO IMPLEMENTORS: Implementors are cautioned that libraries or tools which are specified as capable of producing "DBF files" may not in fact produce dBASE III compatible DBF files; many database and database tools vendors have introduced variations on the DBF format which are incompatible. This burden falls primarily on vendors of COF generating (writing) software; it is expected that most tools will properly read this flavor of "DBF file", since it is the most generic and best documented of the flavors.

11.1.1 Data File Fields

The Data File incorporates the fields identified in the following table with the indicated ordering, field names, types and lengths.

Table 1. Fields of the Data File

Common Name	X9.37 Field Length, Type	DBF File Field Name	DBF File Field Length, Type, Indexed?	Definition
Item Sequence Number (Document Identification Number)	15, NB	ItmSeqNum	15, N, yes (Note 1)	A number assigned by the MICR capture system. May also be the same as the microfilm sequence number (Note 2).
Capture Site Indicator	9, N	CaptSite	9, N, no	The identifier of the site that captures the images (Note 3).
Process Date	8, N	ProcDate	D, implicit, yes	The year, month, and day the MICR file is captured (YYYYMMDD).
Capture Date	8, N	CaptDate	D, implicit, no	The year, month, and day the image is captured (YYYYMMDD).
Routing and Transit Number	9, N	RoutTrNum	9, N, yes	The number that identifies the institution by or through which the item is payable (Note 4).
Account Number	17, N	AcctNum	17, N, yes	Data specified by the payor bank. Sometimes called On-U.S. Data usually consists of payor's account number, a serial number or transaction code, or both (Note 5).
Check Number	6, N	CheckNum	6, N, yes	Check number (Note 6).
Process Control	6, N	ProcCtrl	6, N, yes	Process Control (Note 6).
Aux On US	15, NBSM	AuxOnUs	15, C, no (Note 7)	A code used on commercial checks at the discretion of the payor bank (Note 8).
EPC	1, N	EPC	1, N, no	A code used for special purpose as authorized by the Accredited Standards Committee X9. Sometimes known as position 44 (Note 9).
Amount	10, N	CheckAmt	10, N, yes (Note 10)	The US dollar value of the check expressed in cents (Note 11).
Item Offset in Images File	20, N	ItmImgOff	20, N	A number representing the number of bytes from the beginning of the Images File to the beginning of the item's section of the Images File (Note 12).

Table Notes

Note 1: The X9.37 data type NB means “numeric blank”, or characters from the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, blank}. The COF standard does not permit embedded blanks in a numeric field such as this one.

Note 2: The Item Sequence Number comes from either the on-us file transmitted from the bank or it uses the image sequence number provided by the image system if a high-speed sorter sequence number is not available.

Note 3: Typically the routing/transit number of the originating institution.

Note 4: The 8-digit routing/transit number plus check digit from positions 34-42 of the MICR line.

Note 5: Usually from positions 16-32 on the MICR line.

Note 6: The Check Number and Process Control fields may replicate the same information or one of them may be zero. This 6-digit field is usually taken from positions 13-18 of the MICR line. Typically, this on-us field contains the check number which usually is positioned between the amount and account number field delimiters.

Note 7: The X9.37 data type NBSM means “numeric blank special MICR”, or characters from the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, blank, asterisk, dash}. The special symbols are used in the following way:

asterisk represents a MICR character which could not be read.
dash represents the MICR symbol ‘dash’

Note 8: This number comes from the auxiliary on-us field in positions 47-64 of the MICR line.

Note 9: EPC is an abbreviation for Extended Process Control. This is from positions 44 or 45 of the MICR line. Typically, this field identifies a truncation item, a qualified check return, or an imageable truncation item.

Note 10: The CheckAmt field will never contain a decimal point and is therefore expressed in cents. It will never contain a dollar sign.

Note 11: The amount is taken from positions 2-11 of the MICR line.

Note 12: The beginning of an item’s section in the Images File is indicated by the byte offset (from the beginning of the Images File) of the capital letter “I” at the beginning of the string “Item”. If no images exist for the item, the Item it points to shall simply contain no views. In this case, the ItemViewLoopHeader <cr><lf> is followed immediately by the ItemViewLoopTrailer <cr><lf>. See *Section 13. Images File Structure*.

The following table indicates the allowed values in the various field data types used in the DBF files under this standard. More information about this is found in *Table 5. DBF File Records*.

Table 2. Allowed Values of DBF File Data Field Types

DBF File Data Type	Data Input
C Character	Any ASCII characters.
D Date	8 ASCII digits in YYYYMMDD format.
N Numeric	ASCII representation of a number using: - (minus) . (decimal point) 0 1 2 3 4 5 6 7 8 9

11.1.2 dBASE III Compatible DBF File Format

The DBF file format for the dBASE III and dBASE III PLUS products is documented in a file located on the Borland International World Wide Web site. This document is reproduced in its entirety as Appendix A. The portion of that document which specifies the structure of dBASE III PLUS compatible DBF files has been re-formatted here for clarity. That information is described as coming from the *Using dBASE III PLUS* manual, Appendix C.

An useful reference is the book *The File Formats Handbook* by Günter Born (see references). This book documents the dBASE III and dBASE III PLUS DBF file format and appears to be consistent with the Borland file.

Another reference is the book *File Formats for Popular PC Software. A Programmer's Reference* by Jeff Walden (see references). This book documents the dBASE III DBF file format and appears to be consistent with the Borland file.

A useful reference on DBF files is the DBUTIL library public domain source code developed at Lawrence Berkeley Laboratories by Nathan Parker (ngparker@lbl.gov) for the reading of Census Bureau CD-ROMs in DBF format. Information on this package is located at:

<http://cedr.lbl.gov/data1/dbutil/doc/DBUTILSourceCode-doc.html>

In particular, the file *db.h* contains a lot of useful information. Implementors are cautioned that in cases where that code disagrees with this standard, then this standard holds precedence.

In this section, numbers appearing with a leading "0x" are in hexadecimal format.

A DBF file consists of the following component parts:

1. One DBF File Header, which contains:
 - A variable number of DBF File Field Descriptors, one per field.
2. A variable number of DBF File Records, one per item.
3. An end-of-file terminator.

Table 3. DBF File Header

Offset	Length (bytes)	Description and Required Values
0	1	= 0x03 indicates valid dBASE III PLUS table file without an associated memo DBT file.
1-3	3	Date of last update; in binary YYMMDD format. example: first byte: 0x60 for 1996 second byte: 0x0A for October third byte: 0x1F for the 31st COF Note: COF Readers: Do not use this date. Better (4-digit year) date fields occur elsewhere in the COF.
4-7	4, 32-bit number	Number of records in the table. Intel order integer.
8-9	2, 16-bit number	Number of bytes in the header. Intel order integer. Includes 0x0D terminator of the header.
10-11	2, 16-bit number	Number of bytes in a data record. Intel order integer. This is always one more than the sum of all the field lengths, since the first byte of a record is always reserved for marking deleted records.
12-31	20	Reserved bytes. Must be written as 0x0 bytes. Ignore on read.
32	32 times the number of fields	Array of Field Descriptors. The size of the array depends on the number of fields in the table file. The structure of each Field

		Descriptor is shown in Table 4., below.
n+1	1	0x0D stored as the terminator for the header. n is the last byte in the Array of Field Descriptors.

Table 4. DBF File Field Descriptor

Offset	Length (bytes)	Description and Required Values
0-10	11	Field name in ASCII. Unused bytes after the name are filled with 0x0 bytes.
11	1	Field type in ASCII. C for character D for date N for numeric.
12-15	4	Field data address (not useful; memory address used by dBASE) Must be written as 0x0 bytes. Ignore on read.
16	1	Field length in bytes (binary). For character: exact field length (no terminators). For numeric: includes entire field, including decimal point and places after decimal point.
17	1	Field decimal count (binary). For numeric: number of places after the decimal point.
18-31	14	Reserved bytes. Must be written as 0x0 bytes. Ignore on read.

Table 5. DBF File Records

Requirements	
1.	The records immediately follow the 0x0D terminator of the DBF File Header.
2.	Each record begins with one byte which is a space (0x20) if the record is not deleted, or is an asterisk (0x2A) if the record is deleted.
3.	Fields are in the order of and obey the formatting specified in the Array of Field Descriptors in the DBF File Header.
4.	Fields are packed into records without field separators or record terminators.
5.	Character fields are in ASCII, without terminators. If the length of the text is shorter than the length of the field as specified in the corresponding Field Descriptor, blank characters (0x20) are placed into the remaining bytes ("trailing blanks"). COF Note: Where field values are unknown or undefined, they shall be set to zero in the DBF file.
6.	Numeric fields are in ASCII, without terminators. If the length of the numeric string is shorter than the length of the field as specified in the corresponding Field Descriptor, the string is right-justified in the field, with blank characters (0x20) replacing any leading zeros ("leading blanks"). Example: " 10003". Numeric fields used to express amounts shall not contain currency symbols (for example "\$") or decimal points. COF Note: Where field values are unknown or undefined, they shall be set to zero in the DBF file.
7.	A date field contains the date as an 8 character ASCII string in the format YYYYMMDD, without blanks, separators or terminators. "Y", "M", and "D" as used here, represent the appropriate characters from the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Example: "19970131".

