The following are the known types of zipfile extra fields as of this writing. Extra fields are documented in PKWARE's appnote.txt and are intended to allow for backward- and forward-compatible extensions to the zipfile format. Multiple extra-field types may be chained together, provided that the total length of all extra-field data is less than 64KB. (In fact, PKWARE requires that the total length of the entire file header, including timestamp, file attributes, filename, comment, extra field, etc., be no more than 64KB.)

Each extra-field type (or subblock) must contain a four-byte header consisting of a two-byte header ID and a two-byte length (little-endian) for the remaining data in the subblock. If there are additional subblocks within the extra field, the header for each one will appear immediately following the data for the previous subblock (i.e., with no padding for alignment).

All integer fields in the descriptions below are in little-endian (Intel) format unless otherwise specified. Note that "Short" means two bytes, "Long" means four bytes, and "Long-Long" means eight bytes, regardless of their native sizes. Unless specifically noted, all integer fields should be interpreted as unsigned (non-negative) numbers.

Christian Spieler, 20010517

Updated to include the Unicode extra fields. Added new Unix extra field.

Ed Gordon, 20060819, 20070607, 20070909, 20080426, 20080509

          Header ID's of 0 thru 31 are reserved for use by PKWARE.
          The remaining ID's can be used by third party vendors for
          proprietary usage.

          The current Header ID mappings defined by PKWARE are:

          0x0001   ZIP64 extended information extra field
          0x0007   AV Info
          0x0009   OS/2 extended attributes   (also Info-ZIP)
          0x000a   NTFS (Win9x/WinNT FileTimes)
          0x000c   OpenVMS   (also Info-ZIP)
          0x000d   Unix
          0x000f   Patch Descriptor
          0x0014   PKCS#7 Store for X.509 Certificates
          0x0015   X.509 Certificate ID and Signature for
                        individual file
          0x0016   X.509 Certificate ID for Central Directory

          The Header ID mappings defined by Info-ZIP and third parties are:

          0x0065   IBM S/390 attributes - uncompressed

```
          0x0066        IBM S/390 attributes - compressed
          0x07c8        Info-ZIP Macintosh (old, J. Lee)
          0x2605        ZipIt Macintosh (first version)
          0x2705        ZipIt Macintosh v 1.3.5 and newer (w/o full filename)
          0x334d        Info-ZIP Macintosh (new, D. Haase's 'Mac3' field )
          0x4154        Tandem NSK
          0x4341        Acorn/SparkFS (David Pilling)
          0x4453        Windows NT security descriptor (binary ACL)
          0x4704        VM/CMS
          0x470f        MVS
          0x4854        Theos, old inofficial port
          0x4b46        FWKCS MD5 (see below)
          0x4c41        OS/2 access control list (text ACL)
          0x4d49        Info-ZIP OpenVMS (obsolete)
          0x4d63        Macintosh SmartZIP, by Macro Bambini
          0x4f4c        Xceed original location extra field
          0x5356        AOS/VS (binary ACL)
          0x5455        extended timestamp
          0x5855        Info-ZIP Unix (original; also OS/2, NT, etc.)
          0x554e        Xceed unicode extra field
          0x6375        Info-ZIP Unicode Comment
          0x6542        BeOS (BeBox, PowerMac, etc.)
          0x6854        Theos
          0x7075        Info-ZIP Unicode Path
          0x756e        ASi Unix
          0x7855        Info-ZIP Unix (previous new)
          0x7875        Info-ZIP Unix (new)
          0xfb4a        SMS/QDOS

The following are detailed descriptions of the known extra-field block types:

         -OS/2 Extended Attributes Extra Field:
          ====================================

          The following is the layout of the OS/2 extended attributes "extra"
          block.  (Last Revision 19960922)

          Note: all fields stored in Intel low-byte/high-byte order.

          Local-header version:

          Value         Size        Description
          -----         ----        -----------
  (OS/2)  0x0009        Short       tag for this extra block type
          TSize         Short       total data size for this block
          BSize         Long        uncompressed EA data size
          CType         Short       compression type
          EACRC         Long        CRC value for uncompressed EA data
          (var.)        variable    compressed EA data

          Central-header version:
```

```
        Value         Size        Description
        -----         ----        -----------
(OS/2)  0x0009        Short       tag for this extra block type
        TSize         Short       total data size for this block (4)
        BSize         Long        size of uncompressed local EA data

        The value of CType is interpreted according to the "compression
        method" section above; i.e., 0 for stored, 8 for deflated, etc.

        The OS/2 extended attribute structure (FEA2LIST) is compressed and
        then stored in its entirety within this structure.  There will only
        ever be one block of data in the variable-length field.


       -OS/2 Access Control List Extra Field:
        ====================================

        The following is the layout of the OS/2 ACL extra block.
        (Last Revision 19960922)

        Local-header version:

        Value         Size        Description
        -----         ----        -----------
(ACL)   0x4c41        Short       tag for this extra block type ("AL")
        TSize         Short       total data size for this block
        BSize         Long        uncompressed ACL data size
        CType         Short       compression type
        EACRC         Long        CRC value for uncompressed ACL data
        (var.)        variable    compressed ACL data

        Central-header version:

        Value         Size        Description
        -----         ----        -----------
(ACL)   0x4c41        Short       tag for this extra block type ("AL")
        TSize         Short       total data size for this block (4)
        BSize         Long        size of uncompressed local ACL data

        The value of CType is interpreted according to the "compression
        method" section above; i.e., 0 for stored, 8 for deflated, etc.

        The uncompressed ACL data consist of a text header of the form
        "ACL1:%hX,%hd\n", where the first field is the OS/2 ACCINFO acc_attr
        member and the second is acc_count, followed by acc_count strings
        of the form "%s,%hx\n", where the first field is acl_ugname (user
        group name) and the second acl_access.  This block type will be
        extended for other operating systems as needed.
```

