

```
--- ODA-5.4.1.md          2024-09-28 19:30:32.454895413 +0200
+++ ODA-5.4.2-libredwg.md  2025-04-02 12:55:26.192804546 +0200
@@ -8,20 +8,20 @@
```

2 BIT CODES AND DATA DEFINITIONS

NOTE: Unless otherwise stated, all data in this manual is in little-endian order, with the least significant byte first.

-Much of the data in the DWG file format versions 13/14/2000/2004/2007/2010 must be read at the bit level. Various parts of the drawing use data in compressed forms, which are explained below. Here are the abbreviations used in this document for the various compressed forms:

+Much of the data in the DWG file format versions 13/14/2000/2004/2007/2010/2013/2018 must be read at the bit level. Various parts of the drawing use data in compressed forms, which are explained below. Here are the abbreviations used in this document for the various compressed forms:

```
-----
      B : bit (1 or 0)
      BB : special 2 bit code (entmode in entities, for instance)
-      3B : bit triplet (1-3 bits) (R24)
+      3B : bit triplet (1-3 bits) (R2010)
      BS : bitshort (16 bits)
      BL : bitlong (32 bits)
-      BLL : bitlonglong (64 bits) (R24)
+      BLL : bitlonglong (64 bits) (R2010)
      BD : bitdouble
      2BD : 2D point (2 bitdoubles)
      3BD : 3D point (3 bitdoubles)
      RC : raw char (not compressed)
      RS : raw short (not compressed)
@@ -29,16 +29,16 @@
      RL : raw long (not compressed)
      2RD : 2 raw doubles
      3RD : 3 raw doubles
      MC : modular char
      MS : modular short
-      H : handle reference (see the HANDLE REFERENCES section)
+      H : handle reference (see the [HANDLE REFERENCES] (#213-handle-references) section)
      T : text (bitshort length, followed by the string).
      TU : Unicode text (bitshort character length, followed by Unicode string, 2 bytes
per
      character). Unicode text is read from the â\200\234string streamâ\200\235 within the object data,
      see the main Object description section for details.
-      TV : Variable text, T for 2004 and earlier files, TU for 2007+ files.
+      TV : Variable text, T for R2004 and earlier files, TU for R2007+ files.
      X : special form
      U : unknown
      SN : 16 byte sentinel
      BE : BitExtrusion
      DD : BitDouble With Default
@@ -114,11 +114,11 @@
      01 00001111 (15)
      10 (0)
```

2.4 BITLONGLONG

-The first 1-3 bits indicate the length 1 (see paragraph 2.1). Then 1 bytes follow, which represent the

+The first 1-3 bits indicate the length 1 (see paragraph [2.1] (#21-3b)). Then 1 bytes follow, which represent the
number (the least significant byte is first).

2.5 BITDOUBLE:

```
| 1^st 2 bits | what it is |
@@ -303,11 +303,11 @@
```

For R13-R14, this is a BD. For R2000+, this is a single bit followed optionally by a B D. If the bit is one, the thickness value is assumed to be 0.0. If the bit is 0, then a BD that represents the thickness follows.

2.11 CmColor

-R15 and earlier: BS color index

+R2000 and earlier: BS color index

R2004+: There are two types of color definitions, below named as CMC and ENC:

CMC:

```
@@ -375,10 +375,12 @@
| 0xC | result is reference handle minus offset
```

We will call these OFFSETOBJHANDLES. These handles are described with (CODE X), where X indicates the code if the offset is an ABSOLUTE reference (0x2 â\200\223 0x5).

COUNTER tells how many bytes of HANDLE follow.

+In most cases the COUNTER is not larger than 4, meaning the HANDLE is max 32bit (4 bytes).

+But there are some cases with COUNTER 5, revealing some undocumented bit in this HANDLE, the value is still 32bit then.

EXAMPLE: An entity on a layer whose handle is 5E7 has the following handle reference near the end of the entity data (its code being 5):

```
5 2 0 5 E 7
01010010 00000101 11100111 (0101.0010.00000101.11100111)
@@ -453,11 +455,11 @@
```

This function takes as its input an initial CRC value, a pointer to the data to be CRC'd, and the number of bytes of data. The return value is the new CRC. This function can be used to accumulate a CRC by running the first set of bytes with an initial value of 0 (or the "starting value" for this type of object), and subsequent calls with the initial value equal to the last returned CRC.

2.14.2 32-bit CRC

-From R18 onwards a 32-bit CRC is used. The algorithm is similar to the 8-bit version, but uses a CRC lookup table containing 256 32-bit values.

+From R18/R2004 onwards a 32-bit CRC is used. The algorithm is similar to the 8-bit version, but uses a CRC lookup table containing 256 32-bit values.

```
```c
OdUInt32 crc32Table[] =
{
@@ -517,17 +519,17 @@
 return ~invertedCrc;
}
```
```

-# 3 R13-R15 DWG FILE FORMAT ORGANIZATION

+# 3 R13-R2000 DWG FILE FORMAT ORGANIZATION

3.1 FILE STRUCTURE

The structure of the DWG file format changed between R13 C2 and R13 C3. Notations regarding C3 below indicate the differences.

-The general arrangement of data in an R13/R14/R15 file is as follows:
+The general arrangement of data in an R13/R14/R2000 file is as follows:

```

HEADER
  FILE HEADER
  DWG HEADER VARIABLES
  CRC
@@ -537,26 +539,42 @@
  PADDING (R13C3 AND LATER, 200 bytes, minutes the template section above if present
)
  IMAGE DATA (PRE-R13C3)
  OBJECT DATA
    All entities, table entries, dictionary entries, etc. go in this section.
  OBJECT MAP
-  OBJECT FREE SPACE (optional)
-  TEMPLATE (R14-R15, optional)
+  OBJECT FREE SPACE (R14-R2000, optional)
  SECOND HEADER
+  TEMPLATE (R14-R2000, optional)
  IMAGE DATA (R13C3 AND LATER)

```

3.2 FILE HEADER

3.2.1 VERSION ID:

The first 6 bytes are:

| | Bytes (ascii encoded) | Version |
|----------|-----------------------|---------------|
| | :----- | :----- |
| + MC0.0 | | MicroCAD R1.1 |
| + AC1.2 | | R1.2 |
| + AC1.3 | | R1.3 |
| + AC1.40 | | R1.4 |
| + AC1.50 | | R2.0 |
| + AC2.10 | | R2.10 |
| + AC2.21 | | R2.21 |
| + AC2.22 | | R2.22 |
| + AC1001 | | R2.4 |
| + AC1002 | | R2.5 |
| + AC1003 | | R2.6 |
| + AC1004 | | R9 |
| + AC1006 | | R10 |
| + AC1009 | | R11 |
| AC1012 | | R13 |
| + AC1013 | | R13C3 |
| AC1014 | | R14 |
| AC1015 | | R2000 |
| + AC1016 | | R2000i |
| AC1018 | | R2004 |
| AC1021 | | R2007 |
| AC1024 | | R2010 |
| AC1027 | | R2013 |
| AC1032 | | R2018 |

@@ -567,20 +585,68 @@

At 0x0D is a seeker (4 byte long absolute address) for the beginning sentinel of the image data.

3.2.3 OBJECT FREE SPACE

-***TODO.**

+See [chapter 21](#21-data-section-acdbobjfreespace).

3.2.4 TEMPLATE

-This section is optional, see chapter 22.

+This section is optional, see [chapter 22] (#22-data-section-acdbtemplate).

3.2.5 DWGCODEPAGE:

Bytes at 0x13 and 0x14 are a raw short indicating the value of the code page for this drawing file.

| + Codepage | Name |
|------------|---|
| + 0 | UTF8 (Unused) |
| + 1 | US_ASCII |
| + 2 | ISO-8859-1 |
| + 3 | ISO-8859-2 |
| + 4 | ISO-8859-3 |
| + 5 | ISO-8859-4 |
| + 6 | ISO-8859-5 |
| + 7 | ISO-8859-6 |
| + 8 | ISO-8859-7 |
| + 9 | ISO-8859-8 |
| + 10 | ISO-8859-9 |
| + 11 | CP437 (DOS English) |
| + 12 | CP850 (DOS Latin-1) |
| + 13 | CP852 (DOS Central European) |
| + 14 | CP855 (DOS Cyrillic) |
| + 15 | CP857 (DOS Turkish) |
| + 16 | CP860 (DOS Portuguese) |
| + 17 | CP861 (DOS Icelandic) |
| + 18 | CP863 (DOS Hebrew) |
| + 19 | CP864 (DOS Arabic IBM) |
| + 20 | CP865 (DOS Nordic) |
| + 21 | CP869 (DOS Greek) |
| + 22 | CP932 (DOS Japanese, shiftjis) |
| + 23 | MACINTOSH |
| + 24 | BIG5 |
| + 25 | CP949 (Korean, Wansung + Johab) |
| + 26 | JOHAB |
| + 27 | CP866 (Russian) |
| + 28 | ANSI-1250 (Windows Central + Eastern European) |
| + 29 | ANSI-1251 (Windows Cyrillic) |
| + 30 | ANSI-1252 (Windows Western European) |
| + 31 | GB2312 (Windows EUC-CN Chinese) |
| + 32 | ANSI-1253 (Windows Greek) |
| + 33 | ANSI-1254 (Windows Turkish) |
| + 34 | ANSI-1255 (Windows Hebrew) |
| + 35 | ANSI-1256 (Windows Arabic) |
| + 36 | ANSI-1257 (Windows Baltic) |
| + 37 | ANSI-874 (Windows Thai) |
| + 38 | ANSI-932 (Windows Japanese, extended shiftjis, windows-31j) |
| + 39 | ANSI-936 (Windows Simplified Chinese) |
| + 40 | ANSI-949 (Windows Korean Wansung) |
| + 41 | ANSI-950 (Windows Trad Chinese) |
| + 42 | ANSI-1361 (Windows Korean Wansung) |
| + 43 | UTF16 (Default since R2007) |
| + 44 | ANSI-1258 (Windows Vietnamese) |

3.2.6 SECTION-LOCATOR RECORDS:

At 0x15 is a long that tells how many sets of recno/seeker/length records follow. Each record has the following format:

Record number (raw byte) | Seeker (raw long) | Size (raw long)
 @@ -590,26 +656,20 @@

- 0 : Header variables (covers beginning and ending sentinels).
- 1 : Class section.
- 2 : Object map.

```

- 3 : (C3 and later.) A special table (no sentinels). See unknown section (R13 C3 and
d
- later). The presence of the 4th record (3) indicates that the C3 file format
- applies. Just look at the long at 21; if it's 4 or greater, it's the C3-and-la
ter
- format.
- 4 : In R13-R15, points to a location where there may be data stored. Currently we
- have seen only the MEASUREMENT variable stored here. See chapter 22.
+ 3 : R13 and later: OBJECT FREE SPACE (optional, without sentinels),
+ followed by the SECOND HEADER (with sentinels).
+ 4 : In R13-R2000, TEMPLATE with the MEASUREMENT variable. See chapter 22.
+ This section is optional.
+ 5: Auxheader. See chapter 27.
+ This section is optional.

```

-Remarks: We have seen files with up to 6 sets in this section; the meaning of the sixth one is unknown. The Open Design Toolkit emits files with the first 5 sets only.

+Remarks: We have seen files with up to 6 sets in this section. The Open Design Toolkit emits files with the first 5 sets only.

```

- RS : CRC for BOF to this point. Use 0 for the initial value, and depending on the
- number of sets of section-locators, XOR the result with one of the following:
- 3 : 0xA598
- 4 : 0x8101
- 5 : 0x3CC4
- 6 : 0x8461
+ RS : CRC from 0 to to this point, with the standard seed 0xC0C1

```

The following 16 byte sentinel appears after the CRC:

@@ -690,11 +750,11 @@

| | | |
|--------|---|---|
| 0x50 | 4 | Section Page Map Id |
| 0x54 | 8 | Section Page Map address (add 0x100 to this value) |
| 0x5C | 4 | Section Map Id |
| 0x60 | 4 | Section page array size |
| 0x64 | 4 | Gap array size |
| - 0x68 | 4 | CRC 32 (long). See paragraph 2.14.2 for the 32-bit |
| + 0x68 | 4 | CRC 32 (long). See paragraph [2.14.2] (#2142-32-bit-crc) for the 32-bit |
| | | CRC calculation, the seed is zero. Note that the |
| | | CRC calculation is done including the 4 CRC bytes |
| | | that are initially zero! So the CRC calculation takes |
| | | into account all of the 0x6c bytes of the data in this |
| | | table. |

@@ -772,25 +832,25 @@

4.5 2004 Data section map

The data section map is a map for locating all data sections (i.e. system sections are not present in this map).

-The uncompressed Section Info section contains the following data:

+The decompressed Section Info section contains the following data:

| | Offset | Length | Description |
|---|--------|--------|--|
| | :----- | :----- | :----- |
| | 0x00 | 4 | Number of section descriptions (NumDescriptions) |
| - | 0x04 | 4 | 0x02 (long) |
| - | 0x08 | 4 | 0x00007400 (long) |
| - | 0x0C | 4 | 0x00 (long) |
| + | 0x04 | 4 | Compressed 0x02 (long) |
| + | 0x08 | 4 | Max size 0x7400 (long) |
| + | 0x0C | 4 | Encrypted 0x00 (long) |
| | 0x10 | 4 | Unknown (long), ODA writes NumDescriptions here. |

Next, the following data is repeated NumDescriptions times:

| | Offset | Length | Description |
|-----------------------|--------|--------|--|
| - | 0x00 | 8 | Size of section (OdUInt64) |
| + | 0x00 | 8 | Size of section (uint64_t) |
| | 0x08 | 4 | Page count (PageCount). Note that there can be more pages than PageCount, as PageCount is just the number of pages written to file. If a page contains zeroes only, that page is not written to file. These zero pages can be detected by checking if the page's start offset is bigger than it should be based on the sum of previously read pages decompressed size (including zero pages). After reading all pages, if the total decompressed size of the pages is not equal to the section's size, add more zero pages to the section until this condition is met. |
| | 0x0C | 4 | Max Decompressed Size of a section page of this type (normally 0x7400) |
| | 0x10 | 4 | Unknown (long) |
| | 0x14 | 4 | Compressed (1 = no, 2 = yes, normally 2) |
| | 0x18 | 4 | Section Id (starts at 0). The first section (empty section) is numbered 0, consecutive sections are numbered descending from (the number of sections - 1) down to 1. |
| @@ -869,13 +929,13 @@ | | | |
| | 0x00 | 4 | Section page type, since it's always a data section: 0x4163043b |
| | 0x04 | 4 | Section number |
| | 0x08 | 4 | Data size (compressed) |
| | 0x0C | 4 | Page Size (decompressed) |
| | 0x10 | 4 | Start Offset (in the decompressed buffer) |
| - | 0x14 | 4 | Page header Checksum (section page checksum calculated from uncompressed header bytes, with the data checksum as seed) |
| - | 0x18 | 4 | Data Checksum (section page checksum calculated from compressed data bytes, with seed 0) |
| - | 0x1C | 4 | Unknown (ODA writes a 0) |
| + | 0x14 | 4 | Unknown (ODA writes a 0) |
| + | 0x18 | 4 | Page header Checksum (section page checksum calculated from uncompressed header bytes, with the data checksum as seed) |
| + | 0x1C | 4 | Data Checksum (section page checksum calculated from compressed data bytes, with seed 0) |

Each section page must start on a 0x20 byte boundary of the raw data stream. The empty bytes between the start of this section and then end of the previous section are filled with as many bytes as needed from the magic number sequence.

4.7 Compression

@@ -966,11 +1026,11 @@

5 R2007 DWG FILE FORMAT ORGANIZATION

5.1 Sections and pages overview

-Like the R18 format the R21 format has sections and pages. There are system sections and data sections.

+Like the R18/R2004 format the R21/R2007 format has sections and pages. There are system sections and data sections.

The system sections contain information about where the data sections and their pages are in the stream.

A system section only has a single page, while a data section can have multiple pages. The page map contains information about where each data page is in the file stream. The section map has information about which pages belong to which section. The file header, which is at the beginning of the file, just after the meta data, contains the stream locations of the page map and section map.

@@ -1010,11 +1070,11 @@

| | | | |
|--|------|---|---|
| | 0x0D | 4 | Preview address (long) |
| | 0x11 | 1 | Dwg version (Acad version that writes the file) |
| | 0x12 | 1 | Maintenance release version (Acad maintenance version that writes the file) |

| | | | | |
|---|------|---|---|--|
| | 0x13 | 2 | Codepage | |
| | 0x15 | 3 | Unknown (ODA writes zeroes) | |
| - | 0x18 | 4 | SecurityType (long), see R2004 meta data, the definition is the same, paragraph 4.1. | |
| + | 0x18 | 4 | SecurityType (long), see R2004 meta data, the definition is the same, paragraph [4.1] (#41-r2004-file-header) | |
| | 0x1C | 4 | Unknown long | |
| | 0x20 | 4 | Summary info Address in stream | |
| | 0x24 | 4 | VBA Project Addr (0 if not present) | |
| | 0x28 | 4 | 0x00000080 | |
| | 0x2C | 4 | Application Info Addr | |

@@ -1078,11 +1138,11 @@

The page map is stored in a single system section page. The page size of this system section page depends on how much data is stored in it. One page should be able to fit ((dataSectionPageCount + 5) * 16) bytes.

PagesMapOffset indicates the starting address of the Page Map section of the file, PagesMapSizeCompressed is the compressed size of this section, PagesMapSizeUncompressed is the uncompressed size, PagesMapCorrectionFactor is the correction factor used, and PagesMapCrcCompressed and PagesMapCrcUncompressed are the compressed and uncompressed CRC values, respectively. The data at PagesMapOffset is in the following format (to be referred to as â\200\234System Pageâ\200\235 format throughout the remainder of this document) should be decoded and optionally decompressed using the OdDwgR21FileController::loadSysPage function. The resulting pages map data consists of a sequence of pairs, where each pair consists of an Int64 **SIZE** value, and an Int64 **ID** value. This sequence creates a set of pages where each. These values create a pages map using the following algorithm:

```

```c
-OdInt64 offset = 0;
+int64 offset = 0;

while (!pStream->isEof()) {
 size = OdPlatformStreamer::rdInt64(*pStream);
 id = OdPlatformStreamer::rdInt64(*pStream);
 ind = id > 0 ? id : -id;
@@ -1093,11 +1153,11 @@
}
```

```

The **File Header** value PagesMaxId indicates the largest index that will be used for the m_pages array.