Table 6. DBF File End-of-Valid-Data

Requirements	
1.	The DBF End-of-Valid-Data is a single byte, functioning as a terminator. This is an ASCII character value of 0x1A. This marker is placed there not by the operating system, but by the database or export application. Some operating systems may also place a 0x1A byte at the end of the file, but the writing application must also write one explicitly.
2.	There may be deleted records (marked by an asterisk in their first byte) located after the DBF End-of-Valid-Data marker and before the actual end of the file as defined by the operating system.

12. Index Files Structure

Accompanying each DBF format Data File are a set of Index Files; one Index File for each indexed data field. The name of each of the Index Files is derived from both its key field's name and from the name of its associated COF Data File as in the table below. Not all fields are indexed, only those with a yes in their entry in Table 1.

For a Data File named COFxxxxx.DBF (where xxxxx is a 5-digit string having leading zero digits if necessary), the following are the names of the Index Files. These are all required.

Table 7. Index File Names Derived from Key Field Names

Key Field Name	Index File Name
ProcDate	PRDxxxxx.NDX
RoutTrNum	RTNxxxxx.NDX
AcctNum	ACTxxxxx.NDX
CheckNum	CKNxxxxx.NDX
ProcCtrl	PRCxxxxx.NDX
CheckAmt	CKAxxxxx.NDX
ItmSeqNum	ISNxxxxx.NDX

The Index Files are in the so-called NDX file format which is associated with the DBF file format. The version of the NDX file used in this standard is that used by the dBASE III product from Ashton Tate, since this is the best documented and most widely supported of database index file formats.

12.1 Detailed Structure of Index Files

An NDX file is composed of multiple 512 byte pages.

The first page is a header page. It is followed by multiple index pages which make up a B-tree data structure.

Table 8. NDX File Header Page

Offset	Length (bytes)	Description and Required Values
0x00	4	Start key page (root page). Record number of root (starting) page of the B-tree. The byte offset of the root page from the start of file is this number times 512. The root page is not necessarily the first page after the header page. Intel order integer.

0x04	4	Total index pages. Does not include this header page. Intel order integer.
0x08	4	Reserved. Write as 0x0. Ignore on read.
0x0C	2	Index key length. The length of the key field upon which the NDX file is based. Numeric key fields are always 8 bytes long. Intel order integer.
0x0E	2	Maximum keys per page. Count of the maximum number of instances of this key which can fit within a single index page, based on the length of the key field. The actual number placed in a given index page (which may vary) is placed in a parameter in each index page. Intel order integer.
0x10	2	NDX key type. = 0x01 for a numeric key (including a date key) = 0x00 for an alphanumeric key Intel order integer.
0x12	4	Key record size. This is the byte offset between two consecutive key records within an index page. Intel order integer.
0x16	1	Reserved. Write as 0x0. Ignore on read.
0x17	1	Unique flag. Not used by this standard. Write as 0x0. Ignore on read.
0x18	488	Key name. This contains the name of the key field as an ASCII string. The remainder of the 488 bytes is filled with zeros.

Table 9. NDX File Index Pages

Offset	Length (bytes)	Description and Required Values
0x00	4	Key records in page. The number of key entries actually present in this index page. Intel order integer.
0x04	4	Left page number 1. This contains the page number of the page located to the left of this node in the B-tree. A page to the left in the B-tree will contain "all the keys that in the sort sequence are smaller than or the same size as the key required." [Born, p. 17].
0x08	4	DBF record number 1. If this is a leaf node in the data structure, this field is non-zero and contains the DBF record number within the associated DBF file where the item is located. To calculate a byte offset within the DBF file of the item's record, use the record length from the DBF file header and the size of the DBF file header. If this parameter is zero, this is not a leaf node, and the B-tree data structure must continue to be traversed to find this item's DBF record number. Intel order integer.
0x0C	length	Key data 1. The actual key data is stored in this parameter. The length of all these fields was specified in the NDX header page in the Index key size parameter. For an alphanumeric key, the data is in ASCII. When the key is smaller than the index key size, the remaining bytes are filled with blanks (right blank filled). For a numeric key (including date keys), the key is represented as an 8 byte IEEE floating point number.
...	4	Left page number 2.
...	4	DBF record number 2.
...	length	Key data 2.
- - -		
...	4	Left page number n. Where n is given by the Key records in page parameter at the top of the page.
...	4	DBF record number n.
...	length	Key data n.

13. Images File Structure

Within the Images File may be placed multiple TIFF Group 4, TIFF JPEG, IOCA ABIC Binary, and IOCA ABIC Grayscale images. While these are referred to as image files when they stand alone, when they become a part of a COF "Images File", it is clearer to refer to them as TIFF and IOCA data streams, since they no longer exist as separate files. All of the different image data stream types may occur within one single Images File. The Media Header File contains information to forewarn a receiving application of which image data stream types may be present.

13.1 Supported Image Compression Types

The following image compression types are supported by the Common Output Format:

1. CCITT T.6 (Group 4)
2. JPEG Baseline DCT
3. ABIC Arithmetic Binary Image Coding for binary images
4. ABIC Arithmetic Binary Image Coding for grayscale images

13.2 Supported Image File Types

X9.46 has a means of indicating that an image file with its own format or header exists within the X9.46 formatting. This is referred to as the embedded header mechanism.

Subsets of each of the identifiable file formats in X9.46 have been defined which are acceptable for use with the embedded header mechanism of X9.46. These same restrictions are in the Common Output Format.

In broad outline, these restrictions were intended to ensure that the raw compressed data contained within the various file format wrappers could be referenced directly from an X9.46 pointer without knowledge of which wrapper had been used.

An example is to disallow TIFF's (optional) use of *strips* to break compressed data into more easily accessed chunks of separately compressed data. This cannot be used under the X9.46 embedded header approach, since a decompressor which is not TIFF-aware would think the image had ended at the first such strip boundary.

Similarly, ABIC images support compressed data only in 64 KByte chunks, called segments. An ABIC decompressor which is not IOCA-aware would think the image ended at the first such segment boundary.

13.3 Aggregation Method

The aggregation method is the technique used to bind multiple images together into a single file. Two levels of aggregation are important:

1. Keeping the front and back images associated.
2. Keeping the interchange from having too many separate files.

13.3.1 Relation to X9.46 Approach

X9.46 uses two nested looping mechanisms to place multiple images (called views) of multiple checks (called items) into a single combined interchange. Since this is the approach which will be implemented in an X9.46 environment, the COF adopts this structural approach. As in X9.46, the financial data associated with an item is replicated adjacent to the images of an item, despite the presence of this item data elsewhere in the interchange.

Specific aspects of the X9.46 syntax are not adopted, however. For example, the ANSI X12 EDI

character sequences used for the loop headers are not used. Instead, literal character sequences or tokens are used as seen in the next section.

An additional difference from X9.46 is the omission of the Item View Data, which would occur in X9.46 between a view and the actual image file of the view. The Item View Data would (if it were present) replicate all the image data parameters found in the TIFF tags or IOCA attributes of the front image file and therefore is not needed in this standard.

13.4 Detailed Images File Structure

An Images File uses ASCII character strings called tokens to delineate its structure. These tokens are separated from one another and from image data streams by a carriage return character and a linefeed character. These are indicated in the example below as <cr><lf>. Any additional white space (spaces, tabs) between these tokens is to be ignored.

The tokens are case sensitive and may not have embedded spaces.

Binary data streams containing individual images are interspersed with these strings. The binary data streams have lengths associated with them which permit navigation over them.

Comments may be inserted in the following way. Comments are indicated by the occurrence of a semicolon (“;”) immediately following the <cr><lf> pair of the preceding line. The comment field continues until the next occurrence of a <cr><lf> pair. Note that the indented form shown below for clarity does not adhere to this requirement on comment fields; all comment fields would have to be outdented to column 1 in a valid Images File. Only printable ASCII characters may be present in comments.