-Windows NT Security Descriptor Extra Field:
 ==

 The following is the layout of the NT Security Descriptor (another
 type of ACL) extra block. (Last Revision 19960922)

 Local-header version:

 Value   Size   Description
 -----   ----   -----------
(SD)   0x4453   Short   tag for this extra block type ("SD")
 TSize   Short   total data size for this block
 BSize   Long   uncompressed SD data size
 Version   Byte   version of uncompressed SD data format
 CType   Short   compression type
 EACRC   Long   CRC value for uncompressed SD data
 (var.)   variable   compressed SD data

 Central-header version:

 Value   Size   Description
 -----   ----   -----------
(SD)   0x4453   Short   tag for this extra block type ("SD")
 TSize   Short   total data size for this block (4)
 BSize   Long   size of uncompressed local SD data

 The value of CType is interpreted according to the "compression
 method" section above; i.e., 0 for stored, 8 for deflated, etc.
 Version specifies how the compressed data are to be interpreted
 and allows for future expansion of this extra field type. Currently
 only version 0 is defined.

 For version 0, the compressed data are to be interpreted as a single
 valid Windows NT SECURITY_DESCRIPTOR data structure, in self-relative
 format.

-PKWARE Win95/WinNT Extra Field:
 ==============================

 The following description covers PKWARE's "NTFS" attributes
 "extra" block, introduced with the release of PKZIP 2.50 for
 Windows. (Last Revision 20001118)

 (Note: At this time the Mtime, Atime and Ctime values may
 be used on any WIN32 system.)
[Info-ZIP note: In the current implementations, this field has
 a fixed total data size of 32 bytes and is only stored as local
 extra field.]

 Value   Size   Description

```
        -----         ----        -----------
(NTFS)  0x000a        Short       Tag for this "extra" block type
        TSize         Short       Total Data Size for this block
        Reserved      Long        for future use
        Tag1          Short       NTFS attribute tag value #1
        Size1         Short       Size of attribute #1, in bytes
        (var.)        SubSize1    Attribute #1 data
        .
        .
        .
        TagN          Short       NTFS attribute tag value #N
        SizeN         Short       Size of attribute #N, in bytes
        (var.)        SubSize1    Attribute #N data

        For NTFS, values for Tag1 through TagN are as follows:
        (currently only one set of attributes is defined for NTFS)

        Tag         Size        Description
        -----       ----        -----------
        0x0001      2 bytes     Tag for attribute #1
        Size1       2 bytes     Size of attribute #1, in bytes (24)
        Mtime       8 bytes     64-bit NTFS file last modification time
        Atime       8 bytes     64-bit NTFS file last access time
        Ctime       8 bytes     64-bit NTFS file creation time

        The total length for this block is 28 bytes, resulting in a
        fixed size value of 32 for the TSize field of the NTFS block.

        The NTFS filetimes are 64-bit unsigned integers, stored in Intel
        (least significant byte first) byte order. They determine the
        number of 1.0E-07 seconds (1/10th microseconds!) past WinNT "epoch",
        which is "01-Jan-1601 00:00:00 UTC".


       -PKWARE OpenVMS Extra Field:
        ==========================

        The following is the layout of PKWARE's OpenVMS attributes "extra"
        block.  (Last Revision 12/17/91)

        Note: all fields stored in Intel low-byte/high-byte order.

        Value         Size        Description
        -----         ----        -----------
(VMS)   0x000c        Short       Tag for this "extra" block type
        TSize         Short       Total Data Size for this block
        CRC           Long        32-bit CRC for remainder of the block
        Tag1          Short       OpenVMS attribute tag value #1
        Size1         Short       Size of attribute #1, in bytes
        (var.)        Size1       Attribute #1 data
        .
```

```
        .
        .
        TagN          Short       OpenVMS attribute tage value #N
        SizeN         Short       Size of attribute #N, in bytes
        (var.)        SizeN       Attribute #N data

        Rules:

        1. There will be one or more of attributes present, which
           will each be preceded by the above TagX & SizeX values.
           These values are identical to the ATR$C_XXXX and
           ATR$S_XXXX constants which are defined in ATR.H under
           OpenVMS C.  Neither of these values will ever be zero.

        2. No word alignment or padding is performed.

        3. A well-behaved PKZIP/OpenVMS program should never produce
           more than one sub-block with the same TagX value.  Also,
           there will never be more than one "extra" block of type
           0x000c in a particular directory record.


       -Info-ZIP VMS Extra Field:
        ========================

        The following is the layout of Info-ZIP's VMS attributes extra
        block for VAX or Alpha AXP.  The local-header and central-header
        versions are identical.  (Last Revision 19960922)

        Value         Size        Description
        -----         ----        -----------
(VMS2)  0x4d49        Short       tag for this extra block type ("JM")
        TSize         Short       total data size for this block
        ID            Long        block ID
        Flags         Short       info bytes
        BSize         Short       uncompressed block size
        Reserved      Long        (reserved)
        (var.)        variable    compressed VMS file-attributes block

        The block ID is one of the following unterminated strings:

              "VFAB"          struct FAB
              "VALL"          struct XABALL
              "VFHC"          struct XABFHC
              "VDAT"          struct XABDAT
              "VRDT"          struct XABRDT
              "VPRO"          struct XABPRO
              "VKEY"          struct XABKEY
              "VMSV"          version (e.g., "V6.1"; truncated at hyphen)
              "VNAM"          reserved
```

