Volume 4

OGC CDB Rules for Encoding CDB Vector Data using Shapefiles (Best Practice)

Open Geospatial Consortium

Submission Date: 2020-01-21

Approval Date: 2020-08-24

Publication Date: 2020-xx-xx

External identifier of this OGC® document: http://www.opengis.net/doc/BP/shapefile-guidance/1.2

Internal reference number of this OGC® document: 16-070r4

Version: 1.2

Category: OGC® Best Practice

Editor: Carl Reed

Volume 4: OGC CDB Rules for Encoding CDB Vector Data using Shapefiles (Best Practice)

Copyright notice

Copyright © 2020 Open Geospatial Consortium

To obtain additional rights of use, visit http://www.opengeospatial.org/legal/

Warning

This document defines an OGC Best Practices on a particular technology or approach related to an OGC standard. This document is **not** an OGC Standard and may not be referred to as an OGC Standard. It is subject to change without notice. However, this document is an **official** position of the OGC membership on this particular technology topic.

Document type:	OGC® Best Practice
Document subtype:	
Document stage:	Approved
Document language:	English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of Contents

1. Scope . . . 5
2. Conformance . . . 6
3. References . . . 7
4. Terms and Definitions . . . 8
5. Conventions . . . 9
5.1. Identifiers . . . 9
6. General Guidance on the use of Shapefiles . . . 10
Annex A: Conformance Class Abstract Test Suite (Normative) . . . 15
A.1. Conformance Test Class: OGC CDB Shapefiles for vector data storage . . . 15
A.1.1. General Shapefile Implementation Feature Rule . . . 15
A.1.2. Shapefile Point Vertices . . . 15
Annex B: Revision History . . . 17
Annex C: Shapefile dBASE III guidance . . . 18

i. Abstract

This CDB volume provides the information and guidance required to store vector data and attributes using the Esri Shapefile specification in a CDB data store. All shape types are supported to represent point, line, and polygon features.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, shapefile, cdb, vector, point, line, polygon

iii. Preface

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

Organization name(s)

- CAE Inc.
- Carl Reed, OGC Individual Member
- Envitia, Ltd
- Glen Johnson, OGC Individual Member
- KaDSci, LLC
- Laval University
- Open Site Plan
- University of Calgary
- UK Met Office

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name	Affiliation
Carl Reed	Carl Reed & Associates
David Graham	CAE Inc.

Chapter 1. Scope

This CDB Best Practice volume defines the requirements and provides guidance on how to use Esri ShapeFiles in a CDB data store.

For ease of editing and review, the standard has been separated into 16 Volumes, one being a schema repository.

- Volume 0: OGC CDB Companion Primer for the CDB standard (Best Practice).
- Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure. The main body (core) of the CDB standard (Normative).
- Volume 2: OGC CDB Core Model and Physical Structure Annexes (Best Practice).
- Volume 3: OGC CDB Terms and Definitions (Normative).
- Volume 4: OGC CDB Rules for Encoding CDB Vector Data using Shapefiles (Best Practice).
- Volume 5: OGC CDB Radar Cross Section (RCS) Models (Best Practice).
- Volume 6: OGC CDB Rules for Encoding CDB Models using OpenFlight (Best Practice).
- Volume 7: OGC CDB Data Model Guidance (Best Practice).
- Volume 8: OGC CDB Spatial Reference System Guidance (Best Practice).
- Volume 9: OGC CDB Schema Package: http://schemas.opengis.net/cdb/ provides the normative schemas for key features types required in the synthetic modeling environment. Essentially, these schemas are designed to enable semantic interoperability within the simulation context (Normative).
- Volume 10: OGC CDB Implementation Guidance (Best Practice).
- Volume 11: OGC CDB Core Standard Conceptual Model (Normative).
- Volume 12: OGC CDB Navaids Attribution and Navaids Attribution Enumeration Values (Best Practice).
- Volume 13: OGC CDB Rules for Encoding CDB Vector Data using GeoPackage (Normative, Optional Extension).
- Volume 14: OGC CDB Guidance on Conversion of CDB Shapefiles into CDB GeoPackages (Best Practice).
- Volume 15: OGC CDB Optional Multi-Spectral Imagery Extension (Normative).