Here is an example Images File

```
[FileInfo] <cr><lf>
FileName=COF00001.IMG <cr><lf>
ItemLoopHeader <cr><lf>
    ; example comment: The "Item Offset in Images File" points to <cr><lf>
    ; the "I" in the word "Item" on the next line. <cr><lf>
Item <cr><lf>
    ItemData <cr><lf>
        ;ItemData goes here.
        ;See Section 13.4.1 ItemData in Images Files.
    ItemViewLoopHeader <cr><lf>
        ; example comment: views from an XYZ transport <cr><lf>
View <cr><lf>
    ;even the first view is optional
ViewSide <cr><lf>
    Front <cr><lf>
ViewType <cr><lf>
    TIFFJPEG <cr><lf>
ViewLength <cr><lf>
    32468 <cr><lf>
TIFF file first byte
...
TIFF file last byte
View <cr><lf>
    ;additional views beyond the first are optional <cr><lf>
ViewSide <cr><lf>
    Front <cr><lf>
ViewType <cr><lf>
```

```
        IOCABinary <cr><lf>
ViewLength <cr><lf>
    32400 <cr><lf>
IOCA file first byte
...
IOCA file last byte
View <cr><lf>
;additional views beyond the first are optional <cr><lf>
ViewSide <cr><lf>
    Back <cr><lf>
ViewType <cr><lf>
    IOCABinary <cr><lf>
ViewLength <cr><lf>
    16971 <cr><lf>
IOCA file first byte
...
IOCA file last byte
View <cr><lf>
;additional views beyond the first are optional <cr><lf>
ViewSide <cr><lf>
    Back <cr><lf>
ViewType <cr><lf>
    TIFFG4 <cr><lf>
ViewLength <cr><lf>
    10973 <cr><lf>
TIFF file first byte
...
TIFF file last byte
ItemViewLoopTrailer <cr><lf>
Item <cr><lf>
ItemData <cr><lf>
    ; ItemData goes here.
    ; See Section 13.4.1 ItemData in Images Files.
ItemViewLoopHeader <cr><lf>
View <cr><lf>
;even the first view is optional
ViewSide <cr><lf>
    Front <cr><lf>
ViewType <cr><lf>
    IOCAGray <cr><lf>
ViewLength <cr><lf>
    12972 <cr><lf>
IOCA file first byte
...
IOCA file last byte
View <cr><lf>
;additional views beyond the first are optional <cr><lf>
ViewSide <cr><lf>
    Back <cr><lf>
ViewType <cr><lf>
    TIFFJPEG <cr><lf>
ViewLength <cr><lf>
    16381 <cr><lf>
TIFF file first byte
```

```

    ...
    TIFF file last byte
ItemViewLoopTrailer <cr><lf>
    ...
ItemLoopTrailer <cr><lf>
```

An item may have no views or it may be desired to not include the views in the interchange. In the case where no views are interchanged for an item, the ItemViewLoopHeader <cr><lf> is immediately followed by the ItemViewLoopTrailer <cr><lf> (possibly with intervening white space or comment lines). This case may occur in an "image-to-follow" ECP situation where the MICR data is sent in a first COF interchange without any images.

Items may have multiple views for each side.

Each view has a ViewType which may be indicated by one of the following tokens:

- IOCAGray
- IOCABinary
- TIFFJPEG
- TIFFG4

Each view has a ViewSide which indicates whether the view is of the front face or of the back face of the item. The following tokens are used:

- Front
- Back

Each view has a ViewLength. This is an ASCII-formatted decimal number indicating the exact length of the sequence of individual image file bytes for a single view. The first byte of the TIFF or IOCA data stream representing the view occurs immediately after the <cr><lf> pair which follows the length. Thus, the tab and space characters used in the example above to improve readability could all be present in an actual interchange, except those tab and space characters between the ViewLength number and the actual image data bytes. (The additional tab and space characters may also not be present before the semicolon introducing a comment line.) An application parsing an Images File would be wise to check the "magic numbers" for TIFF and IOCA files to confirm that the proper starting byte of the image data stream has been found. These are 0x49 or 0x4D for TIFF files and 0x70 for IOCA files.

A modular design of COF reader software is recommended, with (possibly multiple) separate viewers being invoked to display the extracted image data stream.

13.4.1 ItemData in Images Files

In the description of the structure of Images Files in the prior section, the construction

```
    ; ItemData goes here
```

was used. The current section describes the format of the ItemData line which appears near the top of each Item structure in the ItemLoop before the ItemViewLoopHeader.

The ItemData contains the database information for the row corresponding to a given item. On CD-ROM media, this information is required to be consistent with the information in the corresponding row of the DBF format Data File. This information is formatted in a "flat file" form, specifically, as ASCII strings delimited by single tab characters. The ordering of the fields across the line mirrors the ordering of the rows in Table 1.

Numeric fields are decimal numbers and may or may not be padded to their specified length by means of leading zeros.

Character fields may or may not be padded to their specified length by means of trailing blanks.

Date fields have the format YYYYMMDD, with all character positions using ASCII numeric digits.

Here is an example of such an ItemData row. In the example, numeric fields and character fields have been padded to their specified lengths by means of leading zeros and trailing blanks, respectively. Additional line breaks have been introduced for readability in this text, but are not to be present in actual ItemData. Tab characters are shown as <tab>. A comment line is present which assists with identifying the fields.

```
; Item Sequence Number      Capture Site Identifier    Process Date
  Capture Date              Routing and Transit Number Account Number
  Check Number              Process Control            Aux On Us
  EPC                       Amount                     Item Offset in Images File
  <cr><lf>

000000123456789<tab>      123456789<tab>          19970110<tab>
19970109<tab>             234567890<tab>          1234567890123456<sp><tab>
001234<tab>               001234<tab>             5432112345678<sp><sp><tab>
2<tab>                    0000002114<tab>         0000000001234567890<tab>
<cr><lf>
```

Note the final <tab><cr><lf>.

13.5 Detailed TIFF File Structure

The term "TIFF file" is used here to refer to the self-contained TIFF data stream for a single image, but it is understood that individual TIFF files do not appear on the media for reasons of efficiency. Instead, these individual TIFF files are concatenated together, along with additional structure, into an Images File. There may be multiple Images Files on a piece of media containing multiple File Suites.

13.5.1 General Notes on TIFF Tags

1. ImageWidth and ImageLength need not be multiples of 8 and reading applications should not make that assumption.
2. For the purposes of this standard, only the first image in a TIFF file will be treated as an interchanged image. Front and back images must each be in their own TIFF files, not placed in a single multi-page TIFF file.
3. Despite the TIFF recommendation that strips be less than 8K (TIFF 6.0 Specification, page 39), common practice in check imaging is to use a single strip, no matter how large the compressed data size. The COF requires writers to generate files having a single strip.
4. If the default value for a given tag applies, the tag need not be present.
5. TIFF files may be in either Intel or Motorola byte order. As a reminder, the "II" or "MM" indicators of byte order apply only to tag information, not to the compressed or uncompressed image data.

13.5.2 Group 4 Compressed Binary Files

The following tags are mandatory and (unless defaulted) must appear in numerical order by tag number within the TIFF file. This list is from the table on page 21 of the TIFF Specification (see

references) and constitutes the requirements for Baseline TIFF Bilevel Images.

Beyond the requirements for Baseline TIFF Bilevel Images, the Orientation tag has also been included. COF readers must properly interpret any of the 8 possible values of the Orientation tag and, after decompressing, must paint the raster data in the appropriate directions for proper viewing.

Table 10. Required Tags for Group 4 Compressed TIFF Files

Tag Number	Tag Name	Required Value
256	ImageWidth	Any value
257	ImageLength	Any value
258	BitsPerSample	1
259	Compression	4
262	PhotometricInterpretation	0
273	StripOffsets	This array will have only one entry
274	Orientation	Any valid value
278	RowsPerStrip	Must be equal to ImageLength
279	StripByteCounts	This array will have only one entry
282	XResolution	Any value
283	YResolution	Any value
296	ResolutionUnit	Any valid value

TIFF requires readers to gracefully ignore any optional tags which they do not expect or understand, so in a sense, any other additional tags are allowed. It would be unreasonable to expect the recipient to properly interpret any tags beyond those listed in this section, however.

13.5.2.1 Notes Regarding the Group 4 Image Tags

1. The FillOrder tag is not listed as necessary since the default is FillOrder = 1, which specifies bits are placed into a byte in the MSB to LSB direction. This standard specifically does not permit the FillOrder = 2 ordering, which is LSB to MSB.
2. The omission of the T6Options tag implies its value is its default, T6Options=0. This standard specifically does not permit the value T6Options=1. This means that the obscure and little-used "uncompressed mode" of ITU-T Recommendation T.6 (Group 4) is not allowed.

13.5.3 JPEG Files

The following tags are mandatory and (unless defaulted) must appear in numerical order by tag number within the TIFF file. This list is derived from the table on page 22 of the TIFF Specification (see references) and in large part constitutes the requirements for Baseline TIFF Grayscale Images.

Beyond the base requirements for TIFF JPEG Images, the Orientation tag has also been included. COF readers must properly interpret any of the 8 possible values of the Orientation tag and, after decompressing, must paint the raster data in the appropriate directions for proper viewing.