The lower three bits of Flags indicate the compression method. The currently defined methods are:

```
      0       stored (not compressed)
      1       simple "RLE"
      2       deflated
```

The "RLE" method simply replaces zero-valued bytes with zero-valued bits and non-zero-valued bytes with a "1" bit followed by the byte value.

The variable-length compressed data contains only the data corresponding to the indicated structure or string. Typically multiple VMS2 extra fields are present (each with a unique block type).

-Info-ZIP Macintosh Extra Field:

The following is the layout of the (old) Info-ZIP resource-fork extra block for Macintosh. The local-header and central-header versions are identical. (Last Revision 19960922)

	Value	Size	Description
(Mac)	0x07c8	Short	tag for this extra block type
	TSize	Short	total data size for this block
	"JLEE"	beLong	extra-field signature
	FInfo	16 bytes	Macintosh FInfo structure
	CrDat	beLong	HParamBlockRec fileParam.ioFlCrDat
	MdDat	beLong	HParamBlockRec fileParam.ioFlMdDat
	Flags	beLong	info bits
	DirID	beLong	HParamBlockRec fileParam.ioDirID
	VolName	28 bytes	volume name (optional)

All fields but the first two are in native Macintosh format (big-endian Motorola order, not little-endian Intel). The least significant bit of Flags is 1 if the file is a data fork, 0 otherwise. In addition, if this extra field is present, the filename has an extra 'd' or 'r' appended to indicate data fork or resource fork. The 28-byte VolName field may be omitted.

-ZipIt Macintosh Extra Field (long):

The following is the layout of the ZipIt extra block for Macintosh. The local-header and central-header versions are identical. (Last Revision 19970130)

Value	Size	Description

```
        -----         ----        -----------
(Mac2)  0x2605        Short       tag for this extra block type
        TSize         Short       total data size for this block
        "ZPIT"        beLong      extra-field signature
        FnLen         Byte        length of FileName
        FileName      variable    full Macintosh filename
        FileType      Byte[4]     four-byte Mac file type string
        Creator       Byte[4]     four-byte Mac creator string


       -ZipIt Macintosh Extra Field (short):
        ====================================

        The following is the layout of a shortened variant of the
        ZipIt extra block for Macintosh (without "full name" entry).
        This variant is used by ZipIt 1.3.5 and newer for entries that
        do not need a "full Mac filename" record.
        The local-header and central-header versions are identical.
        (Last Revision 19980903)

        Value         Size        Description
        -----         ----        -----------
(Mac2b) 0x2705        Short       tag for this extra block type
        TSize         Short       total data size for this block (12)
        "ZPIT"        beLong      extra-field signature
        FileType      Byte[4]     four-byte Mac file type string
        Creator       Byte[4]     four-byte Mac creator string


       -Info-ZIP Macintosh Extra Field (new):
        =====================================

        The following is the layout of the (new) Info-ZIP extra
        block for Macintosh, designed by Dirk Haase.
        All values are in little-endian.
        (Last Revision 19981005)

        Local-header version:

        Value         Size        Description
        -----         ----        -----------
(Mac3)  0x334d        Short       tag for this extra block type ("M3")
        TSize         Short       total data size for this block
        BSize         Long        uncompressed finder attribute data size
        Flags         Short       info bits
        fdType        Byte[4]     Type of the File (4-byte string)
        fdCreator     Byte[4]     Creator of the File (4-byte string)
        (CType)       Short       compression type
        (CRC)         Long        CRC value for uncompressed MacOS data
        Attribs       variable    finder attribute data (see below)
```

Central-header version:

	Value	Size	Description
(Mac3)	0x334d	Short	tag for this extra block type ("M3")
	TSize	Short	total data size for this block
	BSize	Long	uncompressed finder attribute data size
	Flags	Short	info bits
	fdType	Byte[4]	Type of the File (4-byte string)
	fdCreator	Byte[4]	Creator of the File (4-byte string)

The third bit of Flags in both headers indicates whether
the LOCAL extra field is uncompressed (and therefore whether CType
and CRC are omitted):