-Next, the Section Map should be loaded. The offset of the section map data is the m_offset value of the page with index SectionsMapId in the Page Map of the file. The File Header values SectionsMapSizeCompressed, SectionsMapSizeUncompressed, SectionsMapCrcCompressed, SectionsMapCrcUncompressed, and SectionsMapCorrectionFactor make of the remainder of the arguments to pass to the OdDwgR21FileController::loadSysPage function (see paragraph 5.3) for decoding and decompression of the Section Map data. The decoded and decompressed Section Map data consists of the following attributes for each section in the file:

+Next, the Section Map should be loaded. The offset of the section map data is the m_offset value of the page with index SectionsMapId in the Page Map of the file. The File Header values SectionsMapSizeCompressed, SectionsMapSizeUncompressed, SectionsMapCrcCompressed, SectionsMapCrcUncompressed, and SectionsMapCorrectionFactor make of the remainder of the arguments to pass to the OdDwgR21FileController::loadSysPage function (see paragraph [5.3] (#53-system-section-page)) for decoding and decompression of the Section Map data. The decoded and decompressed Section Map data consists of the following attributes for each section in the file:

| Address | Length | Description |
|---------|--------|-------------|
| :----- | :----- | :----- |
| 0x00 | 8 | Data size |
| 0x08 | 8 | Max size |

@@ -1192,21 +1252,21 @@

By default data/properties are not encrypted. Encryption still needs to be described.

5.2.1 File header creation

-Creating the R21 file header is very complex:

+Creating the R2007 file header is very complex:

-Compute and set all the file header fields. In this process also compute CRC and generate check data, derived from a CRC seed value (paragraph 5.2.1.1).

+Compute and set all the file header fields. In this process also compute CRC and generate check data, derived from a CRC seed value (paragraph [5.2.1.1] (#5211-calculating-the-file-header-crcs-and-check-data)).

-Write the file header data to a buffer and calculate/write the 64-bit CRC (paragraph 5.2.1.2).

+Write the file header data to a buffer and calculate/write the 64-bit CRC (paragraph [5.2.1.2] (#5212-calculate-file-header-data-64-bit-crc-decompressed)).

-Compress the file header data and calculate the 64-bit CRC (paragraph 5.2.1.3).

+Compress the file header data and calculate the 64-bit CRC (paragraph [5.2.1.3] (#5213-compress-and-calculate-64-bit-crc-compressed)).

-Create a checking sequence and calculate a CRC over this sequence data (paragraph 5.2.1.4).

+Create a checking sequence and calculate a CRC over this sequence data (paragraph [5.2.1.4] (#5214-create-checking-sequence-and-64-bit-crc)).

-Create a buffer in preparation of Reed-Solomon encoding (Pre-Reed-Solomon encoded data). This contains checking sequence, compressed CRC, compressed size, compressed data and random data (as padding) (paragraph 5.2.1.5).

+Create a buffer in preparation of Reed-Solomon encoding (Pre-Reed-Solomon encoded data). This contains checking sequence, compressed CRC, compressed size, compressed data and random data (as padding) (paragraph [5.2.1.5] (#5215-create-a-buffer-in-preparation-of-reed-solomon-encoding)).

Encode the data using Reed-Solomon (for error correction).

Write the encoded data, followed by the check data from the first step.

@@ -1214,23 +1274,23 @@

The file header data consists of regular data fields and CRC values and check data to verify the data's correctness. All fields pertaining to the file header's correctness are discussed in more detail in the following paragraphs. Note that the order of CRC calculation is important, so the order of the following paragraphs should be used.

5.2.1.1.1 RandomSeed

-Is filled with the CRC random encoding's seed (see paragraph 5.11).

+Is filled with the CRC random encoding's seed (see paragraph [5.11] (#511-crc-random-encoding)).

5.2.1.1.2 CrcSeed

The ODA always initializes this with value 0.

5.2.1.1.3 SectionsMapCrcSeed

-Is filled with crcSeed initially. Then it's encoded using the CRC random encoding as described in paragraph 5.11.

+Is filled with crcSeed initially. Then it's encoded using the CRC random encoding as described in paragraph [5.11] (#511-crc-random-encoding).

5.2.1.1.4 PagesMapCrcSeed

-Is filled with crcSeed initially. Then it's encoded using the CRC random encoding

ing as described in paragraph 5.11.

+Is filled with crcSeed initially. Then itâ\200\231s encoded using the CRC random encoding as described in paragraph [5.11] (#511-crc-random-encoding).

5.2.1.1.5 Check data

The check data for the file header page is present at the end of the header page at location 0x3d8. It contains data generated based on the CrcSeed and the current state of the CRC random encoder. The check data contains the following UInt64 fields (computed in this order):

```
@@ -1302,44 +1362,44 @@  
}  
...
```

5.2.1.1.6 CrcSeedEncoded

-Encoded value of CrcSeed, using the CRC random encoding as described in paragraph 5.11.

+Encoded value of CrcSeed, using the CRC random encoding as described in paragraph [5.11] (#511-crc-random-encoding).

5.2.1.2 Calculate file header data 64-bit CRC (decompressed)

-The last field in the file header is a normal 64-bit CRC (see paragraph 5.12) which is the CRC calculated from the file header data, including the 64-bit CRC with value zero. The CRC seed value is 0, and then updated with method UpdateSeed2 before calling UpdateCrc (see again paragraph 5.12). The initial CRC value of 0 is replaced with the calculated value.

+The last field in the file header is a normal 64-bit CRC (see paragraph [5.12] (#512-64-bit-crc-calculation)) which is the CRC calculated from the file header data, including the 64-bit CRC with value zero. The CRC seed value is 0, and then updated with method UpdateSeed2 before calling UpdateCrc (see again paragraph [5.12] (#512-64-bit-crc-calculation)). The initial CRC value of 0 is replaced with the calculated value.

5.2.1.3 Compress and calculate 64-bit CRC (compressed)

-The file header data is compressed. If the compressed data is not shorter than the uncompressed data, then the uncompressed data itself is used. Another normal 64-bit CRC value is calculated from the resulting data (see paragraph 5.12).

+The file header data is compressed. If the compressed data is not shorter than the uncompressed data, then the uncompressed data itself is used. Another normal 64-bit CRC value is calculated from the resulting data (see paragraph [5.12] (#512-64-bit-crc-calculation)).

5.2.1.4 Create checking sequence and 64-bit CRC

-Another checking sequence of 2 UInt64 values is created, very similar to the check data in paragraph 5.2.1.1.5. The first value is filled with the next value from the random encoder (see paragraph 5.11). The second value is calculated using the check dataâ\200\231s Encode function, with the first sequence value passed as first (value) and second (control) parameter. The sequence bytes are then converted to little endian format. The last step is calculating a normal 64-bit CRC value (see paragraph 5.12). The CRC seed value is 0, updated by method UpdateSeed1.

+Another checking sequence of 2 UInt64 values is created, very similar to the check data in paragraph [5.2.1.1.5.] (#5.2.1.1.5.) The first value is filled with the next value from the random encoder (see paragraph [5.11] (#511-crc-random-encoding)). The second value is calculated using the check dataâ\200\231s Encode function, with the first sequence value passed as first (value) and second (control) parameter. The sequence bytes are then converted to little endian format. The last step is calculating a normal 64-bit CRC value (see paragraph [5.12] (#512-64-bit-crc-calculation)). The CRC seed value is 0, updated by method UpdateSeed1.

5.2.1.5 Create a buffer in preparation of Reed-Solomon encoding

-In preparation of the next step, which is Reed-Solomon (RS) encoding, a buffer is created which is going to be encoded. The size of this buffer is 3 x 239 bytes (239 is the

RS data size for a block (k) used for system pages, see paragraph 5.13). First a block is created, of which the size is a multiple of 8 bytes:

+In preparation of the next step, which is Reed-Solomon (RS) encoding, a buffer is created which is going to be encoded. The size of this buffer is 3 x 239 bytes (239 is the RS data size for a block (k) used for system pages, see paragraph [5.13] (#513-reed-solomon-encoding)). First a block is created, of which the size is a multiple of 8 bytes:

| | Position | Size | Description |
|---|----------|------|--|
| - | 0 | 8 | Checking sequence CRC (paragraph 5.2.1.4) |
| - | 8 | 8 | Checking sequence first UInt64 value (paragraph 5.2.1.4) |
| - | 16 | 8 | Compressed data CRC (paragraph 5.2.1.3) |
| + | 0 | 8 | Checking sequence CRC (paragraph [5.2.1.4] (#5214-create-checking-sequence-and-64-bit-crc)) |
| + | 8 | 8 | Checking sequence first UInt64 value (paragraph [5.2.1.4] (#5214-create-checking-sequence-and-64-bit-crc)) |
| + | 16 | 8 | Compressed data CRC (paragraph [5.2.1.3] (#5213-compress-and-calculate-64-bit-crc-compressed)) |
| | 24 | 8 | Compressed data size. In case the compressed data size is larger than the uncompressed data size, then the negated uncompressed data size is written. |
| | 32 | n | Compressed data in case the size is smaller than the uncompressed data size. Otherwise the uncompressed data. |
| - | 32 + n | m | Padding so the block size is a multiple of 8 bytes. The padding bytes are gotten from the CRC random encoding, see paragraph 5.11. This block is repeated as many times as possible within the buffer. The remaining bytes are filled using random padding data from the CRC random encoding (see paragraph 5.11). |
| + | 32 + n | m | Padding so the block size is a multiple of 8 bytes. The padding bytes are gotten from the CRC random encoding, see paragraph [5.11] (#511-crc-random-encoding). This block is repeated as many times as possible within the buffer. The remaining bytes are filled using random padding data from the CRC random encoding (see paragraph [5.11] (#511-crc-random-encoding)). |

5.2.1.6 Encode the data using Reed-Solomon

-In this step the header data is encoded using the Reed-Solomon (RS) encoding for interleaved system pages (see paragraph 5.13). The encoded size is 3 x 255 bytes. The remaining bytes of the page (of total size 0x400) are filled using random padding data from the CRC random encoding (see paragraph 5.11).

+In this step the header data is encoded using the Reed-Solomon (RS) encoding for interleaved system pages (see paragraph [5.13] (#513-reed-solomon-encoding)). The encoded size is 3 x 255 bytes. The remaining bytes of the page (of total size 0x400) are filled using random padding data from the CRC random encoding (see paragraph [5.11] (#511-crc-random-encoding)).

5.2.1.7 Add check data at the end of the page

-The last 0x20 bytes of the page should be overwritten using the check data, calculated in paragraph 5.2.1.1.5. The page size remains 0x400 bytes.

+The last 0x20 bytes of the page should be overwritten using the check data, calculated in paragraph [5.2.1.1.5.] (#5.2.1.1.5.) The page size remains 0x400 bytes.

5.2.1.8 Write the file header to the file stream

The file header is written to position 0x80 and to the end of the file stream.

@@ -1349,26 +1409,26 @@

Inputs for writing a system section page are:

* The data.

* The 64-bit CRC seed.

-* The page size (minimum 0x400). The page size is determined from the decompressed data size as described in paragraph 5.3.1.

+* The page size (minimum 0x400). The page size is determined from the decompressed data size as described in paragraph [5.3.1.] (#5.3.1.)

Outputs are:

* Compressed and Reed-Solomon (RS) encoded data.

-* Derived properties of the (compressed/encoded) data: compressed 64-bit CRC, decompressed 64 bit CRC, data repeat count (or data factor). These derived properties are written in the file header (see paragraph 5.2).

+* Derived properties of the (compressed/encoded) data: compressed 64-bit CRC, decompressed 64 bit CRC, data repeat count (or data factor). These derived properties are written in the file header (see paragraph [5.2] (#5.2)).

-First the 64-bit CRC of the decompressed data is calculated, using the mirrored 64-bit CRC calculation (see paragraph 5.12). This uses theUpdateSeed1 method to update the CRC seed before entering the CRC computation.

+First the 64-bit CRC of the decompressed data is calculated, using the mirrored 64-bit CRC calculation (see paragraph [5.12] (#512-64-bit-crc-calculation)). This uses theUpdateSeed1 method to update the CRC seed before entering the CRC computation.

Next step is compression. If the compressed data isnâ\200\231t shorter than the original data, then the original data is used instead of the compressed data.

Of the resulting data (either compressed or not), another 64-bit CRC is computed (similarly to described above).

The resulting data is padded with zeroes so the length is a multiple of the CRC block size (8).

-Now the resulting data is repeated as many times as possible within the page, RS encoded (see paragraph 5.13) and padded. The maximum RS block count (integer) is the page size divided by the RS codeword size (255). The maximum RS pre-encoded size is the maximum RS block count times the k value of the RS system page encoding (239). So the data repeat count is the maximum RS pre-encoded size divided by the resulting (padded) data length. Next a buffer is created, with the resulting (padded) data repeated (data repeat count times). This buffer is encoded using RS encoding for system pages, interleaved. Note that the actual RS block count is less than or equal to the maximum RS block count calculated above. The encoded size is the RS block count times 255. The final step is to add padding using random data from the random encoding to fill the remainder of the page, see paragraph 5.11.

+Now the resulting data is repeated as many times as possible within the page, RS encoded (see paragraph [5.13] (#513-reed-solomon-encoding)) and padded. The maximum RS block count (integer) is the page size divided by the RS codeword size (255). The maximum RS pre-encoded size is the maximum RS block count times the k value of the RS system page encoding (239). So the data repeat count is the maximum RS pre-encoded size divided by the resulting (padded) data length. Next a buffer is created, with the resulting (padded) data repeated (data repeat count times). This buffer is encoded using RS encoding for system pages, interleaved. Note that the actual RS block count is less than or equal to the maximum RS block count calculated above. The encoded size is the RS block count times 255. The final step is to add padding using random data from the random encoding to fill the remainder of the page, see paragraph [5.11] (#511-crc-random-encoding).

5.3.1 System section page size calculation

The data stored in a system section is first padded until its size is a multiple of the CRC block size (8). This is called the aligned size. The Reed-Solomon encoded aligned data should fit the system section at least two times. The minimum page size is 0x400 bytes.

@@ -1403,21 +1463,21 @@

5.4 Data section page

Data sections are used for all sections except the data section map and the section page map. The sectionâ\200\231s data is partitioned into pages, each of Max size length, except for the last page which may be of size less than Max size. The following steps are taken when writing data page.

-First a 32-bit data checksum of the pageâ\200\231s data is calculated. The pseudo code for this calculation is presented in paragraph 5.4.1.

+First a 32-bit data checksum of the pageâ\200\231s data is calculated. The pseudo code

for this calculation is presented in paragraph [5.4.1.] (#5.4.1.)

Next the page data is optionally compressed (depending on the section). If the compressed data isn't shorter than the original data, then this page's data is not compressed.

If the file is encrypted, the page is encrypted (to be described).

-The page's 64-bit CRC is calculated (mirrored CRC, see paragraph 5.12). The page CRC seed is the file's CRC seed updated using UpdateSeed1 (see again paragraph 5.12).

+The page's 64-bit CRC is calculated (mirrored CRC, see paragraph [5.12] (#512-64-bit-crc-calculation)). The page CRC seed is the file's CRC seed updated using UpdateSeed1 (see again paragraph [5.12] (#512-64-bit-crc-calculation)).

Pad the data with zero bytes so the size becomes a multiple of the CRC block size (0x8).

-The data is Reed-Solomon encoded (see paragraph 5.13). Depending on the section encoding, the data is either interleaved (value 4) or not (value 1).

+The data is Reed-Solomon encoded (see paragraph [5.13] (#513-reed-solomon-encoding)). Depending on the section encoding, the data is either interleaved (value 4) or not (value 1).

The page start position should be aligned on a 0x20 byte boundary (if all is well nothing has to be done at this point to achieve this). The data is written and padded with zero bytes so the stream position is again at a 0x20 byte boundary.

Finally the current page ID is incremented.

@@ -1506,11 +1566,11 @@

The AcDb:Security section is optional in the file; it is present if the file was saved with a password. The data in this section is in the same format as in the R2004 format, 2 unknown 32-bit integers, a 32-bit integer with value 0xABCDABCD, etc.

5.6 AcDb:AuxHeader Section

-This section is in the same format as in R2004. See details in chapter 27.

+This section is in the same format as in R2004. See details in [chapter 27] (#27-data-section-acdbauxheader-auxiliary-file-header).

5.7 AcDb:Handles Section

This section is in the same format as in R2004.

@@ -1557,15 +1617,15 @@

We read sets of these until we exhaust the data.

5.9 AcDb:Header Section

-This section contains the "DWG Header Variables" data in a similar format as R15 files (see details in the DWG HEADER VARIABLES section of this document), except that string data is separated out into a string stream. See the Objects Section for details about string stream location within an object. Also, the handles are separated out into a separate stream at the end of the header, in the same manner as is done for Objects.

+This section contains the "DWG Header Variables" data in a similar format as R2000 files (see details in the DWG HEADER VARIABLES section of this document), except that string data is separated out into a string stream. See the Objects Section for details about string stream location within an object. Also, the handles are separated out into a separate stream at the end of the header, in the same manner as is done for Objects.

5.10 Decompression

-The compression uses another variant of the LZ77 algorithm, different from the one used in R18. Like the R18 compression, the compressed stream (source buffer) contains opco

des, offsets and lengths of byte chunks to be copied from either compressed or decompressed buffer.

+The compression uses another variant of the LZ77 algorithm, different from the one used in R18/R2004. Like the R18/R2004 compression, the compressed stream (source buffer) contains opcodes, offsets and lengths of byte chunks to be copied from either compressed or decompressed buffer.

An opcode consists of a single byte. The first byte contains the first opcode. If the first opcode's high nibble equals a 2, then:

* the source buffer pointer is advanced 2 bytes, and a length is read from the next byte, bitwise and-ed with 0x07

@@ -1993,15 +2053,15 @@

0xa6df411fbfb21ca3, 0xdc0731d78f8795da, 0x536fa08fd90e51, 0x29b7d047efec8728
...

5.13 Reed-Solomon encoding

-R21 uses Reed-Solomon (RS) encoding to add error correction. Error correction codes are typically used in telecommunication to correct errors during transmission or on media to correct e.g. errors caused by a scratch on a CD. RS coding takes considerably study to master, and books on the subject require at least some mathematical base knowledge on academic level. For this reason it's recommended to use an existing RS implementation, rather than to build one from scratch. When choosing to learn about the subject, a good book on the subject is "Error Control Coding, Second Edition" (2002), by Shu Lin and Daniel J. Costello, Jr. This book is taught over two semesters, to give an idea of the depth of the subject. RS coding is treated in Chapter 7 out of 22, to have a full understanding of the subject chapters 1-7 should be read.

+R2007 uses Reed-Solomon (RS) encoding to add error correction. Error correction codes are typically used in telecommunication to correct errors during transmission or on media to correct e.g. errors caused by a scratch on a CD. RS coding takes considerably study to master, and books on the subject require at least some mathematical base knowledge on academic level. For this reason it's recommended to use an existing RS implementation, rather than to build one from scratch. When choosing to learn about the subject, a good book on the subject is "Error Control Coding, Second Edition" (2002), by Shu Lin and Daniel J. Costello, Jr. This book is taught over two semesters, to give an idea of the depth of the subject. RS coding is treated in Chapter 7 out of 22, to have a full understanding of the subject chapters 1-7 should be read.

An open source RS implementation is available from <<http://www.eccpage.com/>>, item "Reed-Solomon (RS) codes", by Simon Rockliff, 1989. This implementation uses Berlekamp-Massey for decoding. Note that there are many ways to encode and decode, the implementation above is just one example. Though only 404 lines of code, the math involved is very sophisticated.