Table 11. Required Tags for JPEG Compressed TIFF Files

Tag Number	Tag Name	Required Value
256	ImageWidth	Any value
257	ImageLength	Any value
258	BitsPerSample	8
259	Compression	6
262	PhotometricInterpretation	1
273	StripOffsets	This array will have only one entry
274	Orientation	Any valid value
278	RowsPerStrip	Must be equal to ImageLength
279	StripByteCounts	This array will have only one entry
282	XResolution	Any value
283	YResolution	Any value
296	ResolutionUnit	Any valid value
512	JPEGProc	1
513	JPEGInterchangeFormat	Byte offset into the TIFF file of the 0xFF byte of the JPEG Start of Image (SOI) marker code. Since this standard only supports JPEG data in JPEG Interchange Format as defined in Annex B of the JPEG standard, this tag may not be 0 and thus must be present.
514	JPEGInterchangeFormatLength	Length in bytes of the JPEG Interchange Format data stream.
515	JPEGRestartInterval	If restart markers are present in the JPEG Interchange Format data stream, this value tells the number of Minimum Coded Units (MCUs) between restart markers. If no restart markers are used, this value shall be 0 or the entire tag may be omitted, since 0 is the default value of the tag.
519	JPEGQTables	This field points to the byte offset within the JPEG data stream of the marker for the single quantization table used with a grayscale image.
520	JPEGDCTables	This field points to a single offset within the JPEG data stream of the marker for the single DC Huffman table used with a grayscale image.
521	JPEGACTables	This field points to a single offset within the JPEG data stream of the marker for the single AC Huffman table used with a grayscale image.

TIFF requires readers to gracefully handle any tags which they do not expect or understand, so in a sense, any additional tags are allowed. It would be unreasonable to expect the recipient to properly interpret any tags beyond those listed in this section, however.

13.5.3.1 Notes Regarding the JPEG TIFF Tags

1. An emerging new approach to JPEG in TIFF uses a Compression tag value of 7,

versus the value of 6 used in TIFF 6.0. It is strongly recommended that readers of COF Images Files anticipate the migration of JPEG under TIFF to this simpler approach and support both Compression=6 and Compression=7. Under the new approach, no JPEG attributes are broken out into tags (as in the TIFF 6.0 scheme), since this replicates the information found down in the JPEG data stream. This approach to JPEG within TIFF is discussed in the TIFF Technical Note Number 2. For further information, please see the URLs:

<http://www.picturel.com/PDF/ttn2drft.pdf>, or
<http://www.picturel.com/TXT/ttn2drft.txt>

13.6 Detailed IOCA File Structure

The term "IOCA file" is used here to refer to the self-contained IOCA data stream for a single image, but it is understood that individual IOCA files do not appear on the media for reasons of efficiency. Instead, these individual IOCA files are concatenated together, along with additional structure, into an Images File. There may be multiple Images Files on a piece of media containing multiple File Suites.

13.6.1 ABIC Compressed Binary Files

The following attributes are mandatory:

Table 12. Required Attributes for ABIC Compressed Binary IOCA Files

IOCA Field	Parameter, length (bytes)	Allowed Values	Notes
Begin Segment	ID, 1	0x70	
	LENGTH, 1	0x00	
Begin Image Content	ID, 1	0x91	
	LENGTH, 1	0x01	
	OBJTYPE, 1	0xFF	IOCA object
Image Size Parameter	ID, 1	0x94	
	LENGTH, 1	0x09	
	UNITBASE, 1	0x00	10 inches is the unit used for the following resolution values.
	HRESOL, 2	0x0000 -- 0x7FFF	Horizontal resolution. A typical value is 2400 dots per 10 inches (describing 240 dpi), but no constraint is placed on this field.
	VRESOL, 2	0x0000 -- 0x7FFF	Vertical resolution. A typical value is 2400 dots per 10 inches (describing 240 dpi), but no constraint is placed on this field.
Image Encoding Parameter	HSIZE, 2	0x0000 -- 0x7FFF	Horizontal size in pixels.
	VSIZE, 2	0x0000 -- 0x7FFF	Vertical size in pixels.
	ID, 1	0x95	(Note 1)
IDE Size Parameter	LENGTH, 1	0x02	
	COMPRID, 1	0x08	ABIC (Note 2)
	RECID, 1	0x01	RIDIC (normal raster order).
IDE Structure Parameter	ID, 1	0x96	
	LENGTH, 1	0x01	
IDE Structure Parameter	IDESZ, 1	0x01	1 bit per pixel.
	ID, 1	0x9B	
	LENGTH, 1	0x06	
IDE Structure Parameter	FLAGS, 1	0x00	no Gray code.
	FORMAT, 1	0x02	YCrCb color space.

	SIZE1	0x01	1 bit (Note 3).
Image Data	ID, 2	0xFE92	
	LENGTH, 2	0x0001 -- 0xFFFF	Length of actual image data (Note 4).
	DATA, 0 -- 65,535	Any	Actual image data.
End Image Content	ID, 1	0x93	
	LENGTH, 1	0x00	
End Segment	ID, 1	0x71	
	LENGTH, 1	0x00	

Note 1. The BITORDR attribute shall be omitted from this segment and therefore the default bit order for the image data is to be used, which is "bit order within each image data byte is from left to right.

Note 2. CCITT Group 4 compression not allowed here.

Note 3. The SIZE2 and SIZE3 attributes shall be omitted from this segment and thus default to 0.

Note 4. ABIC coded image data may not be more than 65,535 bytes in length in this standard.

Note 5. Use of ABIC decompression routines requires a license from IBM unless the provider of the routines has a license.

13.6.2 ABIC Compressed Grayscale Files

The following attributes are mandatory:

Table 13. Required Attributes for ABIC Compressed Grayscale IOCA Files

IOCA Field	Parameter, length (bytes)	Allowed Values	Notes
Begin Segment	ID, 1	0x70	
	LENGTH, 1	0x00	
Begin Image Content	ID, 1	0x91	
	LENGTH, 1	0x01	
	OBJTYPE, 1	0xFF	IOCA object
Image Size Parameter	ID, 1	0x94	
	LENGTH, 1	0x09	
	UNITBASE, 1	0x00	10 inches is the unit used for the following resolution values.
	HRESOL, 2	0x0000 -- 0x7FFF	Horizontal resolution. A typical value is 800 dots per 10 inches (describing 80 dpi), but no constraint is placed on this field.
	VRESOL, 2	0x0000 -- 0x7FFF	Vertical resolution. A typical value is 800 dots per 10 inches (describing 80 dpi), but no constraint is placed on this field.
	HSIZE, 2	0x0000 -- 0x7FFF	Horizontal size in pixels.
	VSIZE, 2	0x0000 -- 0x7FFF	Vertical size in pixels.
Image Encoding Parameter	ID, 1	0x95	(Note 1)
	LENGTH, 1	0x02	
	COMPRID, 1	0x0A	Concatenated ABIC (Note 2).
	RECID, 1	0x01	RIDIC (normal raster order).
IDE Size Parameter	ID, 1	0x96	
	LENGTH, 1	0x01	
	IDESZ, 1	0x04	4 bits per pixel.
IDE Structure Parameter	ID, 1	0x9B	
	LENGTH, 1	0x06	
	FLAGS, 1	0x40	0x40 apparently specifies use of the Gray code listed below (Note 5), although this is not documented in the Fourth Edition of the IOCA Reference (see references).
	FORMAT, 1	0x02	YCrCb color space, since this is grayscale.
	SIZE1	0x04	4 bits (Note 3).
Image Data	ID, 2	0xFE92	
	LENGTH, 2	0x0001 -- 0xFFFF	Length of actual image data (Note 4).

	DATA, 0 -- 65,535	Any	Actual image data (Note 6).
End Image Content	ID, 1	0x93	
	LENGTH, 1	0x00	
End Segment	ID, 1	0x71	
	LENGTH, 1	0x00	

Note 1. The BITORDR attribute shall be omitted from this segment and therefore the default bit order for the image data is to be used, which is “bit order within each image data byte is from left to right.”

Note 2. JPEG compression not allowed here.

Note 3: The SIZE2 and SIZE3 attributes shall be omitted from this segment and thus default to 0.

Note 4. ABIC coded image data may not be more than 65,535 bytes in length in this standard.

Note 5. The Gray code used is specified by the following table of binary values:

Input Value After Decompression and Plane Re-assembly	Recovered Original Value for Display
0000	0000
0001	0001
0010	0011
0011	0010
0100	0111
0101	0110
0110	0100
0111	0101
1000	1111
1001	1110
1010	1100
1011	1101
1100	1000
1101	1001
1101	1011
1111	1010

Note 6. This coded image data contains four concatenated separate binary images, each representing a different bit plane, with the most significant bit plane occurring first. This concatenated data stream is ABIC-coded as a single entity. This concatenated data stream is then passed in its entirety through the ABIC decompressor, then plane-reassembled into Gray-coded grayscale nibbles after decompression. Use of ABIC decompression routines requires a license from IBM unless the provider of the routines has a license.

13.7 Financial Data In Image Files

Financial or other non-image data may be placed into private or registered tags within a TIFF file. Creators of COF interchanges are cautioned, however, that such a use of a TIFF file is not standardized and that the receiver of the COF interchange will rely instead on the financial data found in the Data File or in the Images File.

The most appropriate use for such tags would then be as a cross-check against some of the data elements in the Data File to confirm that the correct image was retrieved from the Images File. The ItemSequenceNumber placed in the Images File within each Item could serve this cross-checking purpose as well.