Bits of the Flags:

bit 0	if set, file is a data fork; otherwise unset
bit 1	if set, filename will be not changed
bit 2	if set, Attribs is uncompressed (no CType, CRC)
bit 3	if set, date and times are in 64 bit if zero date and times are in 32 bit.
bit 4	if set, timezone offsets fields for the native Mac times are omitted (UTC support deactivated)
bits 5-15	reserved;

Attributes:

Attribs is a Mac-specific block of data in little-endian format with
the following structure (if compressed, uncompress it first):

Value	Size	Description
fdFlags	Short	Finder Flags
fdLocation.v	Short	Finder Icon Location
fdLocation.h	Short	Finder Icon Location
fdFldr	Short	Folder containing file
FXInfo	16 bytes	Macintosh FXInfo structure

FXInfo-Structure:

fdIconID	Short	
fdUnused[3]	Short	unused but reserved 6 bytes
fdScript	Byte	Script flag and number
fdXFlags	Byte	More flag bits
fdComment	Short	Comment ID
fdPutAway	Long	Home Dir ID

FVersNum	Byte	file version number may be not used by MacOS
ACUser	Byte	directory access rights

```
FlCrDat       ULong       date and time of creation
FlMdDat       ULong       date and time of last modification
FlBkDat       ULong       date and time of last backup
  These time numbers are original Mac FileTime values (local time!).
  Currently, date-time width is 32-bit, but future version may
  support be 64-bit times (see flags)

CrGMTOffs     Long(signed!)   difference "local Creat. time - UTC"
MdGMTOffs     Long(signed!)   difference "local Modif. time - UTC"
BkGMTOffs     Long(signed!)   difference "local Backup time - UTC"
  These "local time - UTC" differences (stored in seconds) may be
  used to support timestamp adjustment after inter-timezone transfer.
  These fields are optional; bit 4 of the flags word controls their
  presence.

Charset       Short       TextEncodingBase (Charset)
                          valid for the following two fields

FullPath      variable    Path of the current file.
                          Zero terminated string (C-String)
                          Currently coded in the native Charset.

Comment       variable    Finder Comment of the current file.
                          Zero terminated string (C-String)
                          Currently coded in the native Charset.


-SmartZIP Macintosh Extra Field:
 ===================================

 The following is the layout of the SmartZIP extra
 block for Macintosh, designed by Marco Bambini.

 Local-header version:

 Value         Size        Description
 -----         ----        -----------
 0x4d63        Short       tag for this extra block type ("cM")
 TSize         Short       total data size for this block (64)
 "dZip"        beLong      extra-field signature
 fdType        Byte[4]     Type of the File (4-byte string)
 fdCreator     Byte[4]     Creator of the File (4-byte string)
 fdFlags       beShort     Finder Flags
 fdLocation.v  beShort     Finder Icon Location
 fdLocation.h  beShort     Finder Icon Location
 fdFldr        beShort     Folder containing file
 CrDat         beLong      HParamBlockRec fileParam.ioFlCrDat
 MdDat         beLong      HParamBlockRec fileParam.ioFlMdDat
 frScroll.v    Byte        vertical pos. of folder's scroll bar
 fdScript      Byte        Script flag and number
```

```
        frScroll.h    Byte        horizontal pos. of folder's scroll bar
        fdXFlags      Byte        More flag bits
        FileName      Byte[32]    full Macintosh filename (pascal string)

        All fields but the first two are in native Macintosh format
        (big-endian Motorola order, not little-endian Intel).
        The extra field size is fixed to 64 bytes.
        The local-header and central-header versions are identical.


       -Acorn SparkFS Extra Field:
        ========================

        The following is the layout of David Pilling's SparkFS extra block
        for Acorn RISC OS.  The local-header and central-header versions are
        identical.  (Last Revision 19960922)

        Value         Size        Description
        -----         ----        -----------
 (Acorn) 0x4341        Short       tag for this extra block type ("AC")
        TSize         Short       total data size for this block (20)
        "ARC0"        Long        extra-field signature
        LoadAddr      Long        load address or file type
        ExecAddr      Long        exec address
        Attr          Long        file permissions
        Zero          Long        reserved; always zero

        The following bits of Attr are associated with the given file
        permissions:

              bit 0           user-writable ('W')
              bit 1           user-readable ('R')
              bit 2           reserved
              bit 3           locked ('L')
              bit 4           publicly writable ('w')
              bit 5           publicly readable ('r')
              bit 6           reserved
              bit 7           reserved


       -VM/CMS Extra Field:
        ==================

        The following is the layout of the file-attributes extra block for
        VM/CMS.  The local-header and central-header versions are
        identical.  (Last Revision 19960922)

        Value         Size        Description
        -----         ----        -----------
(VM/CMS) 0x4704        Short       tag for this extra block type
        TSize         Short       total data size for this block
```

```
          flData        variable    file attributes data

          flData is an uncompressed fldata_t struct.


         -MVS Extra Field:
          ===============

          The following is the layout of the file-attributes extra block for
          MVS.  The local-header and central-header versions are identical.
          (Last Revision 19960922)

          Value         Size        Description
          -----         ----        -----------
  (MVS)   0x470f        Short       tag for this extra block type
          TSize         Short       total data size for this block
          flData        variable    file attributes data

          flData is an uncompressed fldata_t struct.


         -PKWARE Unix Extra Field:
          ========================

          The following is the layout of PKWARE's Unix "extra" block.
          It was introduced with the release of PKZIP for Unix 2.50.
          Note: all fields are stored in Intel low-byte/high-byte order.
          (Last Revision 19980901)

          This field has a minimum data size of 12 bytes and is only stored
          as local extra field.

          Value         Size        Description
          -----         ----        -----------
  (Unix0) 0x000d        Short       Tag for this "extra" block type
          TSize         Short       Total Data Size for this block
          AcTime        Long        time of last access (UTC/GMT)
          ModTime       Long        time of last modification (UTC/GMT)
          UID           Short       Unix user ID
          GID           Short       Unix group ID
          (var)         variable    Variable length data field

          The variable length data field will contain file type
          specific data.  Currently the only values allowed are
          the original "linked to" file names for hard or symbolic
          links, and the major and minor device node numbers for
          character and block device nodes.  Since device nodes
          cannot be either symbolic or hard links, only one set of
          variable length data is stored.  Link files will have the
          name of the original file stored.  This name is NOT NULL
          terminated.  Its size can be determined by checking TSize -
```