\\\\ For later https://github.com/opengeospatial/cdb-volume-1/blob/master/list_of_volumes.adoc \\\\

Chapter 2. Conformance

This standard defines conformance class for testing the use of Esri Shapefiles for storing vector data in a CDB data store.

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site[1].

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

[1] www.opengeospatial.org/cite

Chapter 3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

Esri ShapeFile Technical Description (*https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf*)

Chapter 4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

Please see the CDB Volume 3: Terms and Definitions document. http://www.opengeospatial.org/standards/cdb

Chapter 5. Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this standard are denoted by the URI namespace

```
http://www.opengis.net/spec/cdb/1.0/shapefile
```

All requirements that appear in this document are denoted by partial URIs which are relative to the namespace shown above.

For the sake of brevity, the use of “req” in a requirement URI denotes: http://www.opengis.net/spec/cdb/1.0/shapefile/req

An example might be:

req/shapefile/storage

All conformance tests that appear in this document are denoted by partial URIs which are relative to the namespace shown above.

For the sake of brevity, the use of “conf” in a requirement URI denotes: http://www.opengis.net/spec/cdb/1.0/shapefile/conf

Chapter 6. General Guidance on the use of Shapefiles

A Shapefile is a specification for encoding and storing vector data storage format for storing the location, shape, and attributes of geographic features. A Shapefile is stored as a set of related files and contains one feature class. The Shapefile specification was developed by Esri.

In a CDB Shapefile, the position of all points is expressed using WGS-84 geographic coordinates (latitude, longitude, altitude), as explained in Volume 8: CDB Spatial Reference Systems Guidance.

As per Esri Shapefile Technical Description, the set of attributes of Vector features are stored in dBase III+ files. Refer to Annex E in Volume 2[2] of the CDB standard: CDB Annexes for the dBase III file format description and recommended encodings. The CDB standard provides three attribution schemas to represent attribution data:

- Instance-level attribution schema
- Class-level attribution schema
- Extended-level attribution schema

Requirements Class Shapefiles (1-2)	
/req/shapefile/general	
Target type	Operations
Dependency	Shapefile specification
Requirement 1	/req/shapefile/vector-shp-rule
Requirement 2	/req/shapefile/shapefile-reader

To completely represent the vector data and attributes in a given tile as a Shapefile, the CDB standard requires that a Vector dataset consists of some or all of the following files:

- ***.shp** – feature shape files that provides the geometric aspects of each instance of a vector feature (point, lineal, and polygon features).

Requirement 1 Vector Shapefile Shape Type	http://www.opengis.net/spec/cdb/1.1/shapefile/vector-shp-rule All instances of the feature *SHALL* be of the same Shape type. While the Shapefile format supports up to 13 different types (each one stored in a different shape file), the CDB standard requires a maximum of one Shapefile type for point features, a maximum of one Shapefile type for lineal features and a maximum of one Shapefile type for polygon features for each tile (for a maximum of 3 feature Shapefiles per tile).

- ***.shx** – feature index files that stores the file offsets and content lengths for each of the records

of the feature files. The only purpose of these files is to provide a simple means for clients to step through the individual records of the feature files (i.e., it contains no CDB modeled data).

- ***.dbf** – feature instance-level files that provide the instance-level attribution data for each of the records of the feature.
- ***.dbf** – feature class-level files that provide the class-level attribution data for each class of features present in the feature shape files.
- ***.dbf** – feature extended-level files that provide optional extended-level attribution data for entries in either the feature instance- or class-level files.
- ***.shp** – figure point shape files allow modelers the ability to assign specific attribution for each point in lineal or polygon features. Without this additional Shapefile, the Shapefile format only allows specifying a single attribution for the entire lineal or polygon feature. The CDB standard extends the concept to allow specific attribution to each point of these features while enforcing position correlation. For instance, in case of a PowerLine feature, it is possible to associate, within the same dataset, a different geometric representation of a PowerLine pylon for each point of the lineal and still maintain the relationship between the point and the lineal.
- ***.shx** – figure point index files that stores the file offsets and content lengths for each of the records of the figure point shape files.
- ***.dbf** – figure point instance-level files that provide the instance-level attribution data for each of the records of the figure point shape files.
- ***.dbf** – figure point class-level files that provide the class-level attribution data for each class of features present in the figure point shape files
- ***.dbf** – figure point extended-level files that provide optional extended-level attribution data for entries in either the figure point instance- or class-level files.
- ***.dbf** – 2D relationship files. These files establish the relationship of point, lineal, and polygon features of a single or different datasets in a tile and between tiles.