14. Physical Media Types

The supported physical media types include:

- CD-ROM
- various tape cartridge formats

14.1 Physical Labeling Requirements

Each piece of media generated by a writer shall have a human-readable adhesive label containing at a minimum the information identified in this section. Example text is used to indicate the format.

Date: 14 June 1996
From: FRB Boston
To: Bank of Erehwon
Media Set Info: 1 of 4

NOTE TO IMPLEMENTORS: Writers of CD-ROMs are cautioned that labels which are not radially symmetric may sometimes introduce sufficient wobble to cause read errors. Some labels use adhesives which can attack the active layer of the disk. Some marking devices can similarly attack the upper (active) surface of the disk.

14.2 Media-Specific Issues: CD-ROM

CD-ROM media shall have logical formatting compliant with ISO-9660.

CD-ROM media shall have an 11-character ISO-9660 volume label which is constructed as follows:

FYYMDDSSxxx

where:

The leading F is a literal "F" character indicating a piece of media created by the Federal Reserve.

YY indicates the year, e.g. "97"

M indicates the month in hexadecimal form, as follows:

- 1 for January
- 2 for February
- 3 for March
- 4 for April
- 5 for May
- 6 or June
- 7 for July
- 8 for August
- 9 for September
- A for October
- B for November
- C for December

DD indicates the day of the month, e.g. "31"

SS indicates which Federal Reserve site generated the data

xxx indicates a number unique to a given receiving financial institution for media received by this Federal Reserve site within this day.

The two character "SS" field will be assigned by the generating Federal Reserve sites cooperatively to guarantee its uniqueness. The Media Header File and the physical label

identifies the generating institution in more detailed form for use by the receiving party.

The two character "YY" field is assigned solely to guarantee the uniqueness of the volume labels of CD-ROMs which might co-exist in a common archive. Since such archives will not be holding volumes for 100 years (more likely 7 to 15 years), a longer year designation is not necessary here. The Media Header File and the physical label identifies the date in more detailed form for use by the receiving party.

Other organizations than the Federal Reserve may use the COF format to generate data and may assign unique volume labels in other ways. Such organizations should not use a leading "F" character in their volume labels to avoid confusion.

Unique volume labels may be useful when the received CD-ROMs are placed in jukeboxes from some manufacturers.

The usage of other ISO 9660 fields is at the discretion of the writing implementation.

Note: While a date string is used here in building a volume label which is unique over a limited period, this volume label should not be relied upon to determine the date of the generated volume. Other dates exist elsewhere in this specification for that purpose.

14.3 Media-Specific Issues: Tape

14.3.1 Tape Media Types

The supported tape media types include:

- DLT™ tapes
- 3480, 3490, 3490E compatible tapes
- 8 mm helical scan tapes

14.3.2 Physical Labeling Requirements

Each piece of media generated by a writer shall have a human-readable adhesive label containing at a minimum the information identified in this section. Example text is used to indicate the format.

Date:	14 June 1996
From:	FRB Boston
To:	Bank of Erehwon
Set Info:	1 of 4

14.3.3 Media-Specific Issues: Tape

Tape media shall have physical characteristics as specified in the sections below.

Tape media shall have logical formatting compliant with ANSI X3.27-1987 and shall use "IBM labeling" as specified in the section below.

Tape drives capable of compressing data shall be used in the uncompressed mode, since already-compressed image data likely will not be further compressible by this means and may, in fact, increase in size when so treated. Some drives automatically turn off compression in these cases.

14.3.4 Tape Logical Formatting

IBM labeling is to be used in the logical formatting of the tapes, regardless of physical tape type.

This standard only permits the tape structure described in X3.27 as "Multiple File, Single Volume." This means that a tape volume does not straddle multiple individual pieces of tape media. This standard handles the need for interchanges greater in size than an individual piece of media through the alternate mechanism of Media Sets.

An implication of this requirement is that a writer of COF-compliant tapes needs to know in advance to writing that the data being written to a piece of media will not exceed the capacity of that piece of media.

Tape labels are 80-byte records written to the tape to describe and organize the files present on the tape. IBM standard labels use the EBCDIC character set and thus the present standard does as well.

According to ANSI X3.27, the fields of tape labels may use the so-called "a-characters", a set of characters including the capital letters, the numerical digits and some common printable punctuation characters. The following table (in ASCII order to facilitate translating to write in EBCDIC) lists those characters.

Table 14. Tape Label Characters.

Name	EBCDIC	ASCII
space	0x40	0x20
!	0x5A	0x21
"	0x7F	0x22
%	0x6C	0x25
&	0x50	0x26
'	0x7D	0x27
(0x4D	0x28
)	0x5D	0x29
*	0x5C	0x2A
+	0x4E	0x2B
,	0x6B	0x2C
-	0x60	0x2D
.	0x4B	0x2E
/	0x61	0x2F
0	0xF0	0x30
1	0xF1	0x31
2	0xF2	0x32
3	0xF3	0x33
4	0xF4	0x34
5	0xF5	0x35
6	0xF6	0x36
7	0xF7	0x37
8	0xF8	0x38
9	0xF9	0x39
:	0x7A	0x3A
;	0x5E	0x3B
<	0x4C	0x3C
=	0x7E	0x3D
>	0x6E	0x3E
?	0x6F	0x3F

A	letter	0xC1	0x41
B	letter	0xC2	0x42
C	letter	0xC3	0x43
D	letter	0xC4	0x44
E	letter	0xC5	0x45
F	letter	0xC6	0x46
G	letter	0xC7	0x47
H	letter	0xC8	0x48
I	letter	0xC9	0x49
J	letter	0xD1	0x4A
K	letter	0xD2	0x4B
L	letter	0xD3	0x4C
M	letter	0xD4	0x4D
N	letter	0xD5	0x4E
O	letter	0xD6	0x4F
P	letter	0xD7	0x50
Q	letter	0xD8	0x51
R	letter	0xD9	0x52
S	letter	0xE2	0x53
T	letter	0xE3	0x54
U	letter	0xE4	0x55
V	letter	0xE5	0x56
W	letter	0xE6	0x57
X	letter	0xE7	0x58
Y	letter	0xE8	0x59
Z	letter	0xE9	0x5A
_	underline	0x6D	0x5F

The required structure of a tape with its labels shall be as follows:

1. VOL1 label
2. HDR1 label
3. HDR2 label
4. tape mark
5. single concatenated file
6. tape mark
7. EOF1 label
8. EOF2 label
9. tape mark
10. tape mark

The Concatenated File is constructed by concatenating together the files of the interchange into a single file in the following order:

1. Media Header File
2. File Suite Header File

3. Images File
4. File Suite Trailer File
5. Repeat for multiple file suites as necessary
6. Media Trailer File

The Concatenated File is given a name consisting of 17 characters or fewer which is unique and is included in the File Identifier field of the HDR1 and EOF1 tape labels. A potential scheme is indicated here:

D97365.M01T01.COF

Where:

D indicates day of the year.

The year is represented by two digits, with a wraparound at 2000.

The day is represented by three digits, indicating the day number within the year (001 to 365)

M indicates media set number for the day, represented by two digits.

T indicates tape number within the media set, represented by two digits.

.COF indicates a COF-formatted interchange.

Note: While a date string is used here in building a file identifier which is unique over a limited period, this volume label should not be relied upon to determine the date of the generated volume. Other dates exist elsewhere in this specification for that purpose.

The structure of the VOL1 label is as follows.

Table 15. VOL1 Label

Field Name	Posn.	Length	Description
Label identifier	1	3	"VOL"
Label number	4	1	The number "1"
Volume serial number	5	6	This is an identification code assigned to the volume when it enters the system. COF Note: may use blanks.
Reserved	11	31	blanks
Owner name and address code	42	10	This identifies the owner of the volume. COF Note: may use blanks.
Reserved	52	29	blanks

The structure of the HDR1, EOF1 labels is as follows.