-DWG file format version R21 uses two configurations of RS coding:

+DWG file format version R2007 uses two configurations of RS coding:

* Data pages: use a (n, k) of (255, 251), the primitive polynomial coefficients being (1, 0, 1, 1, 1, 0, 0, 0). This configuration can correct $(255 - 251) / 2 = 2$ error bytes per block of 255 bytes. For each 251 data bytes (k), 4 parity bytes are added to form a 255 byte (code word) block.

* System pages: use a (n, k) of (255, 239), the primitive polynomial coefficients being (1, 0, 0, 1, 0, 1, 1, 0). This configuration can correct $(255 - 239) / 2 = 8$ error bytes per block of 255 bytes. For each 239 data bytes (k), 16 parity bytes are added to form a 255 byte (code word) block.

@@ -2011,54 +2071,54 @@

5.13.1 Non-interleaved

All original data blocks are followed by the parity byte blocks (i.e. the first parity block follows the last data block).

-When the last block is not entirely filled, then random bytes are added from the random

m encoding (see paragraph 5.11) to fill the block to have size k.

+When the last block is not entirely filled, then random bytes are added from the random encoding (see paragraph [5.11](#511-crc-random-encoding)) to fill the block to have size k.

5.13.2 Interleaved

When more than 1 block of data is encoded, the encoded block data is interleaved. E.g. when there are 3 blocks to be encoded, then the data bytes and parity bytes of the first block are written to positions $3 \times i$ (where i is an integer ≥ 0). The encoded bytes of the second block are written to positions $3 \times i + 1$ and of the third block to positions $3 \times i + 2$.

-When the last block is not entirely filled, then random bytes are added from the random encoding (see paragraph 5.11) to fill the block to have size k.

+When the last block is not entirely filled, then random bytes are added from the random encoding (see paragraph [5.11](#511-crc-random-encoding)) to fill the block to have size k.

6 R2010 DWG FILE FORMAT ORGANIZATION

-The 2010 format is based mostly on the 2004 format and somewhat on the 2007 format. The file header, page map, section map, compression are the same as in R2004. The bit coding is the same as in R2007 (see chapter 2), with the exception of the Object Type being encoded differently (see paragraph 2.12).

+The 2010 format is based mostly on the 2004 format and somewhat on the 2007 format. The file header, page map, section map, compression are the same as in R2004. The bit coding is the same as in R2007 (see [chapter 2](#2-bit-codes-and-data-definitions)), with the exception of the Object Type being encoded differently (see paragraph [2.12](#212-object-type)).

Like the R2007 format, the data, strings and handles are separated in header and objects sections.

7 R2013 DWG FILE FORMAT ORGANIZATION

The 2013 format is based mostly on the 2010 format. The file header, summary info, page map, section map, compression are the same as in R2004. The bit coding is the same as in R2010. Like the R2007 format, the data, strings and handles are separated in header and objects sections. The changes in the Header section are minor (only 2 added fields).

-A new data section was introduced, the data storage section (AcDb:AcDsPrototype_1b). At this moment (December 2012), this section contains information about Acis data (regions, solids). See chapter 24 for more details about this section.

+A new data section was introduced, the data storage section (AcDb:AcDsPrototype_1b). At this moment (December 2012), this section contains information about Acis data (regions, solids). See [chapter 24](#24-section-acdbacdsprototype_1b-datastorage) for more details about this section.

Note that at the point of writing (22 March 2013) known valid values for acad maintenance version are 6 and 8. The ODA currently writes value 8.

8 R2018 DWG FILE FORMAT ORGANIZATION

The AutoCAD 2018 format is almost identical to the 2013 format. Structurally they are identical.

Below is a summary of the changes:

-* Three shorts (int16) with value zero have been added to end of the auxiliary file header (see chapter 27).

+* Three shorts (int16) with value zero have been added to end of the auxiliary file header (see [chapter 27](#27-data-section-acdbauxheader-auxiliary-file-header)).

-* In the AcDb:Header nothing changed, but note that the unknown 32-bit int at the start, directly following the section size that was present for R2010/R2013 for acad maintenance

nance version greater than 3, is also present for R2018 (see chapter 9).

+* In the AcDb:Header nothing changed, but note that the unknown 32-bit int at the start, directly following the section size that was present for R2010/R2013 for acad maintenance version greater than 3, is also present for R2018 (see [chapter 9] (#9-data-section-acdbheader-header-variables)).

* Additions/changes in the following entities:

- * ACAD_PROXY_ENTITY (paragraph 20.4.90),
- * ATTRIB (paragraph 20.4.4),
- * ATTDEF (paragraph 20.4.5),
- * MTEXT (see paragraph 20.4.46).
+ * ACAD_PROXY_ENTITY (paragraph [20.4.90] (#20490-proxy-varies)),
+ * ATTRIB (paragraph [20.4.4] (#2044-attrib-2)),
+ * ATTDEF (paragraph [20.4.5] (#2045-attdef-3)),
+ * MTEXT (see paragraph [20.4.46] (#20446-mtext-44)).

-* Object MLINESTYLE (paragraph 20.4.73) references line types in its element by their handle rather than by index.

+* Object MLINESTYLE (paragraph [20.4.73] (#20473-mlinestyle-73)) references line types in its element by their handle rather than by index.

9 Data section AcDb:Header (HEADER VARIABLES)

-The header contains all header (system) variables, except the MEASUREMENT variable, which is present in the AcDb:Template section, see chapter 22.

+The header contains all header (system) variables, except the MEASUREMENT variable, which is present in the AcDb:Template section, see [chapter 22] (#22-data-section-acdbtemplate).

The header variables section indicated by section-locator 0 has the following form:

Beginning sentinel
Size of the section (a 4 byte long)
@@ -2066,11 +2126,11 @@
Unknown (4 byte long), might be part of a 64-bit size.
Data (system variables and possibly other data at the beginning)
CRC (covers the stepper and the data)
Ending sentinel

-This data section appear as one long stream, with no gaps. Most are bit coded. (See the BIT CODES section.) The header is padded with random bits to the next byte boundary.

+This data section appear as one long stream, with no gaps. Most are bit coded. (See the [BIT CODES section] (#2-bit-codes-and-data-definitions).) The header is padded with random bits to the next byte boundary.

The following 16 byte sentinel introduces this section:

0xCF, 0x7B, 0x1F, 0x23, 0xFD, 0xDE, 0x38, 0xA9, 0x5F, 0x7C, 0x68, 0xB8, 0x4E, 0x6D, 0x33, 0x5F
RL : Size of the section.
@@ -2082,11 +2142,11 @@

R2007 Only:
RL : Size in bits
R2013+:
- BLL : Variabele REQUIREDVERSIONS, default value 0, read only.
+ BLL : Variable REQUIREDVERSIONS, default value 0, read only.
Common:
BD : Unknown, default value 412148564080.0
BD : Unknown, default value 1.0
BD : Unknown, default value 1.0
BD : Unknown, default value 1.0
@@ -2111,20 +2171,20 @@
B : REGENMODE
B : FILLMODE
B : QTEXTMODE

```

        B : PSLTSCALE
        B : LIMCHECK
-   R13-R14 Only (stored in registry from R15 onwards):
+   R13-R14 Only (stored in registry from R2000 onwards):
        B : BLIPMODE
R2004+:
        B : Undocumented
Common:
        B : USRTIMER (User timer on/off).
        B : SKPOLY
        B : ANGDIR
        B : SPLFRAME
-   R13-R14 Only (stored in registry from R15 onwards):
+   R13-R14 Only (stored in registry from R2000 onwards):
        B : ATTREQ
        B : ATTDIA
Common:
        B : MIRRTEXT
        B : WORLDVIEW
@@ -2132,33 +2192,33 @@
        B : WIREFRAME Undocumented.
Common:
        B : TILEMODE
        B : PLIMCHECK
        B : VISRETAIN
-   R13-R14 Only (stored in registry from R15 onwards):
+   R13-R14 Only (stored in registry from R2000 onwards):
        B : DELOBJ
Common:
        B : DISPSILH
        B : PELLIPSE (not present in DXF)
        BS : PROXYGRAPHICS
-   R13-R14 Only (stored in registry from R15 onwards):
+   R13-R14 Only (stored in registry from R2000 onwards):
        BS : DRAGMODE
Common:
        BS : TREEDEPTH
        BS : LUNITS
        BS : LUPREC
        BS : AUNITS
        BS : AUPREC
-   R13-R14 Only Only (stored in registry from R15 onwards):
+   R13-R14 Only Only (stored in registry from R2000 onwards):
        BS : OSMODE
Common:
        BS : ATTMODE
-   R13-R14 Only Only (stored in registry from R15 onwards):
+   R13-R14 Only Only (stored in registry from R2000 onwards):
        BS : COORDS
Common:
        BS : PDMODE
-   R13-R14 Only Only (stored in registry from R15 onwards):
+   R13-R14 Only Only (stored in registry from R2000 onwards):
        BS : PICKSTYLE
R2004+:
        BL : Unknown
        BL : Unknown
        BL : Unknown
@@ -2201,11 +2261,11 @@
        BD : CHAMFERC
        BD : CHAMFERD
        BD : FACETRES
        BD : CMLSCALE
        BD : CELTSCALE
-   R13-R18:
+   R13-R2004:
        TV : MENUNAME

```



```

Common:
    BL : TDCREATE (Julian day)
    BL : TDCREATE (Milliseconds into the day)
    BL : TDUPDATE (Julian day)
@@ -2220,11 +2280,11 @@
    BL : TDUSRTIMER (Days)
    BL : TDUSRTIMER (Milliseconds into the day)
    CMC : CECOLOR
    H : HANDSEED The next handle, with an 8-bit length specifier preceding the ha
ndle
        bytes (standard hex handle form) (code 0). The HANDSEED is not part of the
handle
-         stream, but of the normal data stream (relevant for R21 and later).
+         stream, but of the normal data stream (relevant for R2007 and later).
    H : CLAYER (hard pointer)
    H : TEXTSTYLE (hard pointer)
    H : CELTYPE (hard pointer)
R2007+ Only:
    H : CMATERIAL (hard pointer)
@@ -2410,12 +2470,14 @@
    H : LINETYPE CONTROL OBJECT (hard owner)
    H : VIEW CONTROL OBJECT (hard owner)
    H : UCS CONTROL OBJECT (hard owner)
    H : VPORT CONTROL OBJECT (hard owner)
    H : APPID CONTROL OBJECT (hard owner)
-    H : DIMSTYLE CONTROL OBJECT (hard owner) R13-R15 Only:
-    H : VIEWPORT ENTITY HEADER CONTROL OBJECT (hard owner) Common:
+    H : DIMSTYLE CONTROL OBJECT (hard owner)
+R13-R2000 Only:
+    H : VIEWPORT ENTITY HEADER CONTROL OBJECT (hard owner)
+Common:
    H : DICTIONARY (ACAD_GROUP) (hard pointer)
    H : DICTIONARY (ACAD_MLINESSTYLE) (hard pointer)
    H : DICTIONARY (NAMED OBJECTS) (hard owner)
R2000+ Only:
    BS : TSTACKALIGN, default = 1 (not present in DXF)
@@ -2603,11 +2665,11 @@
00240 47 B1 92 CC A0          G.... 0100 0111 1011 0001 1001 0010 1100 1100 1010 0000
```

```

# 10 Data section AcDb:Classes

```

-## 10.1 R13-R15
+## 10.1 R13-R2000

```

This section contains the defined classes for the drawing.

```

 SN : 0x8D 0xA1 0xC4 0xB8 0xC4 0xA9 0xF8 0xC5 0xC0 0xDC 0xF4 0x5F 0xE7 0xCF 0xB6 0x
8A.

```

RL : size of class data area.

```

@@ -2630,13 +2692,13 @@

```

This following 16-byte sentinel appears after the CRC:

```

 0x72,0x5E,0x3B,0x47,0x3B,0x56,0x07,0x3A,0x3F,0x23,0x0B,0xA0,0x18,0x30,0x49,0x75

```

```

-For R18 and later 8 unknown bytes follow. The ODA writes 0 bytes.

```

```

+For R18/R2004 and later 8 unknown bytes follow. The ODA writes 0 bytes.

```

```

-## 10.2 R18+
+## 10.2 R2004+

```

This section is compressed and contains the standard 32 byte section header.

This section contains the defined classes for the drawing.

```

@@ -2688,15 +2750,15 @@

```

# 11 PADDING (R13C3 AND LATER)

0x200 bytes of padding. Can be ignored. When writing, the Open Design Toolkit writes a 11 0s.

-Occasionally AutoCAD will use the first 4 bytes of this area to store the value of the "measurement" variable. This padding was evidently required to allow pre-R13C3 versions of AutoCAD to read files produced by R13C3 and later.

+Occasionally AutoCAD will use the first 4 bytes of this area to store the value of the "measurement" variable, i.e the TEMPLATE section. This padding was evidently required to allow pre-R13C3 versions of AutoCAD to read files produced by R13C3 and later.

# 12 Data section: ""

-The empty data section was introduced in R18. This section contains no data.

+The empty data section was introduced in R18/R2004. This section contains no data.

Section property	Value
Name	â\200\234â\200\235
Section ID	Always 0
@@ -2726,11 +2788,11 @@	
String	2 + n Revision number
String	2 + n Hyperlink base
?	8 Total editing time (ODA writes two zero Int32â\200\231s)
Julian date	8 Create date time
Julian date	8 Modified date time
-  Int16   2 + 2 * (2 + n)   Property count, followed by PropertyCount key/value string pairs.	
+  Int16   2 + 2 * (2 + n)   Custom Property count, followed by CUSTOMPROPERTYTAG and CUSTOMPROPERTY key/value string pairs.	
Int32	4 Unknown (write 0)
Int32	4 Unknown (write 0)

# 14 Data section AcDb:Preview

@@ -2807,43 +2869,42 @@

Name	AcDb:AppInfo
Compressed	1
Encrypted	0
Page size	0x80

-The AppInfo format depends on the application version (Acad version that wrote the file) in the file header. So a R18 .dwg file might have an R21 AppInfo section.

+The AppInfo format depends on the application version (Acad version that wrote the file) in the file header. So a R2004 .dwg file might have an R2007 AppInfo section.

### 16.1 R18

+### 16.1 R2004

-In R18 the app info section consists of the following fields. Strings are encoded as a 16-bit length, followed by the character bytes (0-terminated).

+In R2004 the app info section consists of the following fields. Strings are encoded as a 16-bit length, followed by the single-character bytes (0-terminated).

Type	Length	Description
- String	2 + n	App info name, ODA writes â\200\234AppInfoDataListâ\200\235
- UInt32	4	Unknown, ODA writes 2
- String	2 + n	Unknown, ODA writes â\200\2344001â\200\235
- String	2 + n	App info product XML element, e.g. ODA writes â\200\234<ProductInformation name=â\200\235Teighaâ\200\235 build_version=â\200\2350.0â\200\235
-		registry_version=â\200\2353.3â\200\235 install_id_string=â\200\235 ODAâ\200\235
-		registry_localeID=â\200\2351033â\200\235/>â\200\234

```

-| String | 2 + n | App info version, e.g. ODA writes â\200\2342.7.2.0â\200\235.
+| String | 2 + n | App info name, ACAD writes "AppInfoData", ODA writes "AppInfoDataL
ist"
+| RL | 4 | num strings (default: 0)
+| String | 2 + n | Comment, e.g. "5004", ODA writes "4001"
+| String | 2 + n | App info product string, e.g. "Autodesk Architectural Desktop 2007
"
+| String | 2 + n | App info version, e.g. "5.0.318.0", ODA writes "2.7.2.0".

```

-### 16.2 R21-27

+### 16.2 R2007+ or class\_version == 3

-In R21 (and also R24, R27) the app info section consists of the following fields. Strings are encoded as a 16-bit length, followed by the character bytes (0-terminated), using unicode encoding (2 bytes per character).

+Since R2007 or class\_version 3 the app info section consists of the following fields. Strings are encoded as a 16-bit length, followed by 0-terminated unicode wide-chars (2 bytes per character).

	Type	Length	Description
-	UInt32	4	Unknown (ODA writes 2)
+	RL	4	class_version (default: 3)
+	String	2 + 2 * n + 2	App info name, ODA writes â\200\234AppInfoDataListâ\200\235
-	UInt32	4	Unknown (ODA writes 3)
-	Byte[]	16	Version data (checksum, ODA writes zeroes)
-	String	2 + 2 * n + 2	Version
-	Byte[]	16	Comment data (checksum, ODA writes zeroes)
-	String	2 + 2 * n + 2	Comment
-	Byte[]	16	Product data (checksum, ODA writes zeroes)
-	String	2 + 2 * n + 2	Product
-	String	2 + n	App info version, e.g. ODA writes "2.7.2.0".
+	RL	4	num strings (default: 3)
+	Byte[]	16	Version checksum (ODA and LibreDWG write zeroes)
+	String	2 + 2 * n + 2	Version. Eg "Teigha(R) 4.3.2.0" or AutoCAD: "19.0.55.0.0"
+	Byte[]	16	Comment checksum (ODA and LibreDWG write zeroes)
+	String	2 + 2 * n + 2	Comment. Eg "Autodesk DWG. This file is a Trusted DWG last saved by an
+			Autodesk application or Autodesk licensed application.", or "This file was last saved by an
+			Open Design Alliance (ODA) application or an ODA licensed application." or
+			"This file was last saved by LibreDWG."
+	Byte[]	16	Product checksum (ODA and LibreDWG write zeroes)
+	String	2 + 2 * n + 2	ProductInformation as XML

# 17 Data section AcDb:FileDepList

Contains file dependencies (e.g. IMAGE files, or fonts used by STYLE).

@@ -2852,11 +2913,11 @@

Name	AcDb:FileDepList
Compressed	1
Encrypted	2 (meaning unknown)
Page size	0x80 if number of entries is 0 or 1. If more than 1, then 0x80 x number of entries.

-In R18 the app info section consists of the following fields. Strings are encoded as a 32-bit length, followed by the character bytes (without trailing 0).

+In R2004 the app info section consists of the following fields. Strings are encoded as a 32-bit length, followed by the character bytes (without trailing 0).

Type	Length	Description
Int32	4	Feature count (ftc)
String32	ftc * ( 4 + n)	Feature name list. A feature name is one of the followin

g:

```

@@ -2889,21 +2950,21 @@
| Encrypted | 0
| Page size | 0x7400

```

The contents of this section are unknown. In the following paragraphs is described what the ODA writes in this section.

-## 18.1 R18

+## 18.1 R2004

Type	Length	Description
-----	-----	-----
UInt32	4	Unknown (ODA writes 0)
UInt32	4	Unknown (ODA writes 0)
UInt32	4	Unknown (ODA writes 0)

More unknown bytes may follow.

-## 18.2 R21

+## 18.2 R2007

Type	Length	Description
-----	-----	-----
UInt32	4	Unknown (ODA writes 0)
UInt32	4	Unknown (ODA writes 0)