In addition to *.shp, *.dbf and *.shx files, the Shapefile specification also refers to a memo file with a *.dbt file that is used to store comment fields associated with the attribution *.dbf file.

All of the information that is needed to instance features is organized in accordance to the CDB tile structure. All the tiled Shapefile dataset files are located in the same directory. The dataset's second component selector (CS2) is used to differentiate between files with the same extension or with the same Vector features. Table 6-1: Component Selector 2 for Vector Dataset, presents the list of codes that are allocated. Note that Vector datasets do not necessarily use all of the Dataset Component Selector 2 reserved codes. Users of the CDB standard should refer to the appropriate section for an enumeration of the supported File Component Selector 2 codes as well as the ones specific to the Dataset.

The Vector dataset concept and the feature code concepts overlap somewhat; some of the Vector datasets are generalizations or specializations of feature codes. Section 1.5 of the OGC CDB Core Standard: Model and Physical Data Store Structure (Volume 1) provides a recommended mapping of the feature attributes across the CDB compliant datasets. Note that the same feature *should* not have two representations.

Table 6-1: Component Selector 2 for Vector Datasets

CS2	File Extension	Dataset Component Name	Supported Shape Type
001	*.shp *.shx *.dbf	Point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
002	*.dbf	Point feature class-level attributes	N/A
003	*.shp *.shx *.dbf	Lineal features	PolyLine, PolyLineZ, PolyLineM
004	*.dbf	Lineal feature class-level attributes	N/A
005	*.shp *.shx *.dbf	Polygon features	Polygon, PolygonZ, PolygonM, `MultiPatch`
006	*.dbf	Polygon feature class-level attributes	N/A
007	*.shp *.shx *.dbf	Lineal figure point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
008	*.dbf	Lineal figure point feature class-level attributes	N/A
009	*.shp *.shx *.dbf	Polygon figure point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
010	*.dbf	Polygon figure point feature class-level attributes	N/A
011	*.dbf	2D relationship tile connections	N/A
012		Deprecated	N/A
013		Deprecated	N/A
014		Deprecated	N/A
015	*.dbf	2D relationship dataset connections	N/A
016	*.dbf	Point feature extended-level attributes	N/A

017	*.dbf	Lineal feature extended-level attributes	N/A
018	*.dbf	Polygon feature extended-level attributes	N/A
019	*.dbf	Lineal Figure Point extended-level attributes	N/A
020	*.dbf	Polygon Figure Point extended-level attributes	N/A

Deprecation Note: In CDB Version 1.2, the Shapefile *MultiPatch* geometry type was deprecated. For backwards compatibility, this geometry type will remain in the CDB standard until such time as CDB Version 2.0 is approved as an OGC standard. AT that time, all MultiPatch references will be removed from all CDB volumes.

Notes about Shapefile Polygon Shapes

Even though the Shapefile standard is very versatile, it also enforces some guidelines with respect to the Polygon Shapes. Those guidelines are referred to in Volume 4: OGC CDB Use of Shapefiles for Vector Data Storage (Best Practice).

Requirement 2 Shapefile polygon readers	http://www.opengis.net/spec/cdb/1.1/shapefile/polygon-rules-reader Although the above are guidelines, Shapefile readers *SHALL* handle the following cases with proper error handling and reporting for Polygon shapes: * Has no self-intersections or co-linear segments * Has no identical consecutive points (no zero-length segments) * Does not degenerate into zero-area parts * Does not have clock-wise inner rings (“Dirty Polygon”)

[2] Volume 2: OGC CDB Core Model and Physical Structure Annexes (Best Practice)

Annex A: Conformance Class Abstract Test Suite (Normative)

A.1. Conformance Test Class: OGC CDB Shapefiles for vector data storage

This section describes conformance test for the OGC CDB Standard Core. These abstract test cases describe the conformance criteria for verifying the structure and content of any data store claiming conformance to the CDB standard.