Table 16. HDR1, EOF1 Labels

Field Name	Posn.	Length	Description
Label identifier	1	3	"HDR", or "EOF"
Label number	4	1	The number "1"
File identifier	5	17	These are the rightmost 17 bytes of the file name and includes GnnnVnn if part of the generation data group. COF Note: This is the file name, in upper-case EBCDIC, left justified and padded with blanks.
File serial number	22	6	This is the volume serial number of the tape volume containing the file. COF Note: may use blanks.
Volume sequence number	28	4	This number (0001-9999) indicates the order of the volume within the multi-volume group created at the same time. COF Note: use 0001.
File sequence	32	4	This number (0001-9999) indicates the relative position of the file within a multi-file group. COF Note: use 0001.
Generation number	36	4	This field contains the number from 0000 to 9999 indicating the absolute generation number if the file is part of a generation data group (the first generation is 0000). COF Note: use blanks.
Version number	40	2	This field contains a number from 00 to 99 indicating the version number of the generation if the file is part of a generation data group (the first generation is 0000). COF Note: use blanks.
Creation date	42	6	This is the year and day the file was created and is of the form: c = century (blank=1900; 0=2000; etc.) yy = year (00-99) ddd = day (001-366) COF Note: may use zeros or may use the actual

			file creation date. If zeros are here, a COF reader should assign a date drawn from the Media Header File (HEADER.TXT) to all files extracted from the tape.
Expiration date	48	6	This is the year and day when the file may be scratched or overwritten. The data is of the same form as creation date. COF Note: use zeros.
File security	54	1	This is a code identifying the security status of the file. 0 means no security 1 means security protection on read, write, delete 2 means security protection on write or delete COF Note: use zero.
Block count	55	6	For EOF1, this is the number of data blocks in the file on the current volume (exclusive of labels and tape marks). For HDR1, this field contains zeros. COF Note: use actual block count for each file in its corresponding EOF1 label.
System code	61	13	This code identifies the system. COF Note: use upper-case EBCDIC string to identify the COF writing application vendor.
Reserved	74	3	blanks
	77	4	Block count, high order

The structure of the HDR2, EOF2 labels is as follows.

Table 17. HDR2, EOF2 Labels

Field Name	Posn.	Length	Description
Label identifier	1	3	"HDR", or "EOF"
Label number	4	1	The number "2"
Record format	5	1	This is an alphabetic character that indicates the format of the records in the associated file. F - fixed length V - variable length U - undefined length COF Note: Use V.
Block length	6	5	This is a binary number (up to 32,760) that indicates the block length in bytes. Usage depends on record format field: F - Must be an integral multiple of record length. V - Indicates maximum block length in file, including the 4-byte length field. U - Indicates maximum block length. COF Note: Since V (variable length) record format is used, this indicates the maximum block length in the file, including the 4-byte length field. Some tape drives may experience faster performance

			where this value is large and a multiple of 1024, but note the maximum length restriction above.
Record length	11	5	This is a number that indicates record length in bytes. F - Indicates actual record length. V - Indicates maximum record length in file including 4-byte length field. U - Zeros. COF Note: Record length is equal to block length value minus four.
Tape density	16	1	This is a code that indicates the recording density of the tape. 2 - 800 bpi 3 - 1600 bpi blank for 3480 COF Note: use blank.
File position	17	1	This is a code that indicates a volume switch. 0 indicates that no volume switch has occurred. 1 indicates a volume switch has occurred. COF Note: use 0.
Job/job step identification	18	17	This is an indication of the job/job step that created the file. COF Note: use blanks.
Tape recording technique	35	2	This is either a code for 7-track tapes, or blanks for 9-track tapes. COF Note: use blanks.
Printer control character	37	1	This is a code indicating whether a control character set was used to create the file, and the type of control characters used: A - ANSI control characters M - machine control characters blank - no control characters COF Note: use blank.
Reserved	38	1	blank.
Block attribute	39	1	This is a code indicating the block attribute used to create the file. The codes are: B - blocked records S - spanned records R - blocked and spanned records blank - not blocked and not spanned records COF Note: Use blocked.
Reserved	40-80	41	blanks

14.3.5 Required Files and File Ordering

The set of files required by this standard to be present on tape media is slightly different than the set of files required to be present on CD-ROM media. This is because the non-random-access nature of tape makes the use or generation of certain files awkward or impractical.

A Data File and its associated Index Files may not appear within a File Suite on a piece of tape media. The financial data associated with an item is stored within the Images File, adjacent to

the image views associated with that item, just as it is on CD-ROM media. CD-ROM media include the Data File and Index Files to assist in random access to an item within the Images File, a use not practical with tape media.

The ordering of files for tape media is otherwise identical to that for CD-ROM media.

14.3.6 Media-Specific Issues: Tape, DLT™

DLT™ tapes shall have physical characteristics and physical recording format as described in the following standards.

Among DLT™ formats, the preferred format at this time is DLT4, which permits 20 Gigabytes per tape cartridge.

14.3.6.1 DLT1

ANSI X3.242-1994 P *DLT 1 FORMAT FOR DLT260*
American National Standard for Information Systems.
Magnetic Tape Cartridge for Information Interchange
- 0.50 in (12.65 mm), Serial Serpentine, 48-track,
42,500 bpi (1673 bpmm) DLT 1 Format
ECMA 182 *DLT1 FORMAT FOR DLT260*
DATA INTERCHANGE ON 12,7 MM 48-TRACK
MAGNETIC TAPE CARTRIDGES - DLT 1 FORMAT
ISO-IEC DIS 13421
Information Technology - Data Interchange on 12,7 mm 48-track
magnetic tape cartridge - DLT1 format

14.3.6.2 DLT2

ANSI X3.266-1996 P *DLT 2 FORMAT FOR DLT600*
American National Standard for Information Systems.
Magnetic Tape Cartridge for Information Interchange
- 0.50 in (12.65 mm), Serial Serpentine, 112-track,
42,500 bpi (1673 bpmm) DLT 2 Format
ECMA 197 *DLT2 FORMAT FOR DLT600*
DATA INTERCHANGE ON 12,7 MM 112-TRACK
MAGNETIC TAPE CARTRIDGES - DLT 2 FORMAT
ISO-IEC DIS 13962
Information Technology - Data Interchange on 12,7 mm 112-track
magnetic tape cartridge - DLT2 format

14.3.6.3 DLT3

ANSI X3.282-199X *DLT 3 FORMAT FOR DLT2000*
American National Standard for Information Systems.
Magnetic Tape Cartridge for Information Interchange
- 0.50 in (12.65 mm), Serial Serpentine, 128-track,
62,500 bpi (2460 bpmm) DLT 3 Format
ECMA 209 *DLT3 FORMAT FOR DLT2000*
DATA INTERCHANGE ON 12,7 MM 128-TRACK
MAGNETIC TAPE CARTRIDGES - DLT 3 FORMAT
ISO-IEC DIS 14833
Information Technology - Data Interchange on 12,7 mm 128-track
magnetic tape cartridge - DLT3 format

14.3.6.4 DLT4

ANSI Project No. 1160 Latest 3rd draft X3B5 96-218

DLT 4 FORMAT FOR DLT2000

American National Standard for Information Systems.

Magnetic Tape Cartridge for Information Interchange

- 0.50 in (12.65 mm), Serial Serpentine, 128-track,

81 600 bpi (3213 bpmm) DLT 4 Format

ECMA 231 DLT4 FORMAT FOR DLT4000

DATA INTERCHANGE ON 12,7 MM 128-TRACK

MAGNETIC TAPE CARTRIDGES - DLT 4 FORMAT

ISO-IEC DIS 15307 In review cycle to be published shortly

Information Technology - Data Interchange on 12,7 mm 128-track

magnetic tape cartridge - DLT4 format

14.3.6.5 DLT5

ANSI Project No. 1229 Latest 1st draft X3B5 96-220

DLT 5 FORMAT FOR DLT7000

American National Standard for Information Systems.

Magnetic Tape Cartridge for Information Interchange

- 0.50 in (12.65 mm), Serial Serpentine, 128-track,

85 940 bpi (3383 bpmm) DLT 5 Format

14.3.7 Media-Specific Issues: Tape, 3480-Compatible

3480-compatible tapes shall have physical characteristics and physical recording format as described in the following standards.

14.3.7.1 3480

The following standards are related, with the first one published in 1990. They deal with both the cartridge and the recording format.

ANSI X3.180

ECMA-120

ISO/IEC 9661

14.3.7.2 3490

The following standards are related, with the first one published in 1994. They deal with the recording format. These standards use the cartridge from ANSI X3.180.

ANSI X3.224

ECMA-152

ISO/IEC 11559

14.3.7.3 3490E

The following standards are related, with the first one published in 1996. They deal with the recording format.

ANSI X3.261

ECMA-196

ISO/IEC 14251

The following standards are related, with the first one published in 1996. They deal with the cartridge.

ANSI X3.265

ECMA-196

ISO/IEC 14251

14.3.8 Media-Specific Issues: Tape, 8mm Helical-Scan

8mm helical-scan tapes shall have physical characteristics and physical recording format as described in:

ANSI Helical-Scan Digital Computer Tape Cartridge, X3B5/89-136, Revision 6,
and
EXB-8500 Cartridge Tape Subsystem Product Specification. Document Number 510200-003 from Exabyte Corporation, Boulder, CO. 303-442-4333.