12. Device entries will have eight bytes stored as two 4
byte entries (in little-endian format). The first entry
will be the major device number, and the second the minor
device number.

[Info-ZIP note: The fixed part of this field has the same layout as
Info-ZIP's abandoned "Unix1 timestamps & owner ID info" extra field;
only the two tag bytes are different.]

-PATCH Descriptor Extra Field:
==========================

The following is the layout of the Patch Descriptor "extra"
block.

Note: all fields stored in Intel low-byte/high-byte order.

	Value	Size	Description
(Patch)	0x000f	Short	Tag for this "extra" block type
	TSize	Short	Size of the total "extra" block
	Version	Short	Version of the descriptor
	Flags	Long	Actions and reactions (see below)
	OldSize	Long	Size of the file about to be patched
	OldCRC	Long	32-bit CRC of the file about to be patched
	NewSize	Long	Size of the resulting file
	NewCRC	Long	32-bit CRC of the resulting file

Actions and reactions

Bits	Description
0	Use for autodetection
1	Treat as selfpatch
2-3	RESERVED
4-5	Action (see below)
6-7	RESERVED
8-9	Reaction (see below) to absent file
10-11	Reaction (see below) to newer file
12-13	Reaction (see below) to unknown file
14-15	RESERVED
16-31	RESERVED

Actions

Action	Value
none	0
add	1

```
        delete       2
        patch        3

        Reactions

        Reaction     Value
        --------     -----
        ask          0
        skip         1
        ignore       2
        fail         3


       -PKCS#7 Store for X.509 Certificates:
        ==================================

        This field is contains the information about each
        certificate a file is signed with.  This field should only
        appear in the first central directory record, and will be
        ignored in any other record.

        Note: all fields stored in Intel low-byte/high-byte order.

        Value         Size        Description
        -----         ----        -----------
(Store) 0x0014        2 bytes     Tag for this "extra" block type
        SSize         2 bytes     Size of the store data
        SData         (variable)  Data about the store

        SData
        Value         Size        Description
        -----         ----        -----------
        Version       2 bytes     Version number, 0x0001 for now
        StoreD        (variable)  Actual store data

        The StoreD member is suitable for passing as the pbData
        member of a CRYPT_DATA_BLOB to the CertOpenStore() function
        in Microsoft's CryptoAPI.  The SSize member above will be
        cbData + 6, where cbData is the cbData member of the same
        CRYPT_DATA_BLOB.  The encoding type to pass to
        CertOpenStore() should be
        PKCS_7_ANS_ENCODING | X509_ASN_ENCODING.


       -X.509 Certificate ID and Signature for individual file:
        ======================================================

        This field contains the information about which certificate
        in the PKCS#7 Store was used to sign the particular file.
        It also contains the signature data.  This field can appear
        multiple times, but can only appear once per certificate.
```

Note: all fields stored in Intel low-byte/high-byte order.

	Value	Size	Description
(CID)	0x0015	2 bytes	Tag for this "extra" block type
	CSize	2 bytes	Size of Method
	Method	(variable)	

Method

Value	Size	Description
Version	2 bytes	Version number, for now 0x0001
AlgID	2 bytes	Algorithm ID used for signing
IDSize	2 bytes	Size of Certificate ID data
CertID	(variable)	Certificate ID data
SigSize	2 bytes	Size of Signature data
Sig	(variable)	Signature data

CertID

Value	Size	Description
Size1	4 bytes	Size of CertID, should be (IDSize - 4)
Size1	4 bytes	A bug in version one causes this value to appear twice.
IssSize	4 bytes	Issuer data size
Issuer	(variable)	Issuer data
SerSize	4 bytes	Serial Number size
Serial	(variable)	Serial Number data

The Issuer and IssSize members are suitable for creating a CRYPT_DATA_BLOB to be the Issuer member of a CERT_INFO struct. The Serial and SerSize members would be the SerialNumber member of the same CERT_INFO struct. This struct would be used to find the certificate in the store the file was signed with. Those structures are from the MS CryptoAPI.