The conformance class id is “http://www.opengis.net/spec/http://opengis.net/spec/CDB/1.0/core/lod-hierarchy[cdb-shapefile/1.0]/conf/” and all of the other conformance tests URLs are created in this path. Another issue that the reader should pay attention to is the test method. When the test method is assigned with “Visual”, it means that the purpose of the test should be “visually” investigate the file contents, image, or other content.

A.1.1. General Shapefile Implementation Feature Rule

The following conformance test is designed is to determine if a CDB vector Shapefile adheres to the feature type instance rule.

Test identifier	/conf/shapefile/vector-shape-rule
Test purpose:	Verify that all instances of the feature are of the same Shape type.
Test method:	Visual inspection. Pass if verified.
Requirement:	/req/shapefile/vector-shp-rule
Dependency:	Shapefile specification
Test type:	Conformance

A.1.2. Shapefile Point Vertices

Ensure that Shapefile readers handle the following cases with proper error handling and reporting for Polygon shapes:

- Has no self-intersections or co-linear segments
- Has no identical consecutive points (no zero-length segments)
- Does not degenerate into zero-area parts
- Does not have clock-wise inner rings (“Dirty Polygon”)

Test identifier	/conf/shapefile/polygon-rules-reader
Test purpose:	Verify that a Shapefile reader handles polygon data correctly and that any errors in the polygon data as per requirement 2 are properly handled and reported.

Test method:	Visual inspection. Pass if verified.
Requirement:	/req/shapefile/polygon-rules-reader
Dependency:	Shapefile specification
Test type:	Conformance

Annex B: Revision History

Date	Release	Author	Paragraph modified	Description
2/6/2016	Draft	C Reed	Many	Ready for OAB review
3/12/2016	Draft	C Reed	Many	Add Shapefile normative text from core into this document and redo requirements and ATS
3/18/2015	Draft	C Reed	Various	Remove RCS and put in separate volume.
11/15/16	Final	C Reed	Various	Final edits for publication
12/28/17	Draft	C Reed	Various	Draft edits for version 1.1.
12/22/19	1.2	C Reed	Scope, Cover page	Minor updates for version 1.2

Annex C: Shapefile dBASE III guidance

Was B.1.3 Annex B Volume 2 of the OGC CDB Best Practice

dBASE .DBF File Structure

by Borland Developer Support Staff

Technical Information Database

This document has been annotated to reflect the conventions established by the CDB standard. Collectively, these conventions are referred to as dBASE/CDB. The conventions define how dBASE files are interpreted by a CDB-compliant dBASE reader; the stated conventions supersede or replace related aspects of this annotated specification. Unless stated otherwise, CDB-compliant dBASE readers will ignore any data that fails to conform to the stated conventions.

Note on directory and file names: Shape/CDB Readers: The CDB standard globally provides a set of directory and filename conventions. The conventions do not limit filenames to the 8.3 naming convention

TI838D.txt dBASE .DBF File Structure

Category :Database Programming

Platform :All

Product :Delphi All

Description:

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).

Byte	Contents	Description
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	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 mark 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 .DBTfile 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	1 Byte	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).
1-2	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 Byte	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

Byte	Contents	Meaning
17	1	Field decimal count in binary
18-19	2	Reserved
20	1	Work area ID
21-30	10	Reserved
31	1	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.

[underline]#Allowable Input for dBASE Data Types:#ß

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	. 0 1 2 3 4 5 6 7 8 9 point binary numeric
N	Binary	.0 1 2 3 4 5 6 7 8 9 coded decimal numeric
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

----- -------- ----------

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.

Byte	Contents	Description
30-31	2 bytes	Reserved; filled with zeros.
32-n	32 bytes	Field descriptor array (the structure of this array is each 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 bytes	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.

Data Type	Data Input
D (Date)	Numbers and a character to separate month, day, and year (stored internally as 8 digits in YYYYMMDD format)
F (Floating	- . 0 1 2 3 4 5 6 7 8 9 point binary numeric)
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	i. 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	Field descriptor array (the structure of this array is each 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).
1	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

1.

Table Records (See above)

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.