15. References

1. *TIFF Specification*, Revision 6.0, Final, June 3, 1992. Adobe Corporation.
2. TIFF Technical Note Number 2, draft.
3. Image Object Content Architecture (IOCA) Reference. 4th edition, August 1993, IBM Corporation, Armonk, NY. Document number: SC31-6805-03.
4. ISO 10918-1 / ITU-T Recommendation T.81, *Digital Compression and Coding of Continuous-Tone Still Images* (JPEG Standard).
5. Pennebaker, W.B. and Mitchell, J.M., *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.
6. ITU-T Recommendation T.6.
7. Arps, R.B., et al, *A multi-purpose VLSI chip for adaptive data compression of bilevel images*, IBM Journal of Research and Development, vol. 32, no. 6, November, 1988, pp 775-795. (ABIC Specification).
8. ANSI X9.46. Financial Image Interchange Standard (FIIS).
9. ANSI X9 TG-15, draft. Technical Guideline on FIIS.
10. ANSI X9.37-1994. Specifications for Electronic Check Exchange.
11. ISO 9660 CD-ROM Standard.
12. ANSI X3.27 - 1987, Information Systems -- File structure and labeling of magnetic tapes for information interchange.
13. MVS/DFP Version 3 Release 2, Using Magnetic Tape Labels and File Structure. Document number: SC26-4565-0.
14. DFSMS/MVS Version 1 Release 2, Using Magnetic Tapes. Document number: SC26-4923-01.
15. *ANSI Helical-Scan Digital Computer Tape Cartridge, X3B5/89-136, Revision 6.*
16. *EXB-8500 Cartridge Tape Subsystem Product Specification.* Document number: 510200-003. Exabyte Corporation, Boulder, CO. 303-442-4333.
17. *The File Formats Handbook* by Günter Born, 1995, International Thomson Computer Press, London, ISBN 1-85032-117-5.
18. *File Formats for Popular PC Software. A Programmer's Reference* by Jeff Walden, 1986, John Wiley & Sons, Inc, New York.
19. ANSI/IEEE 754, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, New York, NY.

16. Revision History

COF Version 1.1	
10 January 1997	
Change Number	Description
	Initial release.

COF Version 1.2 Draft	
18 April 1997	
Change Number	Description
	Draft revision to tape portion to use a single concatenated file.

COF Version 1.2 Final	
30 April 1997	
Change Number	Description
	Final revision to tape portion to use a single concatenated file.
1.	Language changes pg. 1, Section 1, Summary.
2.	Section 14.3.4, Tape Logical Formatting. Tightened language specifying formation of the File Identifier for the Concatenated File.
3.	Section 10, deleted multiple references to NumberOfFileSuites
4.	Section 14.3.4, Table 17, corrected Record Length note.

COF Version 1.3	
18 March 1998	
Change Number	Description
	Miscellaneous minor changes.
1.	Section 9, Media Header File Structure. Clarified possible future values of COFVersion.
2.	Throughout. Cleaned up section references.
3.	Section 13.4 added <cr><lf>, two places.
4.	Stopped using [and] for comments.
5.	Added requirement that required CD-ROM files be in root directory.
6.	Section 10. Added [FileInfo] section to Media Trailer.
7.	Section 8.4 Added Media Sources to Media Header File. Added Section 8.4.1.
8.	Changed Rich Puttin's phone and email for new building.
9.	Clarified Note 12 and Section 13 for case of no views for an item.
10.	Repaired line breaks in Annex A.
11.	Section 13.6.2 Note 5 following Table. Correction to Gray code table.
12.	Section 14.3.4 Clarified that indicated file identifier scheme is not required.
13.	Sections 11.1.2, 14.2, 14.3.4 Addressed year 2000 date issues.

17. Annex A (normative). DBF File Format

The following document was retrieved from the Borland International web site technical documents section. To retrieve it, perform a search by the document number from the location:

<http://www.borland.com/techsupport/VdBASE/search.html>

Document Number: TI2821

Title: dBASE .DBF File Structure

Note: This standard only permits use of the format described below as the dBASE III PLUS format, the original format, which appears first in the file. The later generations of the format are also included here to assist in identifying problems in a COF file which appears to incorrectly generate a later version of the DBF format.

In this file, a trailing "h" is used to indicate a hexadecimal value.

Sometimes it is necessary to delve into a dBASE table outside the control of the Borland Database Engine (BDE). For instance, if the .DBT file (that contains memo data) for a given table is irretrievably lost, the file will not be usable because the byte in the file header indicates that there should be a corresponding memo file. This necessitates toggling this byte to indicate no such accompanying memo file. Or, you may just want to write your own data access routine.

Below are the file structures for dBASE table files. Represented are the file structures as used for various versions of dBASE: dBASE III PLUS 1.1, dBASE IV 2.0, dBASE 5.0 for DOS, and dBASE 5.0 for Windows.

The data file header structure for dBASE III PLUS table file.

The table file header:

=====

Byte	Contents	Description
0	1 byte	Valid dBASE III PLUS table file (03h without a memo (.DBT file; 83h with a memo).
1-3	3 bytes	Date of last update; in YYMMDD format.
4-7	32-bit number	Number of records in the table.
8-9	16-bit number	Number of bytes in the header.
10-11	16-bit number	Number of bytes in the record.
12-14	3 bytes	Reserved bytes.
15-27	13 bytes	Reserved for dBASE III PLUS on a LAN.
28-31	4 bytes	Reserved bytes.
32-n	32 bytes each	Field descriptor array (the structure of this array is shown below)
n+1	1 byte	0Dh stored as the field terminator.

n above is the last byte in the field descriptor array. The size of the array depends on the number of fields in the table file.

Table Field Descriptor Bytes

=====

Byte	Contents	Description
0-10	11 bytes	Field name in ASCII (zero-filled).
11	1 byte	Field type in ASCII (C, D, L, M, or N).
12-15	4 bytes	Field data address (address is set in memory; not useful on disk).
16	1 byte	Field length in binary.
17	1 byte	Field decimal count in binary.
18-19	2 bytes	Reserved for dBASE III PLUS on a LAN.
20	1 byte	Work area ID.
21-22	2 bytes	Reserved for dBASE III PLUS on a LAN.
23	1 byte	SET FIELDS flag.
24-31	1 byte	Reserved bytes.

**Table Records
=====**

The records follow the header in the table file. Data records are preceded by one byte, that is, a space (20h) if the record is not deleted, an asterisk (2Ah) if the record is deleted. Fields are packed into records without field separators or record terminators. The end of the file is marked by a single byte, with the end-of-file marker, an OEM code page character value of 26 (1Ah). You can input OEM code page data as indicated below.

**Allowable Input for dBASE Data Types
=====**

Data Type	Data Input
C (Character)	All OEM code page characters.
D (Date)	Numbers and a character to separate month, day, and year (stored internally as 8 digits in YYYYMMDD format).
N (Numeric)	- . 0 1 2 3 4 5 6 7 8 9
L (Logical)	? Y y N n T t F f (? when not initialized).
M (Memo)	All OEM code page characters (stored internally as 10 digits representing a .DBT block number).

**Binary, Memo, and OLE Fields And .DBT Files
=====**

Memo fields store data in .DBT files consisting of blocks numbered sequentially (0, 1, 2, and so on). The size of these blocks are internally set to 512 bytes. The first block in the .DBT file, block 0, is the .DBT file header.

Memo field of each record in the .DBF file contains the number of the block (in OEM code page values) where the field's data actually begins. If a field contains no data, the .DBF file contains blanks (20h) rather than a number.

When data is changed in a field, the block numbers may also change and the number in the .DBF may be changed to reflect the new location.

This information is from the Using dBASE III PLUS manual, Appendix C.

The data file header structure for dBASE IV 2.0 table file.

File Structure:
=====

Byte	Contents	Meaning
0	1byte	Valid dBASE IV file; bits 0-2 indicate version number, bit 3 the presence of a dBASE IV memo file, bits 4-6 the presence of an SQL table, bit 7 the presence of any memo file (either dBASE III PLUS or dBASE IV).
1-3	3 bytes	Date of last update; formatted as YYMMDD.
4-7	32-bit number	Number of records in the file.
8-9	16-bit number	Number of bytes in the header.
10-11	16-bit number	Number of bytes in the record.
12-13	2 bytes	Reserved; fill with 0.
14	1 byte	Flag indicating incomplete transaction.
15	1 byte	Encryption flag.
16-27	12 bytes	Reserved for dBASE IV in a multi-user environment.
28	1 bytes	Production MDX file flag; 01H if there is an MDX, 00H if not.
29	1 byte	Language driver ID.
30-31	2 bytes	Reserved; fill with 0.
32-n*	32 bytes each	Field descriptor array (see below).
n + 1	1 byte	0DH as the field terminator.

* n is the last byte in the field descriptor array. The size of the array depends on the number of fields in the database file.

The field descriptor array:
=====

Byte	Contents	Meaning
0-10	11 bytes	Field name in ASCII (zero-filled).
11	1 byte	Field type in ASCII (C, D, F, L, M, or N).
12-15	4 bytes	Reserved.
16	1 byte	Field length in binary.
17	1 byte	Field decimal count in binary.
18-19	2 bytes	Reserved.
20	1 byte	Work area ID.
21-30	10 bytes	Reserved.
31	1 byte	Production MDX field flag; 01H if field has an index tag in the production MDX file, 00H if not.

Database records:
=====

The records follow the header in the database file. Data records are preceded by one byte; that is, a space (20H) if the record is not deleted, an asterisk (2AH) if the record is deleted. Fields are packed into records without field separators or record terminators. The end of the file is marked by a single byte, with the end-of-file marker an ASCII 26 (1AH) character.