```

@@ -2919,13 +2980,13 @@

```

Name	AcDb:Security
Compressed	1
Encrypted	0
Page size	0x7400

-This section was introduced in R18. The AcDb:Security section is optional in the file. It is present if the file was saved with a password.

+This section was introduced in R2004. The AcDb:Security section is optional in the file. It is present if the file was saved with a password.

-R18: The section is present in the file if the SecurityType entry at location 0x18 in the file is greater than 0.

+R2004: The section is present in the file if the SecurityType entry at location 0x18 in the file is greater than 0.

Strings are prefixed with a 32-bit length (not zero terminated).

Type	Length	Description
-----	-----	-----

```

@@ -2964,34 +3025,34 @@

```

This region holds the actual objects in the drawing. These can be entities, table entries, dictionary entries, and objects. This second use of objects is somewhat confusing; all items stored in the file are "objects", but only some of them are object objects. Others are entities, table entries, etc. The objects in this section can appear in any order.

Not all objects present in the file are actually used. All used objects can eventually be traced back to handle references in the Header section. So the proper way to read a file is to start reading the header and then tracing all references from there until all references have been followed. Very occasionally a file contains e.g. two APPID objects with the same name, of which one is used, and the other is not. Reading both would be incorrect due to a name clash. To complicate matters more, files also exist with table records with duplicate names. This is incorrect, and the software should rename the record to be unique upon reading.

-For R18 and later the section data (right after the page header) starts with a RL value of 0x0dca (meaning unknown).

+For R2004 and later the section data (right after the page header) starts with a RL value of 0x0dca (meaning unknown).

## ## 20.1 Common non-entity object format

Objects (non-entities) have the following general format:

Version	Field type	DXF group	Description
R2010+	MS		Size in bytes of object, not including the CRC
	MC		Size in bits of the handle stream (unsigned, 0x40 is not interpreted as sign). This includes the padding bits at the end of the handle stream (the padding bits make sure the object stream ends on a byte boundary).
-	Common		
+	Common		
R2000-R2007	OT		Object type
	RL		Size of object data in bits (number of bits before the handles), or the <code>endbit</code> of the pre-handles section.
Common:			
	H	5	Object's handle
-	BS		Size of extended object data, if any X Extended object data, if any. See EED section, chapter 28.
+	BS		Size of extended object data, if any X Extended object data, if any. See EED section, [chapter 28] (#28-extended-entity-data-extended-object-data).
R13-R14	RL		Size of object data in bits
	BL		Number of persistent reactors attached to this object
R2004+	B		If 1, no XDictionary handle is stored for this object, otherwise XDictionary handle is stored as in R2000 and earlier.
R2013+	B		Indicates whether the object has associated binary data in the data store section (see chapter 24 for more details about this section).
+	B		Indicates whether the object has associated binary data in the data store section (see [chapter 24] (#24-section-acdbacdsprototype_1b-datastorage) for more details about this section).
Common			
R2007+	X		Object data (varies by type of object)
	X		String data (optional)
	B		String stream present bit (last bit in pre-handles section).

If 1, then the `endbit` location should be decremented by 16 bytes, and a short should be read at location `endbit` (bits), call this short `strDataSize`. If this short has the 0x8000 bit set, then decrement `endbit` by an additional 16 bytes, strip the 0x8000 bit off of `strDataSize`, and read the short at this new location, calling it `hiSize`. Then set `strDataSize` to `(strDataSize | (hiSize << 15))`. `endbit` should then be decremented by this final `strDataSize` value, and this bit location marks the start of the `string stream` within this object. All unicode strings in this object are located in the `string stream`, and should be read from this stream, even though the location of the TV type fields in the object descriptions list these fields in among the normal object data.

@@ -3011,11 +3072,11 @@

Drawing entities, which are of course objects, have the same format as objects, with some additional standard items:

```

 MS : Size of object, not including the CRC
 R2010+:
 MC : Size in bits of the handle stream (unsigned, 0x40 is not interpreted as sign).
- Common:
+ Common:
 OT : Object type
 R2000+ Only:
 RL : Size of object data in bits
 Common:
 H : Object's handle
@@ -3167,10 +3228,11 @@
CELLSTYLEMAP
DBCOLOR

```

```

DICTIONARYVAR
DICTIONARYWDFLT
FIELD
+FIELDLIST
GROUP
HATCH
IDBUFFER
IMAGE
IMAGEDEF
@@ -3182,55 +3244,101 @@
MLEADER
MLEADERSTYLE
OLE2FRAME
PLACEHOLDER
PLOTSETTINGS
-RASTERVARIABLESCALE
+RASTERVARIABLES
+SCALE
 SORTENTSTABLE
 SPATIAL_FILTER
 SPATIAL_INDEX
+SUN
 TABLEGEOMETRY
 TABLESTYLES
 VBA_PROJECT
 VISUALSTYLE
 WIPEOUTVARIABLE
 XRECORD
 ``

+Todo:
+
+``
+ASSOCNETWORK
+ASSOCGEOMDEPENDENCY
+BLOCKGRIPLOCATIONCOMPONENT
+BLOCKALIGNMENTPARAMETER
+BLOCKALIGNMENTGRIP
+BLOCKBASEPOINTPARAMETER
+BLOCKFLIPACTION
+BLOCKFLIPPARAMETER
+BLOCKFLIPGRIP
+BLOCKLINEARGRIP
+BLOCKLOOKUPGRIP
+BLOCKROTATIONGRIP
+BLOCKMOVEACTION
+BLOCKROTATEACTION
+BLOCKSCALEACTION
+BLOCKVISIBILITYGRIP
+DYNAMICBLOCKPURGEPREVENTER
+MESH
+SECTIONOBJECT
+SECTION_MANAGER
+SCALE
+RENDERENVIRONMENT
+SECTION_MANAGER
+DETAILVIEWSTYLE
+SECTIONVIEWSTYLE
+PDFDEFINITION
+DGNDEFINITION
+DWFDEFINITION
+UNDERLAY
+``
+

```

For objects with non-fixed values, taking the object type minus 500 gives an index into the class list, which then determines the type of object. For instance, an object type of 501 means that this object is of the class which is second in the class list; the



```

(links are not used)
+ R2013+:
+ Has_DS_data B 1 if referring to AcDs datastore entry
+ R13+:
+ Color CMC(B) 62
+ Ltype scale BD 48
R2000+:
+ Ltype flags BB 00 = bylayer, 01 = byblock, 10 = continous,
+ 11 = linetype handle present at end of object
+ Plotstyle flags BB 00 = bylayer, 01 = byblock,
+ 11 = plotstyle handle present at end of object
R2007+:
- Material flags BB 00 = bylayer, 01 = byblock, 11 = material handle
+ Material flags BB 347 00 = bylayer, 01 = byblock, 11 = material handle
+ present at end of object
- Shadow flags RC
- Common:
- Invisibility BS 60
+ Shadow flags RC 284 0 both, 1 receives, 2 casts, 3 no
+ R2010+:
+ Has_full_visualstyle B
+ Has_face_visualstyle B
+ Has_edge_visualstyle B
+ R13+:
+ Invisibility BS 60 bit 0: 0 visible, 1 invisible
R2000+:
+ Lineweight RC 370

```

#### #### 20.4.2 Common Entity Handle Data

@@ -3841,11 +3956,11 @@

Class properties:

	App name	ObjectDBX Classes
	Class number	Dynamic (>= 500)
-	DWG version	R18
+	DWG version	R2004
	Maintenance version	0
	Class proxy flags	0x401
	C++ class name	AcDbArcDimension
	DXF name	ARC_DIMENSION

@@ -4062,11 +4177,11 @@

#### ### 20.4.25 DIMENSION (ALIGNED) (22)

...

```

Common Entity Data
- Common Dimension Data See paragraph 20.4.22.
+ Common Dimension Data See paragraph 20.4.22
Common:
+ 13-pt 3BD 13 See DXF documentation.
+ 14-pt 3BD 14 See DXF documentation.
+ 10-pt 3BD 10 See DXF documentation.
+ Ext ln rot BD 52 Extension line rotation; see DXF documentation.

```

@@ -4235,11 +4350,11 @@

Class properties:

	App name	ObjectDBX Classes
	Class number	Dynamic (>= 500)
-	DWG version	R18
+	DWG version	R2004
	Maintenance version	0
	Class proxy flags	0x401
	C++ class name	AcDbRadialDimensionLarge



| DXF name | LARGE\\_RADIAL\\_DIMENSION |

@@ -5076,21 +5191,24 @@

### 20.4.44 DICTIONARY (42)

Basically a list of pairs of string/objhandle that constitute the dictionary entries.

...

```
- Length MS -- Entity length (not counting itself or CRC).
- Type S 0 42 (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 42 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors S number of reactors in this object
+ Numreactors BL number of reactors in this object
R2004+:
 XDic Missing Flag B If 1, no XDictionary handle is stored for this
 object, otherwise XDictionary handle is stored as in
 R2000 and earlier.

Common:
@@ -5170,46 +5288,46 @@
R2000+:
 Linespacing Style BS 73
 Linespacing Factor BD 44
 Unknown bit B
R2004+:
- Background flags BL 90 0 = no background, 1 = background fill, 2 =
+ Background fill flag BL 90 0 = no background, 1 = background fill, 2 =
 background fill with drawing fill color, 0x10 = text
 frame (R2018+)

-IF background flags has bit 0x01 set, or in case of R2018 bit 0x10:
- Background scale factor
+IF Background fill flag has bit 0x01 set, or in case of R2018 bit 0x10:
+ Background fill scale factor
 BL 45 default = 1.5
- Background color CMC 63
- Background transparency
+ Background fill color CMC 63
+ Background fill transparency
 BL 441
-END IF background flags 0x01/0x10
+END IF Background fill flags 0x01/0x10
R2018+
 Is NOT annotative B
 IF MTEXT is not annotative
 Version BS Default 0
 Default flag B Default true
 BEGIN REDUNDANT FIELDS (see above for descriptions)
 Registered application H Hard pointer
- Attachment point BL
- X-axis dir 3BD 10
- Insertion point 3BD 11
+ Ignore Attachment BL
+ X-axis dir 3BD 11
+ Insertion point 3BD 10
 Rect width BD 40
```

```

Rect height BD 41
- Extents width BD 42
 Extents height BD 43
+ Extents width BD 42
END REDUNDANT FIELDS
 Column type BS 71 0 = No columns, 1 = static columns, 2 = dynamic
 columns
 IF Has Columns data (column type is not 0)
 Column height count BL 72
- Column width BD 44
+ Column width BD 44
 Gutter BD 45
 Auto height? B 73
 Flow reversed? B 74
 IF not auto height and column type is dynamic columns
-REPEAT Column heights
+REPEAT Column height count
 Column height BD 46
END REPEAT END
 IF (has column heights)
 END IF (has columns data)
 END IF (not annotative)
@@ -5238,26 +5356,26 @@
20.4.47 LEADER (45)

'''
 Common Entity Data
 Unknown bit B -- Always seems to be 0.
- Annot type BS -- Annotation type (NOT bit-coded):
+ Annot type BS 73 Annotation type (NOT bit-coded):
 Value 0 : MTEXT
 Value 1 : TOLERANCE
 Value 2 : INSERT
 Value 3 : None
- path type BS --
+ path type BS 72
 numpts BL -- number of points
 point 3BD 10 As many as counter above specifies.
 Origin 3BD -- The leader plane origin (by default it's the
first
 point).
 Extrusion 3BD 210
 x direction 3BD 211
 offsettoblockinspt 3BD 212 Used when the BLOCK option is used. Seems to be an
 unused feature.
-R14+:
- Endptproj 3BD -- A non-planar leader gives a point that projects the
+R13c3-R2007:
+ Endptproj 3BD 212 A non-planar leader gives a point that projects the
 endpoint back to the annotation. It's the offset
 from the endpoint of the leader to the annotation,
 taking into account the extrusion direction.

 R13-R14 Only:
 DIMGAP BD -- The value of DIMGAP in the associated DIMSTYLE at
@@ -5269,27 +5387,28 @@
 taller, probably by some DIMvar amount.)
 Box width BD 41 MTEXT extents width. (A text box is slightly wider,
 probably by some DIMvar amount.)
 Hooklineonxdir B hook line is on x direction if 1
 Arrowheadon B arrowhead on indicator
-R13-R14 Only:
 Arrowheadtype BS arrowhead type
+R13-R14 Only:
 Dimasz BD DIMASZ at the time of creation, multiplied by
 DIMSCALE
 Unknown B
 Unknown B

```

```

Unknown BS
Byblockcolor BS
Unknown B
Unknown B
R2000+:
- Unknown BS
 Unknown B
 Unknown B
Common:
 Common Entity Handle Data
- H 340 Associated annotation activated in R14. (hard pointer
)
+R13+:
+ H 340 Associated annotation activated in R14. (soft owner
+Common:
 H 2 DIMSTYLE (hard pointer)
 CRC X --

_20.4.47.1 Example:
@@ -5316,18 +5435,18 @@
02295 6E AB crc

```

### ### 20.4.48 MLEADER

-This entity was introduced in version 21. A significant portion (content block/text and leaders) of the multileader entity is stored in the MLeaderAnnotContext object (see paragraph 20.4.86), which is embedded into this object (stream).

+This entity was introduced in version 21. A significant portion (content block/text and leaders) of the multileader entity is stored in the MLeaderAnnotContext object (see paragraph [20.4.86] (#20486-mleaderannotcontext)), which is embedded into this object (stream).

Version	Field type	DXF group	Description
R2010+ Common	... BS	270	Common entity data. Version (expected to be 2).
-	...		MLeaderAnnotContext fields (see paragraph 20.4.86). This contains the
+	...		MLeaderAnnotContext fields (see paragraph [20.4.86] (#20486-mleaderannotcontext)). This contains the
	H	340	mleader content (block/text) and the leaders.
	BL	90	Leader style handle (hard pointer)
			Override flags:
			1 << 0 = Leader line type,
			1 << 1 = Leader line color,

@@ -5538,20 +5657,23 @@

### ### 20.4.51 BLOCK CONTROL (48)

```

 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 48 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 48 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:

```

-	<b>Numreactors</b>	<b>L</b>	<b>Number of persistent reactors attached to this obj</b>
+	<b>Numreactors</b>	<b>BL</b>	<b>Number of persistent reactors attached to this obj</b>

R2004+:  
 XDic Missing Flag      B      If 1, no XDictionaary handle is stored for this object, otherwise XDictionaary handle is stored as in R2000 and earlier.

Common:  
 @@ -5578,20 +5700,23 @@

### ### 20.4.52 BLOCK HEADER (49)

```

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 49 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 49 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors L Number of persistent reactors attached to this obj
+ Numreactors BL Number of persistent reactors attached to this obj
R2004+:
 XDic Missing Flag B If 1, no XDictionaary handle is stored for this
 object, otherwise XDictionaary handle is stored as in
 R2000 and earlier.

```

Common:  
 @@ -5663,20 +5788,23 @@

### ### 20.4.53 LAYER CONTROL (50) (UNDOCUMENTED)

```

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 50 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 50 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors L Number of persistent reactors attached to this obj
+ Numreactors BL Number of persistent reactors attached to this obj
R2004+:
 XDic Missing Flag B If 1, no XDictionaary handle is stored for this
 object, otherwise XDictionaary handle is stored as in
 R2000 and earlier.

```

Common:  
 @@ -5699,11 +5827,14 @@

### ### 20.4.54 LAYER (51)

```

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 51 (internal DWG type code).
+R2010+:

```

```

+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 51 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 code 0, length followed by the handle bytes.
 EED X -3 See EED section.
@@ -5723,17 +5854,17 @@

 an xref, otherwise this value indicates the index of
 the blockheader for the xref from which this came.
 Xdep B 70 block is dependent on an xref. (16 bit)
R13-R14 Only:
 Frozen B 70 if frozen (1 bit)
- On B if on. Normal Autodesk (and Open Design Toolkit)
+ Off B if off. Normal Autodesk (and Open Design Toolkit)
 policy is not to report this per se, but rather to
 negate the color if the layer is off.
 Frz in new B 70 if frozen by default in new viewports (2 bit)
 Locked B 70 if locked (4 bit)
R2000+:
- Values BS 70,290,370 contains frozen (1 bit), on (2 bit), frozen
+ Values BS 70,290,370 contains frozen (1 bit), off (2 bit), frozen
 by default in new viewports (4 bit), locked (8 bit),
 plotting flag (16 bit), and lineweight (mask with
 0x03E0)

Common:
 Color CMC 62
@@ -5766,20 +5897,23 @@

20.4.55 SHAPEFILE CONTROL (52) (UNDOCUMENTED)

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 52 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 52 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors L Number of persistent reactors attached to this obj
+ Numreactors BL Number of persistent reactors attached to this obj
R2004+:
 XDic Missing Flag B If 1, no XDictionary handle is stored for this
 object, otherwise XDictionary handle is stored as in
 R2000 and earlier.

Common:
@@ -5800,11 +5934,11 @@
024D5 33 8B crc
'''

20.4.56 SHAPEFILE (53)

```

-This contains a text style for the TEXT or MTEXT entity. Mostly the font information is stored in fields Font name and Big font name, but sometimes (for reasons unknown) some true type font information is contained in the table recordâ\200\231s extended data (see paragraph 28). The true type descriptor is stored as follows in the extended data:

+This contains a text style for the TEXT or MTEXT entity. Mostly the font information is stored in fields Font name and Big font name, but sometimes (for reasons unknown) some true type font information is contained in the table recordâ\200\231s extended data (

see paragraph [28] (#28-extended-entity-data-extended-object-data)). The true type descriptor is stored as follows in the extended data:

Group code (Value type)	Value
1001 (String)	Font file name
1002 (Bracket)	â\200\230{â\200\230 (optional)
@@ -5816,11 +5950,14 @@	
	Character set (bitmask) = 0x0000ff00
1002 (Bracket)	â\200\230}â\200\231 (optional)

```

...
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 53 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 53 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 code 0, length followed by the handle bytes.
 EED X -3 See EED section.
@@ -5871,20 +6008,23 @@

```

### 20.4.57 LINETYPE CONTROL (56) (UNDOCUMENTED)

```

...
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 56 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 56 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors L Number of persistent reactors attached to this obj
+ Numreactors BL Number of persistent reactors attached to this obj
R2004+:
 XDic Missing Flag B If 1, no XDictionaary handle is stored for this
 object, otherwise XDictionaary handle is stored as in
 R2000 and earlier.