Sig and SigSize are the actual signature data and size generated by signing the file with the MS CryptoAPI using a hash created with the given AlgID.

-X.509 Certificate ID and Signature for central directory:
==========

This field contains the information about which certificate in the PKCS#7 Store was used to sign the central directory. It should only appear with the first central directory record, along with the store. The data structure is the same as the CID, except that SigSize will be 0, and there

will be no Sig member.

This field is also kept after the last central directory record, as the signature data (ID 0x05054b50, it looks like a central directory record of a different type). This second copy of the data is the Signature Data member of the record, and will have a SigSize that is non-zero, and will have Sig data.

Note: all fields stored in Intel low-byte/high-byte order.

	Value	Size	Description
(CDID)	0x0016	2 bytes	Tag for this "extra" block type
	CSize	2 bytes	Size of Method
	Method	(variable)	

-ZIP64 Extended Information Extra Field:
==

The following is the layout of the ZIP64 extended information "extra" block. If one of the size or offset fields in the Local or Central directory record is too small to hold the required data, a ZIP64 extended information record is created. The order of the fields in the ZIP64 extended information record is fixed, but the fields will only appear if the corresponding Local or Central directory record field is set to 0xFFFF or 0xFFFFFFFF.

Note: all fields stored in Intel low-byte/high-byte order.

	Value	Size	Description
(ZIP64)	0x0001	2 bytes	Tag for this "extra" block type
	Size	2 bytes	Size of this "extra" block
	Original Size	8 bytes	Original uncompresseed file size
	Compressed Size	8 bytes	Size of compressed data
	Relative Header Offset	8 bytes	Offset of local header record
	Disk Start Number	4 bytes	Number of the disk on which this file starts

This entry in the Local header must include BOTH original and compressed file sizes.

-Extended Timestamp Extra Field:
 ==============================

 The following is the layout of the extended-timestamp extra block.
 (Last Revision 19970118)

 Local-header version:

	Value	Size	Description
(time)	0x5455	Short	tag for this extra block type ("UT")
	TSize	Short	total data size for this block
	Flags	Byte	info bits
	(ModTime)	Long	time of last modification (UTC/GMT)
	(AcTime)	Long	time of last access (UTC/GMT)
	(CrTime)	Long	time of original creation (UTC/GMT)

 Central-header version:

	Value	Size	Description
(time)	0x5455	Short	tag for this extra block type ("UT")
	TSize	Short	total data size for this block
	Flags	Byte	info bits (refers to local header!)
	(ModTime)	Long	time of last modification (UTC/GMT)

 The central-header extra field contains the modification time only,
 or no timestamp at all. TSize is used to flag its presence or
 absence. But note:

     If "Flags" indicates that Modtime is present in the local header
     field, it MUST be present in the central header field, too!
     This correspondence is required because the modification time
     value may be used to support trans-timezone freshening and
     updating operations with zip archives.

 The time values are in standard Unix signed-long format, indicating
 the number of seconds since 1 January 1970 00:00:00. The times
 are relative to Coordinated Universal Time (UTC), also sometimes
 referred to as Greenwich Mean Time (GMT). To convert to local time,
 the software must know the local timezone offset from UTC/GMT.

 The lower three bits of Flags in both headers indicate which time-
 stamps are present in the LOCAL extra field:

       bit 0   if set, modification time is present
       bit 1   if set, access time is present
       bit 2   if set, creation time is present
       bits 3-7   reserved for additional timestamps; not set

 Those times that are present will appear in the order indicated, but

any combination of times may be omitted. (Creation time may be present without access time, for example.) TSize should equal (1 + 4*(number of set bits in Flags)), as the block is currently defined. Other timestamps may be added in the future.

-Info-ZIP Unix Extra Field (type 1):
==========

The following is the layout of the old Info-ZIP extra block for Unix. It has been replaced by the extended-timestamp extra block (0x5455) and the Unix type 2 extra block (0x7855).
(Last Revision 19970118)

Local-header version:

	Value	Size	Description
(Unix1)	0x5855	Short	tag for this extra block type ("UX")
	TSize	Short	total data size for this block
	AcTime	Long	time of last access (UTC/GMT)
	ModTime	Long	time of last modification (UTC/GMT)
	UID	Short	Unix user ID (optional)
	GID	Short	Unix group ID (optional)

Central-header version:

	Value	Size	Description
(Unix1)	0x5855	Short	tag for this extra block type ("UX")
	TSize	Short	total data size for this block
	AcTime	Long	time of last access (GMT/UTC)
	ModTime	Long	time of last modification (GMT/UTC)

The file access and modification times are in standard Unix signed-long format, indicating the number of seconds since 1 January 1970 00:00:00. The times are relative to Coordinated Universal Time (UTC), also sometimes referred to as Greenwich Mean Time (GMT). To convert to local time, the software must know the local timezone offset from UTC/GMT. The modification time may be used by non-Unix systems to support inter-timezone freshening and updating of zip archives.