Allowable Input for dBASE Data Types:
=====

Data	Type	Data Input
C	(Character)	All OEM code page characters.
D	(Date)	Numbers and a character to separate month, day, and year (stored internally as 8 digits in YYYYMMDD format).
F	(Floating point binary)	- . 0 1 2 3 4 5 6 7 8 9

N	numeric) (Binary coded decimal numeric)	- . 0 1 2 3 4 5 6 7 8 9
L	(Logical)	? Y y N n T t F f (? when not initialized).
M	(Memo)	All OEM code page characters (stored internally as 10 digits representing a .DBT block number).

Memo Fields And .DBT Files

=====

Memo fields store data in .DBT files consisting of blocks numbered sequentially (0, 1, 2, and so on). SET BLOCKSIZE determines the size of each block. The first block in the .DBT file, block 0, is the .DBT file header.

Each memo field of each record in the .DBF file contains the number of the block (in OEM code page values) where the field's data actually begins. If a field contains no data, the .DBF file contains blanks (20h) rather than a number.

When data is changed in a field, the block numbers may also change and the number in the .DBF may be changed to reflect the new location.

This information is from the dBASE IV Language Reference manual, Appendix D.

The data file header structure for dBASE 5.0 for DOS table file.

The table file header:
=====

Byte	Contents	Description
0	1 byte	Valid dBASE for Windows table file; bits 0-2 indicate version number; bit 3 indicates presence of a dBASE IV or dBASE for Windows memo file; bits 4-6 indicate the presence of a dBASE IV SQL table; bit 7 indicates the presence of any .DBT memo file (either a dBASE III PLUS type or a dBASE IV or dBASE for Windows memo file).
1-3	3 bytes	Date of last update; in YYMMDD format.
4-7	32-bit number	Number of records in the table.
8-9	16-bit number	Number of bytes in the header.
10-11	16-bit number	Number of bytes in the record.
12-13	2 bytes	Reserved; filled with zeros.
14	1 byte	Flag indicating incomplete dBASE transaction.
15	1 byte	Encryption flag.
16-27	12 bytes	Reserved for multi-user processing.
28	1 byte	Production MDX flag; 01h stored in this byte if a production .MDX file exists for this table; 00h if no .MDX file exists.
29	1 byte	Language driver ID.
30-31	2 bytes	Reserved; filled with zeros.
32-n	32 bytes each	Field descriptor array (the structure of this array is shown below)
n+1	1 byte	0Dh stored as the field terminator.

n above is the last byte in the field descriptor array. The size of the array depends on the number of fields in the table file.

Table Field Descriptor Bytes
=====

Byte	Contents	Description
0-10	11 bytes	Field name in ASCII (zero-filled).
11	1 byte	Field type in ASCII (B, C, D, F, G, L, M, or N).
12-15	4 bytes	Reserved.
16	1 byte	Field length in binary.
17	1 byte	Field decimal count in binary.
18-19	2 bytes	Reserved.
20	1 byte	Work area ID.
21-30	10 bytes	Reserved.
31	1 byte	Production .MDX field flag; 01h if field has an index tag in the production .MDX file; 00h if the field is not indexed.

Table Records
=====

The records follow the header in the table file. Data records are preceded by one byte, that is, a space (20h) if the record is not deleted, an asterisk (2Ah) if the record is deleted. Fields are packed into records without field separators or record terminators. The end of the file is marked by a single byte, with the end-of-file marker, an OEM code page character value of 26 (1Ah). You can input OEM code page data as indicated below.

Allowable Input for dBASE Data Types
=====

Data Type	Data Input
C (Character)	All OEM code page characters.
D (Date)	Numbers and a character to separate month, day, and year (stored internally as 8 digits in YYYYMMDD format).
F (Floating point binary numeric)	- . 0 1 2 3 4 5 6 7 8 9
N (Numeric)	- . 0 1 2 3 4 5 6 7 8 9
L (Logical)	? Y y N n T t F f (? when not initialized).
M (Memo)	All OEM code page characters (stored internally as 10 digits representing a .DBT block number).

Memo Fields And .DBT Files
=====

Memo fields store data in .DBT files consisting of blocks numbered sequentially (0, 1, 2, and so on). SET BLOCKSIZE determines the size of each block. The first block in the .DBT file, block 0, is the .DBT file header.

Each memo field of each record in the .DBF file contains the number of the block (in OEM code page values) where the field's data actually begins. If a field contains no data, the .DBF file contains blanks (20h) rather than a number.

When data is changed in a field, the block numbers may also change and the number in the .DBF may be changed to reflect the new location.

Unlike dBASE III PLUS, if you delete text in a memo field, dBASE 5.0 for DOS may reuse the space from the deleted text when you input new text. dBASE III PLUS always appends new text to the end of the .DBT file. In dBASE III PLUS, the .DBT file size grows whenever new text is added, even if other text in the file is deleted.

This information is from the dBASE for DOS Language Reference manual, Appendix C.

The data file header structure for dBASE 5.0 for Windows table file.

The table file header:
=====

Byte	Contents	Description
0	1 byte	Valid dBASE for Windows table file; bits 0-2 indicate version number; bit 3 indicates presence of a dBASE IV or dBASE for Windows memo file; bits 4-6 indicate the presence of a dBASE IV SQL table; bit 7 indicates the presence of any .DBT memo file (either a dBASE III PLUS type or a dBASE IV or dBASE for Windows memo file).
1-3	3 bytes	Date of last update; in YYMMDD format.
4-7	32-bit number	Number of records in the table.
8-9	16-bit number	Number of bytes in the header.
10-11	16-bit number	Number of bytes in the record.
12-13	2 bytes	Reserved; filled with zeros.
14	1 byte	Flag indicating incomplete dBASE IV transaction.
15	1 byte	dBASE IV encryption flag.
16-27	12 bytes	Reserved for multi-user processing.
28	1 byte	Production MDX flag; 01h stored in this byte if a production .MDX file exists for this table; 00h if no .MDX file exists.
29	1 byte	Language driver ID.
30-31	2 bytes	Reserved; filled with zeros.
32-n	32 bytes each	Field descriptor array (the structure of this array is shown below)
n+1	1 byte	0Dh stored as the field terminator.

n above is the last byte in the field descriptor array. The size of the array depends on the number of fields in the table file.

Table Field Descriptor Bytes
=====

Byte	Contents	Description
0-10	11 bytes	Field name in ASCII (zero-filled).
11	1 byte	Field type in ASCII (B, C, D, F, G, L, M, or N).
12-15	4 bytes	Reserved.
16	1 byte	Field length in binary.
17	1 byte	Field decimal count in binary.
18-19	2 bytes	Reserved.
20	1 byte	Work area ID.
21-30	10 bytes	Reserved.
31	1 byte	Production .MDX field flag; 01h if field has an index tag in the production .MDX file; 00h if the field is not indexed.

Table Records
=====

The records follow the header in the table file. Data records are preceded by one byte, that is, a space (20h) if the record is not deleted, an asterisk (2Ah) if the record is deleted. Fields are packed into records without field separators or record terminators. The end of the file is marked by a single byte, with the end-of-file marker, an OEM code page character value of 26 (1Ah). You can input OEM code page data as indicated below.

Allowable Input for dBASE Data Types
=====

Data Type	Data Input
B (Binary)	All OEM code page characters (stored internally as 10 digits representing a .DBT block number).
C (Character)	All OEM code page characters.
D (Date)	Numbers and a character to separate month, day, and year (stored internally as 8 digits in YYYYMMDD format).
G (General)	All OEM code page characters (stored internally as 10 digits or OLE) representing a .DBT block number).
N (Numeric)	- . 0 1 2 3 4 5 6 7 8 9
L (Logical)	? Y y N n T t F f (? when not initialized).
M (Memo)	All OEM code page characters (stored internally as 10 digits representing a .DBT block number).

Binary, Memo, and OLE Fields And .DBT Files
=====

Binary, memo, and OLE fields store data in .DBT files consisting of blocks numbered sequentially (0, 1, 2, and so on). SET BLOCKSIZE determines the size of each block. The first block in the .DBT file, block 0, is the .DBT file header.

Each binary, memo, or OLE field of each record in the .DBF file contains the number of the block (in OEM code page values) where the field's data actually begins. If a field contains no data, the .DBF file contains blanks (20h) rather than a number.

When data is changed in a field, the block numbers may also change and the number in the .DBF may be changed to reflect the new location.

Unlike dBASE III PLUS, if you delete text in a memo field (or binary and OLE fields), dBASE for Windows (unlike dBASE IV) may reuse the space from the deleted text when you input new text. dBASE III PLUS always appends new text to the end of the .DBT file. In dBASE III PLUS, the .DBT file size grows whenever new text is added, even if other text in the file is deleted.

This information is from the dBASE for Windows Language Reference manual, Appendix C.