Common:
@@ -5911,11 +6051,14 @@

```

### 20.4.58 LTYPE (57)

```

...
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 57 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 57 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 code 0, length followed by the handle bytes.
 EED X -3 See EED section.
@@ -5947,23 +6090,23 @@
 X-offset RD 44 (0.0 for a simple dash.)

```

```

Y-offset RD 45 (0.0 for a simple dash.)
Scale BD 46 (1.0 for a simple dash.)
Rotation BD 50 (0.0 for a simple dash.)
Shapeflag BS 74 bit coded:
- if (shapeflag & 1), text is rotated 0 degrees,
- otherwise it follows the segment
- if (shapeflag & 2), complexshapecode holds the
+ if (shapeflag & 1), text/shape is rotated absolutely
by Rotation,
+ otherwise Rotation follows the segment
+ if (shapeflag & 2), complex shapecode holds the
 index of the shape to be drawn
- if (shapeflag & 4), complexshapecode holds the index
+ if (shapeflag & 4), complex shapecode holds the index
 into the text area of the string to be drawn.
'''

```

NOTE: Teigha Classic for .dwg files Toolkit does not present the data this way. It use a separate variable called stroffset which indicates the offsets into the text string area. This is done in order to attempt to make the data easier to understand.

```

'''
}

```

**-R2004 and earlier:**

**+R13-R2004:**

```

Strings area X 9 256 bytes of text area. The complex dashes that
 have text use this area via the 75-group indices.
 It's basically a pile of 0-terminated strings. First
 byte is always 0 for R13 and data starts at byte 1.
 In R14 it is not a valid data start from byte 0.

```

*@@ -5982,20 +6125,23 @@*

### 20.4.59 VIEW CONTROL (60) (UNDOCUMENTED)

```

'''

```

```

Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 60 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 60 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors L Number of persistent reactors attached to this obj
+ Numreactors BL Number of persistent reactors attached to this obj
R2004+:
 XDic Missing Flag B If 1, no XDictionary handle is stored for this
 object, otherwise XDictionary handle is stored as in
 R2000 and earlier.

```

Common:

*@@ -6018,11 +6164,14 @@*

### 20.4.60 VIEW (61)

```

'''

```

```

Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 61 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 61 (internal DWG type code).

```

```

R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 code 0, length followed by the handle bytes.
 EED X -3 See EED section.
@@ -6116,20 +6265,23 @@

20.4.61 UCS CONTROL (62) (UNDOCUMENTED)

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 62 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 62 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors L Number of persistent reactors attached to this obj
+ Numreactors BL Number of persistent reactors attached to this obj
R2004+:
 XDic Missing Flag B If 1, no XDictionary handle is stored for this
 object, otherwise XDictionary handle is stored as in
 R2000 and earlier.

Common:
@@ -6152,11 +6304,14 @@

20.4.62 UCS (63)

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 63 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 63 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 code 0, length followed by the handle bytes.
 EED X -3 See EED section.
@@ -6214,11 +6369,14 @@

20.4.63 TABLE (VPORT) (64) (UNDOCUMENTED)

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 64 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 64 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 code 0, length followed by the handle bytes.
 EED X -3 See EED section.
@@ -6252,11 +6410,14 @@

20.4.64 VPORT (65)

```



```

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 65 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 65 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
@@ -6380,11 +6541,14 @@

```

### 20.4.65 TABLE (APPID) (66) (UNDOCUMENTED)

```

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 66 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 66 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
@@ -6416,11 +6580,14 @@

```

### 20.4.66 APPID (67)

```

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 67 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 67 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
@@ -6463,11 +6630,14 @@

```

### 20.4.67 DIMSTYLE CONTROL (68) (UNDOCUMENTED)

```

'''
 Length MS -- Object length (not counting itself or CRC).
- Type BS 0&2 68 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 68 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
@@ -6498,11 +6668,11 @@
'''

```

### 20.4.68 DIMSTYLE (69)

```

'''
- Length MS -- Entity length (not counting itself or CRC).

```

```

+ Length MS -- Object length (not counting itself or CRC).
 Type BS 0 69 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
@@ -6646,11 +6816,11 @@
 DIMUPT B 288
 DIMFIT BS 287
R2007+:
 DIMFXLON B 290
R2010+:
- DIMTXTDIRECTION B 295
+ DIMTXTDIRECTION B 295 (or 294?)
 DIMALTMZF BD ?
 DIMALTMZS T ?
 DIMMZSF BD ?
 DIMMZS T ?
R2000+:
@@ -6699,21 +6869,24 @@
 \ \ \

```

### ### 20.4.69 VIEWPORT ENTITY CONTROL (70) (UNDOCUMENTED)

```

 \ \ \
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0&2 70 (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 70 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Owner handle (soft pointer) of root object (0).
 EED X -3 See EED section.
R13-R14 Only:
 Obj size RL size of object in bits, not including end handles
Common:
- Numreactors B L Number of persistent reactors attached to this obj
+ Numreactors BL Number of persistent reactors attached to this obj
R2004+:
 XDic Missing Flag B If 1, no XDictionary handle is stored for this
 object, otherwise XDictionary handle is stored as in
 R2000 and earlier.
Common:
@@ -6736,12 +6909,15 @@
 \ \ \

```

### ### 20.4.70 VIEWPORT ENTITY HEADER (71)

```

 \ \ \
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0&2 71 (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 71 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
@@ -6786,22 +6962,25 @@
03587 2F 9E crc

```

\*\*\*

### ### 20.4.71 AcDbAnnotScaleObjectContextData

-This class inherits from class AcDbObjectContextData (see paragraph 20.4.89).  
+This class inherits from class AcDbObjectContextData (see paragraph [20.4.89] (#20489-acdbobjectcontextdata)).

	Version	Field type	DXF group	Description
-	...	H	340	Common AcDbObjectContextData data (see paragraph 20.4.89). Handle to scale (AcDbScale) object (hard pointer). See paragraph 20.4.92.
+	...			Common AcDbObjectContextData data (see paragraph [20.4.89] (#20489-acdbobjectcontextdata)).
+	H	340		Handle to scale (AcDbScale) object (hard pointer). See paragraph [20.4.92] (#20492-scale-acdbscale).

### ### 20.4.72 GROUP (72): Group of ACAD entities

\*\*\*

-	Length	MS	--	Entity length (not counting itself or CRC).
-	Type	BS	0	72 (internal DWG type code).
+	Length	MS	--	Object length (not counting itself or CRC).

+R2010+:

+	Handle Stream Size	MC	--	not counted in the Length
---	--------------------	----	----	---------------------------

+Common:

+	Type	OT	0	72 (internal DWG type code).
---	------	----	---	------------------------------

R2000+:

	Obj size	RL		size of object in bits, not including end handles
--	----------	----	--	---------------------------------------------------

Common:

	Handle	H	5	Length (char) followed by the handle bytes.
	EED	X	-3	See EED section.

@@ -6838,12 +7017,15 @@

\*\*\*

### ### 20.4.73 MLINestyle (73):

\*\*\*

-	Length	MS	--	Entity length (not counting itself or CRC).
-	Type	BS	0	73 (internal DWG type code).
+	Length	MS	--	Object length (not counting itself or CRC).

+R2010+:

+	Handle Stream Size	MC	--	not counted in the Length
---	--------------------	----	----	---------------------------

+Common:

+	Type	OT	0	73 (internal DWG type code).
---	------	----	---	------------------------------

R2000+:

	Obj size	RL		size of object in bits, not including end handles
--	----------	----	--	---------------------------------------------------

Common:

	Handle	H	5	Length (char) followed by the handle bytes.
	EED	X	-3	See EED section.

@@ -6912,12 +7094,15 @@

NOTE: OBJECTS LISTED AFTER THIS POINT DO NOT HAVE FIXED TYPES. THEIR TYPES ARE DETERMINED BY FINDING THE CLASS ENTRY WHOSE POSITION IN THE CLASS LIST + 500 EQUALS THE TYPE OF THIS OBJECT

### ### 20.4.74 DICTIONARYVAR (varies)

\*\*\*

-	Length	MS	--	Entity length (not counting itself or CRC).
-	Type	BS	0	72 (internal DWG type code).
+	Length	MS	--	Object length (not counting itself or CRC).

+R2010+:

+	Handle Stream Size	MC	--	not counted in the Length
---	--------------------	----	----	---------------------------

+Common:

+	Type	OT	0	72 (internal DWG type code).
---	------	----	---	------------------------------

R2000+:

Obj size	RL	size of object in bits, not including end handles
Common:		
Handle	H 5	Length (char) followed by the handle bytes.
EED	X -3	See EED section.

```

@@ -7015,11 +7200,11 @@
 pt0 2RD 10 control point
 if (isrational)
 weight BD 40 weight
 endif
End repeat

```

```
-R24:
```

```

+R2010:
 Numfitpoints BL 97 number of fit points
 Begin repeat numfitpoints times:
 Fitpoint 2RD 11
 End repeat
 Start tangent 2RD 12

```

```
@@ -7125,21 +7310,21 @@
```

```
Class properties:
```

	App name	ObjectDBX Classes
	Class number	Dynamic (>= 500)
-	DWG version	R18
+	DWG version	R2004
	Maintenance version	0
	Class proxy flags	0x480
	C++ class name	AcDbField
	DXF name	FIELD

-Fields are referenced from the field list of a drawing (paragraph 20.4.77).

+Fields are referenced from the field list of a drawing (paragraph [20.4.77] (#20477-fieldlist)).

	Version	Field type	DXF group	Description
	---	---	---	
-	...	...		Common object data (paragraph 20.1).
+	...	...		Common object data (paragraph [20.1] (#201-common-non-entity-object-format)).
	TV	1		Evaluator ID TV 2,3 Field code (in DXF strings longer than 255 characters
or				are written in chunks of 255 characters in one 2 group and one
	BL	90		more 3 groups). Number of child fields Begin repeat child fields
				Invalid code = 16, Invalid context = 32, Other error = 64
	BL	96		Evaluation error code
	TV	300		Evaluation error message
-	...	...		The field value, see paragraph 20.4.99.
+	...	...		The field value, see paragraph [20.4.99] (#20499-value).
	TV	301,9		Value string (DXF: written in 255 character chunks)
-	TV	98		Value string length
+	BL	98		Value string length
	BL	93		Number of child fields Begin repeat child fields
	TV	6		Child field key
-	...	...		The field value, see paragraph 20.4.99.
+	...	...		The field value, see paragraph [20.4.99] (#20499-value).
				End repeat child fields

```
20.4.77 FIELDLIST
```

```
Class properties:
```

App name	ObjectDBX Classes
Class number	Dynamic ( $\geq 500$ )
- DWG version	R18
+ DWG version	R2004
Maintenance version	0
Class proxy flags	0x480
C++ class name	AcDbFieldList, inherits AcDbIdSet
DXF name	FIELDLIST

-Fields (paragraph 20.4.76) are referenced from the field list of a drawing. The field list is stored in the root dictionary entry ACAD\_FIELDLIST.

+Fields (paragraph [20.4.76](#20476-field)) are referenced from the field list of a drawing. The field list is stored in the root dictionary entry ACAD\_FIELDLIST.

Version	Field type	DXF group	Description
---	---	---	---
-	...		Common object data (paragraph 20.1).
+	...		Common object data (paragraph [20.1](#201-common-non-entity-object-format))).
	BL		Number of fields
	B		Unknown
			Begin repeat fields
	H	330	Field handle (soft pointer)
			End repeat fields

@@ -7216,21 +7401,21 @@

Class properties:

App name	ObjectDBX Classes
Class number	Dynamic ( $\geq 500$ )
- DWG version	R21
+ DWG version	R2007
Maintenance version	45
Class proxy flags	0xFFFF
C++ class name	AcDbGeoData
DXF name	GEODATA

The geo data object was introduced in AutoCAD 2009. The format changed considerably in AutoCAD 2010. The objectVersion field discerns between the formats (1 = AutoCAD 2009, 2 = AutoCAD 2010, 3 = AutoCAD 2013, but the format is the same as 2010).

Version	Field type	DXF group	Description
---	---	---	---
-	...		Common object data (paragraph 20.1).
+	...		Common object data (paragraph [20.1](#201-common-non-entity-object-format))).
	BL		Object version formats (1 = AutoCAD 2009, 2 = AutoCAD 2010, 3 = AutoCAD 2013, but the format is the same as 2010)
	H		Soft pointer to host block (model space layout owner block)
	BS		Design coordinate type (0 = unknown, local grid = 1, projected grid = 2, geographic (defined by latitude/longitude)

= 3)

@@ -7246,11 +7431,11 @@

	BD		Light years = 19, Parsecs = 20
	BL		Unit scale factor vertical
			Units value vertical (same enumeration as for the units value horizontal)
	3BD		Up direction
-	3RD		North direction
+	2RD		North direction
	BL		Scale estimation method: None = 1, User specified scale factor
= 2,			
	BD		Grid scale at reference point = 3, Prismsodial = 4
	B		User specified scale factor
			Do sea level correction

	BD	Sea level elevation
@@ -7284,11 +7469,11 @@		
		Repeat for each geo mesh face
	BL	Face index 1
	BL	Face index 2
	BL	Face index 3
		End repeat geo mesh faces
-		If DWG version is R21 or lower:
+		If DWG version is R2007 or lower:
		Below is CIVIL data. AutoCAD 2010 always writes civil data.
	B	Has civil data? (true)
	B	False
	RD	Reference point Y
	RD	Reference point X
@@ -7311,12 +7496,15 @@		

### 20.4.79 IDBUFFER (varies)

(holds list of references to an xref)

```

'''
- Length MS -- Entity length (not counting itself or CRC).
- Type S 0 (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
@@ -7431,12 +7619,15 @@

```

### 20.4.81 IMAGEDEF (varies)

```

'''
(used in conjunction with IMAGE entities)
- Length MS -- Entity length (not counting itself or CRC).
- Type S 0 (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
@@ -7481,12 +7672,15 @@

```

### 20.4.82 IMAGEDEFREACTOR (varies)

```

'''
(used in conjunction with IMAGE entities)
- Length MS -- Entity length (not counting itself or CRC).
- Type S 0 (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.

```

```

EED X -3 See EED section.
@@ -7517,12 +7711,15 @@
'''

20.4.83 LAYER_INDEX

'''
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 (internal DWG type code).
R2000+:
Obj size RL size of object in bits, not including end handles
Common:
Handle H 5 Length (char) followed by the handle bytes.
EED X -3 See EED section.
@@ -7533,12 +7730,12 @@
R2004+:
XDic Missing Flag B If 1, no XDictionary handle is stored for this
object, otherwise XDictionary handle is stored as in
R2000 and earlier.
Common:
- timestamp1 BL 40
- timestamp2 BL 40
+ timestamp1 BL 40 last_updated days
+ timestamp2 BL 40 last_updated msec
numentries BL the number of entries
Repeat numentries times:
Indexlong BL a long
Indexstr TV 8 a layer name
End repeat
@@ -7572,11 +7769,14 @@

20.4.84 LAYOUT (varies)

'''
Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 (internal DWG type code).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 (internal DWG type code).
R2000+:
Obj size RL size of object in bits, not including end handles
Common:
Handle H 5 Length (char) followed by the handle bytes.
EED X -3 See EED section.
@@ -7738,17 +7938,17 @@
03F1C 85 93 crc
'''

20.4.86 MLeaderAnnotContext

```

-This is a helper class for the multileader entity (see paragraph 20.4.48), that inherits from class AcDbAnnotScaleObjectContextData (see paragraph 20.4.71).

+This is a helper class for the multileader entity (see paragraph [20.4.48] (#20448-mleader)), that inherits from class AcDbAnnotScaleObjectContextData (see paragraph [20.4.71] (#20471-acdbannotscaleobjectcontextdata)).

This object mainly contains a content object, which is either a block or multiline text. To the content object one or two leader roots are attached. They are either attached to the left/right or top/bottom depending on the multileaders attachment direction (horizontal/vertical). Each leader root can contain one more leader lines.

Version	Field type	DXF group	Description
----	----	----	-----
-	...	:	Common AcDbAnnotScaleObjectContextData data (see paragraph 20.4.71).
+	...	:	Common AcDbAnnotScaleObjectContextData data (see paragraph [20.4.71] (#20471-acdbannotscaleobjectcontextdata)).
	BL	300	DXF: â\200\234CONTEXT_DATA{â\200\234   Number of leader roots   Begin repeat leader root
	B	302	DXF: â\200\234LEADER{â\200\234
	B	290	Is content valid (ODA writes true)
@@ -7860,17 +8060,17 @@			
	BS	272	Style bottom attachment. See also MLEADER style left text attachment type for values. Relevant if mleader attachment direction is vertical.
	-	301	DXF: â\200\234}â\200\235

### 20.4.87 MLEADERSTYLE (AcDbMLeaderStyle)

-This class inherits from AcDbObject. The provides a style for the MLEADER entity (see paragraph 20.4.48).

+This class inherits from AcDbObject. The provides a style for the MLEADER entity (see paragraph [20.4.48] (#20448-mleader)).

The value of IsNewFormat is true in case the version is R2010 or later, or if the object has extended data for APPID â\200\234ACAD\_MLEADERVERâ\200\235.

Version	Field type	DXF group	Description
----	----	----	-----
-	...	:	Common AcDbAnnotScaleObjectContextData data (see paragraph 20.1).
+	...	:	Common AcDbAnnotScaleObjectContextData data (see paragraph [20.1] (#201-common-non-entity-object-format)).
R2010			
Common	BS	179	Version (expected to have value 2)
	BS	170	Content type (see paragraph on LEADER for more details).
	BS	171	Draw multi-leader order (0 = draw content first, 1 = draw leader first)
@@ -7947,20 +8147,23 @@			

This class inherits from AcDbObject. The object provides contextual data for another object/entity.

Version	Field type	DXF group	Description
----	----	----	-----
-	...	:	Common object data (paragraph 20.1).
+	...	:	Common object data (paragraph [20.1] (#201-common-non-entity-object-format)).
R2010	BS	70	Version (default value is 3).
	B	-	Has file to extension dictionary (default value is true).
	B	290	Default flag (default value is false).

### 20.4.90 PROXY (varies):

```

'''
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 typecode (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.

```



@@ -7995,12 +8198,15 @@

### 20.4.91 RASTERVARIABLES (varies)

...

(used in conjunction with IMAGE entities)

```
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 typecode (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
```

@@ -8037,22 +8243,25 @@

This class inherits from AcDbObject. This represents a ratio of paper units to drawing units, where the drawing units are divided by 10 when using the same distance units (e.g. mm). E.g. a scale of 1 mm to 10 mm is stored as paper units = 1, drawing units = 1. A scale of 1 mm to 1000 mm (= 1 m) is stored as paper units = 1, drawing units = 100.

Version	Field type	DXF group	Description
-	...	:	Common object data (paragraph 20.1).
+	...	:	Common object data (paragraph [20.1] (#201-common-non-entity-object-format)).
	BS	70	Unknown (ODA writes 0).
	TV	300	Name
	BD	140	Paper units (numerator)
	BD	141	Drawing units (denominator, divided by 10).
	B	290	Has unit scale

### 20.4.93 SORTENTISTABLE (varies)

...

```
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 typecode (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
```

@@ -8105,12 +8314,15 @@

### 20.4.94 SPATIAL\_FILTER (varies)

...

(used to clip external references)

```
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 typecode (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
```

```

 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
@@ -8169,12 +8381,15 @@
'''

20.4.95 SPATIAL_INDEX (varies):

'''
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 typecode (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
@@ -8186,12 +8401,12 @@
 XDic Missing Flag B If 1, no XDictionary handle is stored for this
 object, otherwise XDictionary handle is stored as in
 R2000 and earlier.