The local-header extra block may optionally contain UID and GID info for the file. The local-header TSize value is the only indication of this. Note that Unix UIDs and GIDs are usually specific to a particular machine, and they generally require root access to restore.

This extra field type is obsolete, but it has been in use since mid-1994. Therefore future archiving software should continue to

support it. Some guidelines:

    An archive member should either contain the old "Unix1"
    extra field block or the new extra field types "time" and/or
    "Unix2".

    If both the old "Unix1" block type and one or both of the new
    block types "time" and "Unix2" are found, the "Unix1" block
    should be considered invalid and ignored.

    Unarchiving software should recognize both old and new extra
    field block types, but the info from new types overrides the
    old "Unix1" field.

    Archiving software should recognize "Unix1" extra fields for
    timestamp comparison but never create it for updated, freshened
    or new archive members. When copying existing members to a new
    archive, any "Unix1" extra field blocks should be converted to
    the new "time" and/or "Unix2" types.

-Info-ZIP Unix Extra Field (type 2):
 ==================================

 The following is the layout of the new Info-ZIP extra block for
 Unix. (Last Revision 19960922)

 Local-header version:

 Value   Size   Description
 -----   ----   -----------
(Unix2) 0x7855   Short   tag for this extra block type ("Ux")
 TSize   Short   total data size for this block (4)
 UID   Short   Unix user ID
 GID   Short   Unix group ID

 Central-header version:

 Value   Size   Description
 -----   ----   -----------
(Unix2) 0x7855   Short   tag for this extra block type ("Ux")
 TSize   Short   total data size for this block (0)

 The data size of the central-header version is zero; it is used
 solely as a flag that UID/GID info is present in the local-header
 extra field. If additional fields are ever added to the local
 version, the central version may be extended to indicate this.

 Note that Unix UIDs and GIDs are usually specific to a particular
 machine, and they generally require root access to restore.

-ASi Unix Extra Field:
 ====================

 The following is the layout of the ASi extra block for Unix. The
 local-header and central-header versions are identical.
 (Last Revision 19960916)

 Value   Size   Description
 -----   ----   -----------
(Unix3) 0x756e   Short   tag for this extra block type ("nu")
 TSize   Short   total data size for this block
 CRC   Long   CRC-32 of the remaining data
 Mode   Short   file permissions
 SizDev   Long   symlink'd size OR major/minor dev num
 UID   Short   user ID
 GID   Short   group ID
 (var.)   variable   symbolic link filename

 Mode is the standard Unix st_mode field from struct stat, containing
 user/group/other permissions, setuid/setgid and symlink info, etc.

 If Mode indicates that this file is a symbolic link, SizDev is the
 size of the file to which the link points. Otherwise, if the file
 is a device, SizDev contains the standard Unix st_rdev field from
 struct stat (includes the major and minor numbers of the device).
 SizDev is undefined in other cases.

 If Mode indicates that the file is a symbolic link, the final field
 will be the name of the file to which the link points. The file-
 name length can be inferred from TSize.

 [Note that TSize may incorrectly refer to the data size not counting
  the CRC; i.e., it may be four bytes too small.]

-BeOS Extra Field:
 ================

 The following is the layout of the file-attributes extra block for
 BeOS. (Last Revision 19970531)

 Local-header version:

 Value   Size   Description
 -----   ----   -----------
(BeOS) 0x6542   Short   tag for this extra block type ("Be")
 TSize   Short   total data size for this block
 BSize   Long   uncompressed file attribute data size
 Flags   Byte   info bits
 (CType)   Short   compression type

```
        (CRC)         Long        CRC value for uncompressed file attribs
        Attribs       variable    file attribute data

        Central-header version:

        Value         Size        Description
        -----         ----        -----------
(BeOS)  0x6542        Short       tag for this extra block type ("Be")
        TSize         Short       total data size for this block (5)
        BSize         Long        size of uncompr. local EF block data
        Flags         Byte        info bits
```

The least significant bit of Flags in both headers indicates whether the LOCAL extra field is uncompressed (and therefore whether CType and CRC are omitted):

```
      bit 0           if set, Attribs is uncompressed (no CType, CRC)
      bits 1-7        reserved; if set, assume error or unknown data
```

Currently the only supported compression types are deflated (type 8) and stored (type 0); the latter is not used by Info-ZIP's Zip but is supported by UnZip.

Attribs is a BeOS-specific block of data in big-endian format with the following structure (if compressed, uncompress it first):

```
    Value     Size        Description
    -----     ----        -----------
    Name      variable    attribute name (null-terminated string)
    Type      Long        attribute type (32-bit unsigned integer)
    Size      Long Long   data size for this sub-block (64 bits)
    Data      variable    attribute data
```

The attribute structure is repeated for every attribute. The Data field may contain anything--text, flags, bitmaps, etc.

-SMS/QDOS Extra Field:

The following is the layout of the file-attributes extra block for SMS/QDOS. The local-header and central-header versions are identical. (Last Revision 19960929)

```
        Value         Size        Description
        -----         ----        -----------
(QDOS)  0xfb4a        Short       tag for this extra block type
        TSize         Short       total data size for this block
        LongID        Long        extra-field signature
        (ExtraID)     Long        additional signature/flag bytes
        QDirect       64 bytes    qdirect structure
```

LongID may be "QZHD" or "QDOS". In the latter case, ExtraID will be present. Its first three bytes are "02\0"; the last byte is currently undefined.