Common:
- timestamp1 BL
- timestamp2 BL
+ timestamp1 BL 40 last_updated days
+ timestamp2 BL 40 last_updated msec
 unknown X rest of bits to handles
 Handle refs H parenthandle (hard owner)
 [Reactors (soft pointer)]
 xdictionary (hard owner)
'''
@@ -8333,18 +8548,18 @@
0D688 54 B0 crc
'''

```

### ### 20.4.96 TABLE (varies)

-The TABLE entity (entity type ACAD\_TABLE) was introduced in AutoCAD 2005 (a sub release of R18), and a large number of changes were introduced in AutoCAD 2008 (a sub release of R21). The table entity inherits from the INSERT entity. The geometric results, consisting of table borders, texts and such are created in an anonymous block, similarly to the mechanism in the DIMENSION entity.

-The anonymous block name prefix is â\200\234\*Tâ\200\235. For the AutoCAD 2008 changes see paragraph 20.4.96.2.

+The TABLE entity (entity type ACAD\_TABLE) was introduced in AutoCAD 2005 (a sub release of R2004), and a large number of changes were introduced in AutoCAD 2008 (a sub release of R2007). The table entity inherits from the INSERT entity. The geometric results, consisting of table borders, texts and such are created in an anonymous block, similarly to the mechanism in the DIMENSION entity.

+The anonymous block name prefix is â\200\234\*Tâ\200\235. For the AutoCAD 2008 changes see paragraph [20.4.96.2.] (#20.4.96.2.)

TODO: document roundtrip data with connections to AcDbTableContent and AcDbTableGeometry.

-20.4.96.1 \*\*\_Until R21\_\*\*

+20.4.96.1 \*\*\_Until R2007\_\*\*

-This paragraph describes the table DWG format until R21. In R24 the format was changed to make use of table content to contain all data (AcDbTableContent).

+This paragraph describes the table DWG format until R2007. In R2010 the format was changed to make use of table content to contain all data (AcDbTableContent).

```

'''
 Common Entity Data
 Ins pt 3BD 10
R13-R14 Only:
@@ -8469,11 +8684,11 @@

 Left visibility BS 288 override
 Present only if bit 0x20000 is set in cell flag
 override (1 = visible).

R2007+:
 Unknown BL
- Value fields ... See paragraph 20.4.98.
+ Value fields ... See paragraph [20.4.98] (#20498-cell-content-geometry)
.
Common:
End Cell Data (remaining data applies to entire table)
 Has table overrides B
If has table overrides == 1:
 Table flag override BL 93
@@ -8618,13 +8833,13 @@

 0x80000 is set in table overrides flag

CRC X ---
'''

```

**\*\*-20.4.96.2\*\* \*\*\_R24 and later\_\*\***  
**\*\*\*20.4.96.2\*\* \*\*\_R2010 and later\_\*\***

-In the R24 format the old table data structures were replaced with new data structures, of which the root is the `AcDbTableContent` class. The old data structures are still used in the DXF format. An R24 DXF file contains both the old and new structures, where the new structures are optionally used. If AutoCAD can store all data just using the old structures it does not always write the new structures in DXF. In an R24 DWG file, always the new structures are used. The table then points to a `AcDbTableContent` object, which contains most of the actual data. Note that `AcDbTableContent` was already introduced in AutoCAD 2008 (R21), but in R21 it was indirectly referenced through the tables extension dictionary entry '`ACAD_XREC_ROUNDTRIP`' (TODO: describe details on '`ACAD_ROUNDTRIP_2008_TABLE_ENTITY`' and for 2007).

+In the R2010 format the old table data structures were replaced with new data structures, of which the root is the `AcDbTableContent` class. The old data structures are still used in the DXF format. An R2010 DXF file contains both the old and new structures, where the new structures are optionally used. If AutoCAD can store all data just using the old structures it does not always write the new structures in DXF. In an R2010 DWG file, always the new structures are used. The table then points to a `AcDbTableContent` object, which contains most of the actual data. Note that `AcDbTableContent` was already introduced in AutoCAD 2008 (R2007), but in R2007 it was indirectly referenced through the tables extension dictionary entry '`ACAD_XREC_ROUNDTRIP`' (TODO: describe details on '`ACAD_ROUNDTRIP_2008_TABLE_ENTITY`' and for 2007).

Version	Field type	DXF group	Description
R2010+	RC	11	Common entity data
R2010	B	11	Unknown (default 0)
R2013	BL	11	Unknown (default true)
R2010+	...	11	Unknown (default 0)
without the	...	11	Here the table content is present (see <code>TABLECONTENT</code> object), without the
-	BS	11	common OBJECT data. See paragraph 20.4.97.
+	3BD	11	common OBJECT data. See paragraph [20.4.97.] (#20.4.97.)
	BL	11	Unknown (default 38)
	BL	11	Horizontal direction
	BL	11	Has break data flag (0 = no break data, 1 = has break data)
	BL	11	Begin break data (optional)
	BL	11	Option flags:

@@ -8667,11 +8882,11 @@

	BL	End row index
		End repeat row ranges

### ### 20.4.97 TABLECONTENT

-This represents the table content (AcDbTableContent) that replaces the old table data structures that were introduced in AutoCAD 2005. Table content was introduced in AutoCAD 2008 and supports more advanced features like e.g. multiple contents per cell. In AutoCAD 2008 the table content was written as a separate object in DWG and referenced by roundtrip data in the table entity's extension dictionary. In DXF this is still the case even for R24. In a R24 DWG file, the table content is part of the table entity data and is no longer present as a separate object. Possibly for backwards compatibility with the AutoCAD 2007 (R21) format, this separate data container was created instead of extending the ACAD\\_TABLE entity.

+This represents the table content (AcDbTableContent) that replaces the old table data structures that were introduced in AutoCAD 2005. Table content was introduced in AutoCAD 2008 and supports more advanced features like e.g. multiple contents per cell. In AutoCAD 2008 the table content was written as a separate object in DWG and referenced by roundtrip data in the table entity's extension dictionary. In DXF this is still the case even for R2010. In a R2010 DWG file, the table content is part of the table entity data and is no longer present as a separate object. Possibly for backwards compatibility with the AutoCAD 2007 format, this separate data container was created instead of extending the ACAD\\_TABLE entity.

The table content class inherits from 3 other classes, which never exist independently so they will all be described in this paragraph. AcDbTableContent inherits from AcDbFormattedTableData, which inherits from AcDbLinkedTableData, which inherits from AcDbLinkData. Class AcDbLinkedTableData contains most of the data (rows, columns, cells, cell contents).

Version	Field type	DXF group	Description
@@ -8681,12 +8896,12 @@			
	TV	300	Description AcDbLinkedTableData fields
	BL	90	Number of columns
			Begin repeat columns
	TV	300	Column name
	BL	91	32 bit integer containing custom data
-	...		Custom data collection, see paragraph 20.4.100.
-	...		Cell style data, see paragraph 20.4.101.4, this contains cell style overrides for
+	...		Custom data collection, see paragraph [20.4.100.](#20.4.100.)
+	...		Cell style data, see paragraph [20.4.101.4](#20.4.101.4), this contains cell style overrides for
			the column.
	BL	90	Cell style ID, points to the cell style in the table's extension dictionary that is used as the
			base cell style for the column. 0 if not present.
	BD	40	Column width.
			End repeat columns
@@ -8702,11 +8917,11 @@			
			Format locked = 0x10,
			Format readonly = 0x20,
			Format modified after update = 0x40
	TV	300	Tooltip
	BL	91	32 bit integer containing custom data
-	...		Custom data collection, see paragraph 20.4.100.
+	...		Custom data collection, see paragraph [20.4.100.](#20.4.100.)
	BL	92	Has linked data flags, 0 = false, 1 = true If has linked data
	H	340	Handle to data link object (hard pointer).
	BL	93	Row count.
	BL	94	Column count.
	BL	96	Unknown.
@@ -8717,11 +8932,11 @@			

			Unknown = 0,
			Value = 0x1,
			Field = 0x2,
			Block = 0x4
			If cell content type is Value
-	...		<b>Write value (see paragraph 20.4.98)  </b>
+	...		<b>Write value (see paragraph [20.4.98] (#20498-cell-content-geome</b>
try))			
	H	340	Else if cell content type is Field
			Handle to AcDbField object (hard pointer).
	H	340	Else if cell content type is Block
			Handle to block record (hard pointer).
			End if cell content type is Block BL 91 Number of attributes
@@ -8730,15 +8945,15 @@			
	TV	301	Attribute value.
	BL	92	Index (starts at 1).
			End repeat attributes
	BS	170	Has content format overrides flag
			If has content format overrides flag is non-zero
-	...		<b>The content format overrides, see paragraph 20.4.101.3. By def</b>
ault the cell			
+	...		<b>The content format overrides, see paragraph [20.4.101.3] (#2041</b>
013-content-format). By default the cell			
			content uses the cellâ\200\231s cell style, this allows to ove
			rride properties per content.
			End if has content format overrides flag is non-zero
			End repeat cell contents
-	...		<b>Cell style data, see paragraph 20.4.101.4, this contains cell</b>
style overrides for			
+	...		<b>Cell style data, see paragraph [20.4.101.4] (#2041014-cell-styl</b>
e), this contains cell style overrides for			
			the cell.
	BL	90	Cell style ID, points to the cell style in the tableâ\200\231s
			table style that is used as the
			base cell style for the cell. 0 if not present.
	BL	91	Unknown flag
			If unknown flag is non-zero
@@ -8746,28 +8961,28 @@			
	BD	40	Unknown
	BD	41	Unknown
	BL		Geometry data flags
	H		Unknown ()
			If geometry data flags is non-zero
-	...		<b>Cell content geometry, see paragraph 20.4.98.  </b>
+	...		<b>Cell content geometry, see paragraph [20.4.98] (#20498-cell-con</b>
tent-geometry).			
			Enf if geometry data flags is non-zero
			End If unknown flag is non-zero
			End repeat cells
	BL	91	32 bit integer containing custom data
-	...		<b>Custom data collection, see paragraph 20.4.100.  </b>
-	...		<b>Cell style data, see paragraph 20.4.101.4, this contains cell</b>
style overrides for the row.			
+	...		<b>Custom data collection, see paragraph [20.4.100.] (#20.4.100.)</b>
+	...		<b>Cell style data, see paragraph [20.4.101.4] (#20.4.101.4), this</b>
contains cell style overrides for the row.			
	BL	90	Cell style ID, points to the cell style in the tableâ\200\231s
			table style that is used as the
			base cell style for the row. 0 if not present.
	BD	40	Row height.
			End repeat rows.
	BL		Number of cell contents that contain a field reference.
			Begin repeat field references
	H		Handle to field (AcDbField), hard owner.
			End repeat field references
			**AcDbFormattedTableData** fields

```

-| | ... | | The tableâ\200\231s cell style override fields (see paragraph
20.4.101.4). The tableâ\200\231s |
-| | | | base cell style is the table styleâ\200\231s overall cell styl
e (present from R24 onwards). |
+| | ... | | The tableâ\200\231s cell style override fields (see paragraph
[20.4.101.4] (#20.4.101.4)). The tableâ\200\231s |
+| | | | base cell style is the table styleâ\200\231s overall cell styl
e (present from R2010 onwards). |
| | BL | 90 | Number of merged cell ranges |
| | | | Begin repeat merged cell ranges |
| | BL | 91 | Top row index |
| | BL | 92 | Left column index |
| | BL | 93 | Bottom row index |
@@ -8827,21 +9042,21 @@
| Version | Field type | DXF group | Description |
|-----|-----|-----|-----|
| | BL | 90 | Number of custom data items |
| | | | Begin repeat custom data items |
| | TV | 300 | Item name |
-| | ... | | Item value (variant), see paragraph 20.4.98. |
+| | ... | | Item value (variant), see paragraph [20.4.98] (#20498-cell-cont
ent-geometry). |
| | | | End repeat custom data items |

```

### 20.4.101 TABLESTYLE

-The table style object represents the style for the table entity. Like the table entity, table style was introduced in AutoCAD 2005. In AutoCAD 2008 new cell style data was introduced, which was stored in a separate container object: CELLSTYLEMAP, see paragraph 20.4.102 for more details. The cellstyle map can contain custom cell styles, whereas the TABLESTYLE only contains the Table (R24), \_Title, \_Header and \_Data cell style.

+The table style object represents the style for the table entity. Like the table entity, table style was introduced in AutoCAD 2005. In AutoCAD 2008 new cell style data was introduced, which was stored in a separate container object: CELLSTYLEMAP, see paragraph [20.4.102] (#204102-cellstylemap) for more details. The cellstyle map can contain custom cell styles, whereas the TABLESTYLE only contains the Table (R2010), \_Title, \_Header and \_Data cell style.

### 20.4.101.1 \_TABLESTYLE format until R21\_

```

'''
- Common OBJECT data, see paragraph 20.1.
+ Common OBJECT data, see paragraph [20.1] (#201-common-non-entity-object-format).
Common:
Description TV 3
Flow direction BS 70 0 = down, 1 = up
Bit flags BS 71 Meaning unknown.
Hori. cell margin BD 40
@@ -8869,30 +9084,30 @@
Data unit type BL 91 As defined in the ACAD_TABLE entity.
Format string TV 1
End repeat row styles
'''

```

##### 20.4.101.2 R24 TABLESTYLE format

+##### 20.4.101.2 R2010 TABLESTYLE format

```

| Version | Field type | DXF group | Description |
|-----|-----|-----|-----|
| | RC | - | Unknown |
| | TV | 3 | Description |
| | BL | - | Unknown |
| | BL | - | Unknown |
| | H | - | Unknown (hard owner) |
-| | ... | | The cell style with name â\200\234Tableâ\200\235, see paragraph
h 20.4.101.4. |
-| | BL | 90 | Cell style ID, 1 = title, 2 = header, 3 = data, 4 = table (new

```

```

in R24). |
+| ... | The cell style with name â\200\234Tableâ\200\235, see paragrap
h [20.4.101.4] (#2041014-cell-style). |
+| BL | 90 | Cell style ID, 1 = title, 2 = header, 3 = data, 4 = table (new
in R2010). |
| | | | The cell style ID is used by cells, columns, rows to reference
a cell style in the |
| | | | tableâ\200\231s table style. Custom cell style IDâ\200\231s ar
e numbered starting at 101. |
| BL | 91 | Cell style class, 1= data, 2 = label. The default value is lab
el. |
| | | | Cell style name
| TV | 300 | The number of cell styles (should be 3), the non-custom cell s
| BL | | | styles are present
| | | | only in the CELLSTYLEMAP.
| | | | Begin repeat cell styles (for data, title, header in this orde
r) |
-| ... | The cell style fields, see paragraph 20.4.101.4. |
-| BL | - | Cell style ID, 1 = title, 2 = header, 3 = data, 4 = table (new
in R24). |
+| ... | The cell style fields, see paragraph [20.4.101.4] (#2041014-cel
l-style). |
+| BL | - | Cell style ID, 1 = title, 2 = header, 3 = data, 4 = table (new
in R2010). |
| | | | The cell style ID is used by cells, columns, rows to reference
a cell style in the |
| | | | tableâ\200\231s table style. Custom cell style IDâ\200\231s ar
e numbered starting at 101. |
| BL | - | Cell style class, 1= data, 2 = label. The default value is lab
el. |
| | | | Cell style name
| TV | - | End repeat cell styles
@@ -8927,12 +9142,12 @@
| | | | Merge all = 0x8000
| | | | **Table properties:**
| | | | Flow direction bottom to top = 0x10000 |
| BL | 91 | Property flags. Contains property bit values for property Auto
Scale only |
| | | | (0x100).
-| BL | 92 | Value data type, see also paragraph 20.4.98. |
-| BL | 93 | Value unit type, see also paragraph 20.4.98. |
+| BL | 92 | Value data type, see also paragraph [20.4.98] (#20498-cell-cont
ent-geometry). |
+| BL | 93 | Value unit type, see also paragraph [20.4.98] (#20498-cell-cont
ent-geometry). |
| | | | Value format string
| | | | Rotation
| BD | 40 | Block scale
| BD | 140 | Cell alignment:
| BL | 94 | Top left = 1,
@@ -8948,11 +9163,11 @@
| | | | Text style handle (hard pointer) |
| | | | Text height

```

#### #### 20.4.101.4 Cell style

-Table cell style data is present in the cell style map object, in the table entity and also the table content object. A cell style inherits from content format. Cell style adds amongst others cell border style and margin properties to the content style properties of content format (see paragraph 20.4.101.3).

+Table cell style data is present in the cell style map object, in the table entity and also the table content object. A cell style inherits from content format. Cell style adds amongst others cell border style and margin properties to the content style properties of content format (see paragraph [20.4.101.3] (#20.4.101.3)).

Version	Field type	DXF group	Description
-----	-----	----	-----

BL	90	Cell style type: Cell = 1,
@@ -8961,18 +9176,18 @@		
		Formatted table data = 4, Table = 5
BS	170	Data flags, 0 = no data, 1 = data is present If data is present
BL	91	Property override flags. The definition is the same as the con
tent format		
-		property override flags, see paragraph 20.4.101.3.
+		property override flags, see paragraph [20.4.101.3] (#2041013-co
ntent-format).		
BL	92	Merge flags, but may only for bits 0x8000 and 0x10000.
TC	62	Background color
BL	93	Content layout flags: Flow = 1, Stacked horizontal = 2, Stacked vertical = 4
-	...	Content format fields (see paragraph 20.4.101.3).
+	...	Content format fields (see paragraph [20.4.101.3] (#2041013-con
tent-format)).		
BS	171	Margin override flags, bit 1 is set if margin overrides are pr
esent		
		If margin overrides are present
BD	40	Vertical margin
BD	40	Horizontal margin
BD	40	Bottom margin
@@ -9012,15 +9227,15 @@		

The cell style map is connected to the table style through an extension dictionary entry with name â\200\234ACAD\_ROUNDTRIP\_2008\_TABLESTYLE\_CELLSTYLEMAPâ\200\235 in the table styleâ\200\231s extension dictionary. The dictionary entry value points to the cell style map.

Version	Field type	DXF group	Description
-	...		Common AcDbObject fields, see paragraph 20.1.
+	...		Common AcDbObject fields, see paragraph [20.1] (#201-common-non
-entity-object-format).			
BL	90		Number of cell styles Begin repeat cell styles
-	...		Cell style fields, see paragraph 20.4.101.4.
-	BL	90	Cell style ID, 1 = title, 2 = header, 3 = data, 4 = table (new
in R24).			
+	...		Cell style fields, see paragraph [20.4.101.4] (#2041014-cell-st
yle).			
+	BL	90	Cell style ID, 1 = title, 2 = header, 3 = data, 4 = table (new
in R2010).			
a cell style in the			The cell style ID is used by cells, columns, rows to reference
tableâ\200\231s table style. Custom cell style IDâ\200\231s ar			
e numbered starting at 101.			
BL	91		Cell style class, 1= data, 2 = label. The default value is lab
el.			
TV	300		Cell style name End repeat cell styles
@@ -9029,11 +9244,11 @@			

This object represents a tableâ\200\231s geometry and was introduced in AutoCAD 2008. It does not need to be present in a DWG file.

Version	Field type	DXF group	Description
-	...		Common AcDbObject fields, see paragraph 20.1.
+	...		Common AcDbObject fields, see paragraph [20.1] (#201-common-non
-entity-object-format).			
BL	90		Row count



	BL	91	Column count
	BL	92	Row * column count
			Begin repeat rows
			Begin repeat columns
@@ -9052,15 +9267,18 @@			
	BD	95	Unknown (0).
			End repeat contents
			End repeat columns
			End repeat rows

-### 20.4.104 XRECORD (varies):

+### 20.4.104 XRECORD (varies)

'''

```
- Length MS -- Entity length (not counting itself or CRC).
- Type BS 0 typecode (internal DWG type code).
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
R2000+:
 Obj size RL size of object in bits, not including end handles
Common:
 Handle H 5 Length (char) followed by the handle bytes.
 EED X -3 See EED section.
```

@@ -9118,36 +9336,789 @@

```
00B1B 40 3E 00 01 00 46 00 B4 @>...F.. 0100 0000 0011 1110 0000 0000 0000 0001 0000 00
00 0100 0110 0000 0000 1011 0100
00B23 00 40 41 0C 30 .@A.0 0000 0000 0100 0000 0100 0001 0000 1100 0011 00
00
00B28 45 76 crc
'''
```

+### 20.4.105 AcDbEvalExpr subclass

```
+
+'''
+ parentid BL 0
+ major version BL 98 default: 33
+ minor version BL 99 default: 29
+ value_code BS 70 dxf code of the next value
+ If value_code == 40
+ num40 BD 40
+ Else If value_code == 10
+ pt2d 2RD 10
+ Else If value_code == 11
+ pt3d 3RD 11
+ Else If value_code == 1
+ text1 T 1
+ Else If value_code == 90
+ long90 BL 90
+ Else If value_code == 91
+ handle91 H 91 (code 5)
+ Else If value_code == 70
+ short70 BL 70
+ End If value_code
+ nodeid BL -
+'''
+
```

+### 20.4.106 AcDbShHistoryNode subclass

```
+
+'''
+ major version BL 90 Seen 27-33
+ minor version BL 91 Seen 29-106
+ trans 16xBD 40 transformation matrix
+ color CMC 62
+ step_id BL 92
```

```

+ material H 347
+ ``
+
+### 20.4.107 ACSH_BOX_CLASS
+
+Class properties:
+
+ | App name | ObjectDBX Classes |
+ |-----|-----|
+ | Class number | Dynamic (>= 500) |
+ | DWG version | R2000 |
+ | Maintenance version | 0 |
+ | Class proxy flags | 499 |
+ | C++ class name | AcDbShBox |
+ | DXF name | ACSH_BOX_CLASS |
+
+ ``
+
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ length BD 40
+ width BD 41
+ height BD 42
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+
+ ``
+
+### 20.4.108 ACSH_WEDGE_CLASS
+
+Class properties:
+
+ | App name | ObjectDBX Classes |
+ |-----|-----|
+ | Class number | Dynamic (>= 500) |
+ | DWG version | R2000 |
+ | Maintenance version | 0 |
+ | Class proxy flags | 499 |
+ | C++ class name | AcDbShWedge |
+ | DXF name | ACSH_WEDGE_CLASS |
+
+
+Same fields as ACSH_BOX_CLASS.
+
+ ``
+
+ Length MS -- Object length (not counting itself or CRC).

```

```
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ length BD 40
+ width BD 41
+ height BD 42
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+```
+
+### 20.4.109 ACSH_SPHERE_CLASS
+
++Class properties:
+
++ | App name | ObjectDBX Classes |
++ |-----|-----|
++ | Class number | Dynamic (>= 500) |
++ | DWG version | R2000 |
++ | Maintenance version | 0 |
++ | Class proxy flags | 499 |
++ | C++ class name | AcDbShSphere |
++ | DXF name | ACSH_SPHERE_CLASS |
++ |
++ ```
+
++ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
```

```

+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ radius BD 40
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.110 ACSH_CYLINDER_CLASS
+
+Class properties:
+
+| App name | ObjectDBX Classes |
+|-----|-----|
+| Class number | Dynamic (>= 500) |
+| DWG version | R2000 |
+| Maintenance version | 0 |
+| Class proxy flags | 499 |
+| C++ class name | AcDbShCylinder |
+| DXF name | ACSH_CYLINDER_CLASS |
+
+'''
+
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ height BD 40
+ major_radius BD 41
+ minor_radius BD 42
+ x_radius BD 43
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.111 ACSH_CONE_CLASS
+
+Class properties:
+
+| App name | ObjectDBX Classes |
+|-----|-----|
+| Class number | Dynamic (>= 500) |
+| DWG version | R2000 |
+| Maintenance version | 0 |

```

```

+ | Class proxy flags | 499 |
+ | C++ class name | AcDbShCone |
+ | DXF name | ACSH_CONE_CLASS |
+
+Same fields as ACSH_CYLINDER_CLASS.
+
+'''
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ height BD 40
+ major_radius BD 41
+ minor_radius BD 42
+ x_radius BD 43
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.112 ACSH_PYRAMID_CLASS
+
+Class properties:
+
+ | App name | ObjectDBX Classes |
+ |-----|-----|
+ | Class number | Dynamic (>= 500) |
+ | DWG version | R2000 |
+ | Maintenance version | 0 |
+ | Class proxy flags | 499 |
+ | C++ class name | AcDbShPyramid |
+ | DXF name | ACSH_PYRAMID_CLASS |
+
+'''
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles

```

```

+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ height BD 40
+ sides BL 92
+ radius BD 41
+ topradius BD 42
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.113 ACSH_FILLET_CLASS
+
+Class properties:
+
+| App name | ObjectDBX Classes |
+|-----|-----|
+| Class number | Dynamic (>= 500) |
+| DWG version | R2000 |
+| Maintenance version | 0 |
+| Class proxy flags | 499 |
+| C++ class name | AcDbShFillet |
+| DXF name | ACSH_FILLET_CLASS |
+
+'''
+
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ num_edges BL 93
+ Repeat num_edges
+ edges BL 94
+ End Repeat num_edges
+ num_radiuses BL 93
+ Repeat num_radiuses
+ radiuses BD 41

```

```

+ End Repeat num_radiuses
+ num_startsetbacks BL 96
+ num_endsetbacks BL 97
+ Repeat num_endsetbacks
+ endsetbacks BD 43
+ End Repeat num_endsetbacks
+ Repeat num_startsetbacks
+ startsetbacks BD 42
+ End Repeat num_startsetbacks
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.114 ACSI\CHAMFER\CLASS
+
+Class properties:
+
+| App name | ObjectDBX Classes |
+|-----|-----|
+| Class number | Dynamic (>= 500) |
+| DWG version | R2000 |
+| Maintenance version | 0 |
+| Class proxy flags | 499 |
+| C++ class name | AcDbShChamfer |
+| DXF name | ACSI\CHAMFER\CLASS |
+
+'''
+
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ unknown BL 92
+ base_dist BD 41
+ other_dist BD 42
+ num_edges BL 93
+ Repeat num_edges
+ edges BL 94
+ End Repeat num_edges
+ unknown BL 95
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+

```

# ``` #### 20.4.115 ACSI\TORUS\CLASS ```

```
+
+Class properties:
```

App name	ObjectDBX Classes
Class number	Dynamic ( $\geq 500$ )
DWG version	R2000
Maintenance version	0
Class proxy flags	499
C++ class name	AcDbShTorus
DXF name	ACSH\TORUS\CLASS

```
+
+'''
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ major_radius BD 41
+ minor_radius BD 42
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
```

# ``` #### 20.4.116 ACSI\BREP\CLASS ```

```
+
+Class properties:
```

App name	ObjectDBX Classes
Class number	Dynamic ( $\geq 500$ )
DWG version	R2000
Maintenance version	0
Class proxy flags	499
C++ class name	AcDbShBrep
DXF name	ACSH\BREP\CLASS

```
+
+'''
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
```



```

+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ 3DSOLID See chapter 20.4.41
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.117 ACSH_BOOLEAN_CLASS
+
+Class properties:
+
+| App name | ObjectDBX Classes |
+|-----|-----|
+| Class number | Dynamic (>= 500) |
+| DWG version | R2000 |
+| Maintenance version | 0 |
+| Class proxy flags | 499 |
+| C++ class name | AcDbShBoolean |
+| DXF name | ACSH_BOOLEAN_CLASS |
+
+'''
+
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+ AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+ major BL 90
+ minor BL 91
+ operation RC 280
+ operand1 BL 92
+ operand2 BL 93
+
+ Handle refs H parenthandle (soft pointer)

```

```

+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.118 ACSH_HISTORY_CLASS
+
+Class properties:
+
+| App name | ObjectDBX Classes |
+|-----|-----|
+| Class number | Dynamic (>= 500) |
+| DWG version | R2000 |
+| Maintenance version | 0 |
+| Class proxy flags | 499 |
+| C++ class name | AcDbShHistory |
+| DXF name | ACSH_HISTORY_CLASS |
+
+'''
+
+Length MS -- Object length (not counting itself or CRC).
+R2010+:
+Handle Stream Size MC -- not counted in the Length
+Common:
+Type OT 0 typecode (internal DWG type code).
+R2000+:
+Obj size RL size of object in bits, not including end handles
+Common:
+Handle H 5 Length (char) followed by the handle bytes.
+EED X -3 See EED section.
+R13-R14 Only:
+Obj size RL size of object in bits, not including end handles
+Common:
+Numreactors BL number of reactors in this object
+R2004+:
+XDic Missing Flag B If 1, no XDictionary handle is stored for this
+object, otherwise XDictionary handle is stored as in
+R2000 and earlier.
+
+Common:
+AcDbEvalExpr ... See 20.4.105 AcDbEvalExpr subclass
+AcDbShHistoryNode ... See 20.4.106 AcDbShHistoryNode subclass
+major BL 90
+minor BL 91
+owner H 260 code 2
+h_nodeid BL 92
+show_history B 280
+record_history B 281
+
+Handle refs H parenthandle (soft pointer)
+[Reactors (soft pointer)]
+xdictionary (hard owner)
+'''
+
+### 20.4.119 SUN
+
+Hard-owned child of AcDbViewportTableRecord or AcDbViewport 361.
+The AutoDesk DXF docs put that as Entity, wrong.
+
+Class properties:
+
+| App name | ObjectDBX Classes |
+|-----|-----|
+| Class number | Dynamic (>= 500) |
+| DWG version | R2000 |
+| Maintenance version | 0 |
+| Class proxy flags | 499 |
+| C++ class name | AcDbSun |
+| DXF name | SUN |

```

```

+
+'''
+ Length MS -- Object length (not counting itself or CRC).
+R2010+:
+ Handle Stream Size MC -- not counted in the Length
+Common:
+ Type OT 0 typecode (internal DWG type code).
+R2000+:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Handle H 5 Length (char) followed by the handle bytes.
+ EED X -3 See EED section.
+R13-R14 Only:
+ Obj size RL size of object in bits, not including end handles
+Common:
+ Numreactors BL number of reactors in this object
+R2004+:
+ XDic Missing Flag B If 1, no XDictionary handle is stored for this
+ object, otherwise XDictionary handle is stored as in
+ R2000 and earlier.
+
+Common:
+ class_version BL 90
+ is_on B 290
+ color CMC 63/421
+ intensity BD 40
+ has_shadow B 291
+ julian_day BL 91
+ msecs BL 92
+ is_dst B 292 isDayLightSavingsOn
+ shadow_type BL 70 0 raytraced, 1 shadow maps
+ shadow_mapsize BS 71 max 3968
+ shadow_softness RC 280
+
+ Handle refs H parenthandle (soft pointer)
+ [Reactors (soft pointer)]
+ xdictionary (hard owner)
+'''
+
+### 20.4.120 REPEAT (pre-R2.1 only: 5)
+
+No fields. Like a block, followed by entities to be repeated, until
+ENDREP.
+
+ Common Entity Data
+
+### 20.4.121 ENDREP (pre-R2.1 only: 6)
+
+'''
+ Common Entity Data
+ numcols RS 70
+ numrows RS 71
+ colspacing RD 40
+ rowspacing RD 41
+'''
+
+### 20.4.122 3DLINE (R2.4-R11 only: 21)
+
+R2.4-R11 only.
+'''
+ Common Entity Data
+ R_2_4-R_9c1:
+ if (R11OPTS (1)) {
+ FIELD_3RD (start, 10);
+ } else {
+ FIELD_2RD (start, 10);
+ }
+

```

```

+ if (R11OPTS (2)) {
+ FIELD_3RD (end, 11);
+ } else {
+ FIELD_2RD (end, 11);
+ }
+ R10-R11:
+ FIELD_3RD (start, 10)
+ FIELD_3RD (end, 11)
+ if (R11OPTS (1))
+ FIELD_3RD (extrusion, 210);
+```
+### 20.4.123 JUMP (pre-R13 only: 10)
+
+Only R2_0b - R13b1.
+
+When there is no room to extend an existing entity, the entity is
+replaced by a JUMP entity type, which gives the offset into the EXTRAS
+section, until a JUMP in EXTRAS jumps back to the next original
+entity.
+
+```
+ Common Entity Data
+ jump_entity_section RC 0: ENTITIES, 0x40: BLOCKS, 0x80: EXTRAS
+ jump_address 3xRC offset into one of the 3 sections
+R11+:
+ CRC RS
+```
+
+### 20.4.124 LOAD (pre-r2.0 only: 10)
+
+Only before R2_0b.
+
+```
+ Common Entity Data
+ file_name TV 1
+```
+
+# 21 Data section AcDb:ObjFreeSpace

```

-The meaning of this section is not completely known. The ODA knows how to write a valid section, but

-the meaning is not known of every field.

+From R13 to R2000 this section is the third section, which is immediately followed by the SECOND FILE HEADER (R13-R2000). See [chapter 26](#26-second-file-header-r13-r2000).

-## 21.1 Until R18

+## 21.1 Until R2007

Type	Length	Description
Int32	4	0
UInt32	4	Approximate number of objects in the drawing (number of handles).
Julian datetime	8	If version > R14 then system variable TDUPDATE otherwise TDUUPD
ATE.		
UInt32	4	Offset of the objects section in the stream.
UInt8	1	Number of 64-bit values that follow (ODA writes 4).
UInt32	4	ODA writes 0x00000032.
UInt32	4	ODA writes 0x00000000.
UInt32	4	ODA writes 0x00000064.
UInt32	4	ODA writes 0x00000000.
UInt32	4	ODA writes 0x00000200.
UInt32	4	ODA writes 0x00000000.
UInt32	4	ODA writes 0xffffffff.
UInt32	4	ODA writes 0x00000000.
UInt32	4	Offset of the objects section in the stream. 0 since R2000
UInt8	1	Number of 64-bit values that follow (Always 4).

+	UInt64	8	max32, 0x00000032.
+	UInt64	8	max64, 0x00000064.
+	UInt64	8	maxtbl, 0x00000200.
+	UInt64	8	maxrl, 0xffffffff.
+			
+	## 21.2 Since R2010		
+			
+	Type	Length	Description
+	-----	-----	-----
+	Int64	8	0
+	UInt64	8	Approximate number of objects in the drawing (number of handles).
+	Julian datetime	8	If version > R14 then system variable TDUPDATE otherwise TDUPDATE.
+	UInt8	1	Number of 64-bit (resp. 128-bit) values that follow (Always 4).
+	UInt64	8	max32, 0x00000032.
+	UInt64	8	max32 hi, 0x00000000.
+	UInt64	8	max64, 0x00000064.
+	UInt64	8	max64 hi, 0x00000000.
+	UInt64	8	maxtbl, 0x00000200.
+	UInt64	8	maxtbl hi, 0x00000000.
+	UInt64	8	maxrl, 0xffffffff.
+	UInt64	8	maxrl hi, 0x00000000.

# 22 Data section: AcDb:Template

-This section is optional in releases 13-15. The section is mandatory in the releases 18 and newer. The template section only contains the MEASUREMENT system variable.

+This section is optional in releases r13-r2000. The section is mandatory in the releases R2004 and newer. The template section only contains the MEASUREMENT system variable.

	Type	Length	Description
	-----	-----	-----
	Int16	2	Template description string length in bytes (the ODA always writes 0 here).
			codepage to encode the bytes).
	UInt16	2	MEASUREMENT system variable (0 = English, 1 = Metric).

# 23 Data section AcDb:Handles (OBJECT MAP)

## 23.1 R13-15

## 23.1 R13-2000

The Object Map is a table which gives the location of each object in the file This tab

le is broken into sections. It is basically a list of handle/file loc pairs, and goes (something like) this:

```

'''
Set the "last handle" to all 0 and the "last loc" to 0L;
@@ -9177,23 +10148,23 @@
End top repeat
'''

```

Note that each section is cut off at a maximum length of 2032.

```

-## 23.2 R18
+## 23.2 R2004

```

-This section is compressed and contains the standard 32 byte section header. The decompressed data in this section is identical to the â\200\234Object Mapâ\200\235 section data found in R15 and earlier files, excepts that offsets are not absolute file addresses, but are instead offsets into the AcDb:Objects logical section (starting with offset 0 at the beginning of this logical section).

+This section is compressed and contains the standard 32 byte section header. The decompressed data in this section is identical to the â\200\234Object Mapâ\200\235 section data found in R2000 and earlier files, excepts that offsets are not absolute file addresses, but are instead offsets into the AcDb:Objects logical section (starting with offset 0 at the beginning of this logical section).

# 24 Section AcDb:AcDsPrototype\_1b (DataStorage)

At this moment (December 2012), this sections contains information about Acis data (regions, solids).

The data is stored in a byte stream, not a bit stream like e.g. the objects section.

The data store contains several data segments, and index segments that contain lookup information for finding the data segments and objects within these data segments. The file header contains the stream position of the segment index file segment and the segment indexes for the schema index/data index/search file segments. The segment index file segment is a lookup table for finding the stream position of a file segment by its segment index.

-In paragraph 24.3 the default contents of this section is shown when empty.

+In paragraph [24.3] (#243-default-contents) the default contents of this section is shown when empty.

## 24.1 File header

Version	Field type	DXF group	Description
@@ -9202,19 +10173,19 @@		:	
	Int32		Unknown 1 (always 2?)
	Int32		Version (always 2?)
	Int32		Unknown 2 (always 0?)
	Int32		Data storage revision
	Int32		Segment index offset (the stream off set from the data storeâ\200\231s stream start
-			position). See paragraph 24.2.2.1 for the segment index file segment.
+			position). See paragraph [24.2.2.1] (#24221-segment-index-file-segment) for the segment index file segment.
	Int32		Segment index unknown
	Int32		Segment index entry count
	Int32		Schema index segment index. This is the index into the segment index entry
-			array (see paragraph 24.2.2.1) for the schema index file segment (see
-			paragraph 24.2.2.4).
+			array (see paragraph [24.2.2.1] (#24221-segment-index-file-segment)) for the schema index file segment (see

Version	Field type	DXF group	Description
	Int32		paragraph [24.2.2.4] (#24224-blob01-file-segment)).
	Int32		Data index segment index. This is the index into the segment i
	Int32		array (see paragraph 24.2.2.1) for the data index file segment
	Int32		(see paragraph 24.2.2.2).
	Int32		array (see paragraph [24.2.2.1] (#24221-segment-index-file-segm
	Int32		ent)) for the data index file segment (see paragraph [
	Int32		[24.2.2.2] (#24222-data-index-file-segment)).
	Int32		Search segment index
	Int32		Previous save index
	Int32		File size

## 24.2 File segment  
@@ -9263,26 +10234,26 @@  
The segment is looked up by the index in the array.

Version	Field type	DXF group	Description
			Begin repeat segment index entry count (as present in the fil
			e header, see
			paragraph 24.1)
			paragraph [24.1] (#24.1))
	UInt64		Offset. This is the offset from the data storeâ\200\231s stre
	UInt32		Size
			End repeat segment index entry count

#### 24.2.2.2 Data index file segment

-This file segment contains index entries for objects within the data file segment (see paragraph 24.2.2.3).  
+This file segment contains index entries for objects within the data file segment (see paragraph [24.2.2.3] (#24223-data-file-segment)).

Version	Field type	DXF group	Description
	Int32		Entry count Int32 Unknown (always 0?)
	UInt32		Begin repeat of entries (entry count)
	UInt32		Segment index (0 means stub entry and can be ignored).
	UInt32		Local offset. This is a local offset in the stream, relative
			to the file segmentâ\200\231s
			stream start position. This points to a data file segment, se
			e paragraph 24.2.2.3.
			stream start position. This points to a data file segment, se
			e paragraph [24.2.2.3] (#24223-data-file-segment).
	UInt32		Schema index
			End repeat of entries

#### 24.2.2.3 Data file segment

@@ -9290,33 +10261,33 @@

x data records, where each data record is a byte array. Relatively small amounts of data are stored directly in the data file segment (up to 0x40000 bytes).

x A data blob references, where each blob reference references one or more other blob file segments.

-These other file segments represent the pages of the blob (paragraph 24.2.2.3.1). Large byte arrays are stored into multiple of these pages (more than 0x40000 bytes, max 0xffb0 bytes per page).

+These other file segments represent the pages of the blob (paragraph [24.2.2.3.1] (#242231-data-blob-reference-record)). Large byte arrays are stored into multiple of these pages (more than 0x40000 bytes, max 0xfffb0 bytes per page).

-For each entityâ\200\231s binary data stored in the data file segment entries have to be created in the schema search data. See paragraph 24.2.2.7.1. When reading the schema search data can be ignored.

+For each entityâ\200\231s binary data stored in the data file segment entries have to be created in the schema search data. See paragraph [24.2.2.7.1] (#242271-schema-search-data). When reading the schema search data can be ignored.

For each ACIS entity (REGION, 3DSOLID), a data record is created with the SAB stream of the object.

More detailed description of the ACIS/SAB data falls outside the scope of this document. The SAB stream bytes are prefixed with the ASCII encoded bytes of the string â\200\234ACIS BinaryFileâ\200\235. When for an ACIS entity a SAB stream is created from SAT, then if the version >= 21800, the bytes are post fixed with the ASCII encoded bytes of the string â\200\234End-of-ASM-dataâ\200\235, otherwise â\200\234End-of-ACIS-dataâ\200\235.

Version	Field type	DXF group	Description
-----	-----	----	-----

- Begin repeat (data record) headers. Repeats number of local offsets times (this is read earlier from the data index, see paragraph 24.2.2.2). For a particular data file segment, find all data index entries with the segmentâ\200\231s segment index and take the local offsets. Move the stream position according to the current header local offset, which is relative to this data file segments stream start position. UInt32 Entry size UInt32 Unknown (ODA writes 1) UInt64 Handle UInt32 Local offset, a stream offset relative to the data start marker (just after this list of data record headers). End repeat (data record) header offsets Data start marker, this is the beginning of all data records. Begin repeat header entries (that were read above) Each data record starts at the data start marker position + local offset. The maxRecordSize of the record is the difference between two consecutive stream offsets. For the last data record the size is the file segment headerâ\200\231s (object data alignment offset < 4) + segment size the recordâ\200\231s stream offset (i.e. the file segment end position â\200\223 the record start position). UInt32 dataSize If ((dataSize + 4) <= maxRecordSize) Byte[] Data recordâ\200\231s bytes of length dataSize Else If (dataSize == 0xbb106bb1) Data blob reference record, see paragraph 24.2.2.3.1 End If End repeat header entries

+ Begin repeat (data record) headers. Repeats number of local offsets times (this is read earlier from the data index, see paragraph [24.2.2.2] (#24.2.2.2)). For a particular data file segment, find all data index entries with the segmentâ\200\231s segment index and take the local offsets. Move the stream position according to the current header local offset, which is relative to this data file segments stream start position. UInt32 Entry size UInt32 Unknown (ODA writes 1) UInt64 Handle UInt32 Local offset, a stream offset relative to the data start marker (just after this list of data record headers). End repeat (data record) header offsets Data start marker, this is the beginning of all data records. Begin repeat header entries (that were read above) Each data record starts at the data start marker position + local offset. The maxRecordSize of the record is the difference between two consecutive stream offsets. For the last data record the size is the file segment headerâ\200\231s (object data alignment offset < 4) + segment size the recordâ\200\231s stream offset (i.e. the file segment end position â\200\223 the record start position). UInt32 dataSize If ((dataSize + 4) <= maxRecordSize) Byte[] Data recordâ\200\231s bytes of length dataSize Else If (dataSize == 0xbb106bb1) Data blob reference record, see paragraph [24.2.2.3.1] (#242231-data-blob-reference-record) End If End repeat header entries

##### 24.2.2.3.1 Data blob reference record

-A data blob reference references one or more other file segments. These other file segments represent the pages of the blob. Each page is stored in a Blob01 file segment, see paragraph 24.2.2.4.

+A data blob reference references one or more other file segments. These other file segments represent the pages of the blob. Each page is stored in a Blob01 file segment, see paragraph [24.2.2.4] (#24224-blob01-file-segment).

Version	Field type	DXF group	Description
-----	-----	----	-----

UInt64 Total data size UInt32 Page count UInt32 Record size (the size of this data blob)



ob reference record UInt32 Page size UInt32 Last page size UInt32 Unknown 1 (ODA writes 0)

- UInt32 Unknown 2 (ODA writes 0) Begin repeat page count UInt32 Segment index. The pageâ\200\231s blob01 file segment stream position can be found by a lookup in the segment index file segment using the segment index, see paragraph 24.2.2.1. UInt32 Size End repeat page count

+ UInt32 Unknown 2 (ODA writes 0) Begin repeat page count UInt32 Segment index. The pageâ\200\231s blob01 file segment stream position can be found by a lookup in the segment index file segment using the segment index, see paragraph [24.2.2.1] (#24221-segment-index-file-segment). UInt32 Size End repeat page count

#### 24.2.2.4 Blob01 file segment

Version	Field type	DXF group	Description
----	-----	----	-----
@@ -9327,58 +10298,58 @@			
	UInt64		Page data size
	byte[]		Binary data (byte array) of size Page data size

##### 24.2.2.5 Schema index file segment

-The schema index contains references to objects within the schema data file segment, see paragraph 24.2.2.6.

+The schema index contains references to objects within the schema data file segment, see paragraph [24.2.2.6] (#24226-schema-data-file-segment).

Version	Field type	DXF group	Description
----	-----	----	-----
	UInt32		Unknown property count
	UInt32		Unknown (0)
	UInt32		Begin repeat schema unknown property count
	UInt32		Index (starting at 0)
(paragraph	UInt32		Segment index into the segment index file segment entry table
-			24.2.2.1) of the schema data file segment (paragraph 24.2.2.6
)			
+			[24.2.2.1] (#24221-segment-index-file-segment)) of the schema
data file segment (paragraph [24.2.2.6] (#24226-schema-data-file-segment))			
	UInt32		Local offset of the unknown schema property. This is a local
offset in the			stream, relative to the schema data file segmentâ\200\231s st
ream start position.			
	Int64		End repeat schema unknown property count
	UInt32		Unknown (0x0af10c)
	UInt32		Property entry count
	UInt32		Unknown (0)
	UInt32		Begin repeat property entry count
(paragraph	UInt32		Segment index into the segment index file segment entry table
-			24.2.2.1) of the schema data file segment (paragraph 24.2.2.6
)			
+			[24.2.2.1] (#24221-segment-index-file-segment)) of the schema
data file segment (paragraph [24.2.2.6] (#24226-schema-data-file-segment)).			
	UInt32		Local offset of the schema property. This is a local offset i
n the stream, relative			to the schema data file segmentâ\200\231s stream start positi
on.			
	UInt32		Index
			End repeat property entry count

#### 24.2.2.6 Schema data file segment

-The schema data file segment contains unknown properties and schemas. The stream offsets of these objects from the start of this file segment are found in the schema index, see paragraph 24.2.2.5.

+The schema data file segment contains unknown properties and schemas. The stream offse

ts of these objects from the start of this file segment are found in the schema index, see paragraph [24.2.2.5] (#24225-schema-index-file-segment).

Version	Field type	DXF group	Description
-----	-----	----	-----
			Begin repeat schema unknown properties in the associated schema index file
-			segment (paragraph 24.2.2.4), where the property's segment index is equal to
+			segment (paragraph [24.2.2.4] (#24224-blob01-file-segment)), where the property's segment index is equal to
			this file segment's segment index (found in the header).
	UInt32		Data size
	UInt32		Unknown flags
			End repeat schema unknown properties
			Begin repeat schema entries in the associated schema index file segment
-			(paragraph 24.2.2.4), where the property's segment index is equal to this file
+			(paragraph [24.2.2.4] (#24224-blob01-file-segment)), where the property's segment index is equal to this file
			segment's segment index (found in the header).
-			A schema, see paragraph 24.2.2.6.1. The stream position is the file segment's
+			A schema, see paragraph [24.2.2.6.1] (#242261-schema) The stream position is the file segment's
			start position + the schema entry's local offset.
			End repeat schema entries
	UInt32		Property name count
			Begin repeat property name count
	AnsiString		Property name (zero byte delimited). These names are referred to by the
-			schema's schema property's name index (paragraph 24.2.2.6.1.1). Name
+			schema's schema property's name index (paragraph [24.2.2.6.1] (#242261-schema)). Name
			strings can be shared between multiple schema properties this way.
-			See paragraph 24.2.2.6.1 for details about the schema.
+			See paragraph [24.2.2.6.1] (#242261-schema) for details about the schema.
			End repeat property name count

#### ##### 24.2.2.6.1 Schema

A schema is a collection of name value pairs, where the value can have a number of types.

@@ -9389,11 +10360,11 @@

	UInt64		Begin repeat index count
			Index
			End repeat index count
	UInt16		Property count
			Begin repeat property count
-			Schema property, see paragraph 24.2.2.6.1.1.
+			Schema property, see paragraph [24.2.2.6.1.1] (#24226-schema-data-file-segment.1.1.)
			End repeat property count

#### ##### 24.2.2.6.1.1 Schema property

This is a schema (see 24.2.2.6.1) property, having a name and a value of a certain type.

@@ -9403,11 +10374,11 @@

	UInt32	91	Property flags:
			* 1 = Unknown 1 (if set then all other bits are cleared).
			* 2 = Has no type.

	UInt32	2	* 8 = Unknown 2 (if set then all other bits are cleared). Name index. Index into a property names array in the schema data file segment
-			(see paragraph 24.2.2.6). In a DXF file the name is directly written instead of
+			(see paragraph [24.2.2.6] (#24226-schema-data-file-segment)). In a DXF file the name is directly written instead of
			indirectly through a table lookup.
	UInt32	280	If property flags bit 2 is NOT set
			Type (0-15)
	UInt32		If type == 0xe
			Custom type size

@@ -9430,11 +10401,11 @@

Version	Field type	DXF group	Description
	UInt32		Schema count
-			Begin repeat schema count
+			Schema search data, see paragraph 24.2.2.7.1.
-			Schema search data, see paragraph [24.2.2.7.1] (#242271-schema-search-data)
			End repeat schema count

#### ##### 24.2.2.7.1 Schema search data

The purpose of this segment is unknown. It seems to contain redundant data coupling a (sort) index to the objects in the data segment. When reading the schema search data can be ignored.

@@ -9454,11 +10425,11 @@

			If ID indexes count > 0
	UInt32		Unknown (0)
			Begin repeat ID indexes count
	UInt32		ID index count
			Begin repeat ID index count (in this loop the ID entry object is serialized)
-	UInt64		Handle of the object present in the data segment (see paragraph 24.2.2.3).
+	UInt64		Handle of the object present in the data segment (see paragraph [24.2.2.3] (#24223-data-file-segment)).
	UInt64		Index count
			Begin repeat index count
	UInt64		Index (same as Sorted index value above). The ODA only writes one index per
			handle.
			End repeat index count

@@ -9630,119 +10601,74 @@

```

}
handleToDataRecord {
}
...

```

#### -# 25 UNKNOWN SECTION

-This section is largely unknown. The total size of this section is 53. We simply patch in "known to be valid" data. We first write a 0L, then the number of entries in the ob jmap +3, as a long. Then 45 bytes of "known to be valid data". Then we poke in the start address for objects at offset 16.

-The 45 bytes of known to be valid data are:

+# 26 SECOND FILE HEADER (R13-R2000)

```

- 0xA7, 0x62, 0x25, 0x00, 0xF6, 0xAF, 0x25, 0x02,
- 0x3B, 0x04, 0x00, 0x00, 0x04, 0x32, 0x00, 0x00,
- 0x00, 0x00, 0x00, 0x00, 0x00, 0x64, 0x00, 0x00,
- 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00,
- 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF,
- 0xFF, 0x00, 0x00, 0x00, 0x00

```

+This is directly after the ObjFreeSpace section. See [chapter 21] (#21-data-section-acd bobjfreeSpace).

-# 26 SECOND FILE HEADER (R13-R15)

-

-## 26.1 Beginning sentinel

+Beginning sentinel

{0xD4,0x7B,0x21,0xCE,0x28,0x93,0x9F,0xBF,0x53,0x24,0x40,0x09,0x12,0x3C,0xAA,0x01 }

;

- RL : size of this section

- L : Location of this header (long, loc of start of sentinel).

- RC : "AC1012" or "AC1014" for R13 or R14 respectively

- RC : 6 0's

- B : 4 bits of 0

- RC : 0x18,0x78,0x01,0x04 for R13, 0x18,0x78,0x01,0x05 for R14

-

- RC : 0

- L : header address

- L : header size

- RC : 1

- L : class address

- L : class data size

- RC : 2

- L : Object map address (natural table)

- L : Object map size

- RC : 3

- L : Address of unknown section 3

- L : size of that section

-

- S : 14 (# of handle records following)

-

- RC : size of (valid chars in) handseed

- RC : 0

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) block control objhandle

- RC : 1

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) layer control objhandle

- RC : 2

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) shapefile control objhandle

- RC : 3

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) linetype control objhandle

- RC : 4

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) view control objhandle

- RC : 5

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) ucs control objhandle

- RC : 6

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) viewport control objhandle

- RC : 7

- RC : "size" characters of the handle

-

- RC : size of (valid chars in) reg app control objhandle

- RC : 8

```

- RC : "size" characters of the handle
-
- RC : size of (valid chars in) dimstyle control objhandle
- RC : 9
- RC : "size" characters of the handle
-
- RC : size of (valid chars in) viewport entity header objhandle
- RC : 10
- RC : "size" characters of the handle
-
- RC : size of (valid chars in) dictionary objhandle
- RC : 11
- RC : "size" characters of the handle
-
- RC : size of (valid chars in) default multi-line style objhandle
- RC : 12
- RC : "size" characters of the handle
-
- RC : size of (valid chars in) group dictionary objhandle
- RC : 13
+'''
+ RL : Size of this section
+ BL : Location of this header (long, loc of start of sentinel).
+ RC : "AC1012", "AC1013", "AC1014" or "AC1015" for AutoCAD releases.
+ RC : 5 0's
+ RC : Maintenance release version
+ RC : Byte 0x00, 0x01, or 0x03
+ BS : Acad version that writes the file (first byte is application version and second byte is application maintenance release version)
+ RS : Codepage
+
+ BS : Number of sections
+ Repeat Number of sections
+ RC : Id of section
+ BL : Section address
+ BL : Section size
+ End Repeat Number of sections
+
+ BS : 14 (# of handle records)
+ Repeat Number of handles
+ RC : size of handle in bytes
+ RC : index of handle
+ RC : "size" characters of the handle
+ End Repeat Number of handles

```

CRC

RC : 8 bytes of junk (R14 only). Note that the junk is counted in the size of this section at the start.

```

+'''
+
+Handles:
+
+'''
+0: handseed
+1: block control objhandle
+2: layer control objhandle
+3: style control objhandle
+4: ltype control objhandle
+5: view control objhandle
+6: ucs control objhandle
+7: vport control objhandle
+8: appid control objhandle
+9: dimstyle control objhandle
+10: vx control objhandle
+11: dictionary objhandle
+12: mlstyle objhandle

```

+13: group dictionary objhandle  
+```\n

Ending sentinel

{0x2B,0x84,0xDE,0x31,0xD7,0x6C,0x60,0x40,0xAC,0xDB,0xBF,0xF6,0xED,0xC3,0x55,0xFE}

# 27 Data section: AcDb:AuxHeader (Auxiliary file header)

-The auxiliary file header contains mostly redundant information and was introduced in R15.

+The auxiliary file header contains mostly redundant information and was introduced in R2000.

RC : 0xff 0x77 0x01

RS : DWG version:

AC1010 = 17,

AC1011 = 18,

@@ -9796,11 +10722,11 @@

RL : 0

RL : 0

RL : 0

RL : 0

-R2018+

+R2018+:

RS : 0

RS : 0

RS : 0

@@ -9855,11 +10781,11 @@

If that bit is 1, then following it, and preceding the RL which indicates the number of bits in the object, is an RL which indicates the number of bytes of proxy entity graphic data to follow.

Graphics data is padded to 4 byte boundaries! So, for instance, strings which are too short are padded out to the next 4 byte boundary. Similarly for lists of shorts.

-In addition to the data definitions from chapter 2 there are a few additional data types:

+In addition to the data definitions from [chapter 2] (#2-bit-codes-and-data-definitions) there are a few additional data types:

PS : Padded string. This is a string, terminated with a zero byte. The fileâ\200\231s text encoding (code page) is used to encode/decode the bytes into a string.

PUS : Padded Unicode string. The bytes are encoded using Unicode encoding. The bytes consist of byte pairs and the string is terminated by 2 zero bytes.