QDirect contains the file's uncompressed directory info (qdirect struct). Its elements are in native (big-endian) format:

d_length	beLong	file length
d_access	byte	file access type
d_type	byte	file type
d_datalen	beLong	data length
d_reserved	beLong	unused
d_szname	beShort	size of filename
d_name	36 bytes	filename
d_update	beLong	time of last update
d_refdate	beLong	file version number
d_backup	beLong	time of last backup (archive date)

-AOS/VS Extra Field:
==========

The following is the layout of the extra block for Data General AOS/VS. The local-header and central-header versions are identical. (Last Revision 19961125)

	Value	Size	Description
(AOSVS)	0x5356	Short	tag for this extra block type ("VS")
	TSize	Short	total data size for this block
	"FCI\0"	Long	extra-field signature
	Version	Byte	version of AOS/VS extra block (10 = 1.0)
	Fstat	variable	fstat packet
	AclBuf	variable	raw ACL data ($MXACL bytes)

Fstat contains the file's uncompressed fstat packet, which is one of the following:

normal fstat packet	(P_FSTAT struct)
DIR/CPD fstat packet	(P_FSTAT_DIR struct)
unit (device) fstat packet	(P_FSTAT_UNIT struct)
IPC file fstat packet	(P_FSTAT_IPC struct)

AclBuf contains the raw ACL data; its length is $MXACL.

-Tandem NSK Extra Field:
==========

The following is the layout of the file-attributes extra block for

Tandem NSK. The local-header and central-header versions are identical. (Last Revision 19981221)

	Value	Size	Description
(TA)	0x4154	Short	tag for this extra block type ("TA")
	TSize	Short	total data size for this block (20)
	NSKattrs	20 Bytes	NSK attributes

-THEOS Extra Field:

The following is the layout of the file-attributes extra block for Theos. The local-header and central-header versions are identical. (Last Revision 19990206)

	Value	Size	Description
(Theos)	0x6854	Short	'Th' signature
	size	Short	size of extra block
	flags	Byte	reserved for future use
	filesize	Long	file size
	fileorg	Byte	type of file (see below)
	keylen	Short	key length for indexed and keyed files, data segment size for 16 bits programs
	reclen	Short	record length for indexed,keyed and direct, text segment size for 16 bits programs
	filegrow	Byte	growing factor for indexed,keyed and direct
	protect	Byte	protections (see below)
	reserved	Short	reserved for future use

File types

0x80 library (keyed access list of files)
0x40 directory
0x10 stream file
0x08 direct file
0x04 keyed file
0x02 indexed file
0x0e reserved
0x01 16 bits real mode program (obsolete)
0x21 16 bits protected mode program
0x41 32 bits protected mode program

Protection codes

User protection

```
          0x01  non readable
          0x02  non writable
          0x04  non executable
          0x08  non erasable

          Other protection
          ----------------
          0x10  non readable
          0x20  non writable
          0x40  non executable Theos before 4.0
          0x40  modified Theos 4.x
          0x80  not hidden


        -THEOS old inofficial Extra Field:
         ================================

         The following is the layout of an inoffical former version of a
         Theos file-attributes extra blocks.  This layout was never published
         and is no longer created. However, UnZip can optionally support it
         when compiling with the option flag OLD_THEOS_EXTRA defined.
         Both the local-header and central-header versions are identical.
         (Last Revision 19990206)

         Value         Size        Description
         -----         ----        -----------
(THS0)   0x4854        Short       'TH' signature
         size          Short       size of extra block
         flags         Short       reserved for future use
         filesize      Long        file size
         reclen        Short       record length for indexed,keyed and direct,
                                   text segment size for 16 bits programs
         keylen        Short       key length for indexed and keyed files,
                                   data segment size for 16 bits programs
         filegrow      Byte        growing factor for indexed,keyed and direct
         reserved      3 Bytes     reserved for future use


        -FWKCS MD5 Extra Field:
         =====================

         The FWKCS Contents_Signature System, used in automatically
         identifying files independent of filename, optionally adds
         and uses an extra field to support the rapid creation of
         an enhanced contents_signature.
         There is no local-header version; the following applies
         only to the central header.  (Last Revision 19961207)

         Central-header version:

         Value         Size        Description
```

```
        -----         ----        -----------
(MD5)   0x4b46        Short       tag for this extra block type ("FK")
        TSize         Short       total data size for this block (19)
        "MD5"         3 bytes     extra-field signature
        MD5hash       16 bytes    128-bit MD5 hash of uncompressed data
                                  (low byte first)

        When FWKCS revises a .ZIP file central directory to add
        this extra field for a file, it also replaces the
        central directory entry for that file's uncompressed
        file length with a measured value.

        FWKCS provides an option to strip this extra field, if
        present, from a .ZIP file central directory. In adding
        this extra field, FWKCS preserves .ZIP file Authenticity
        Verification; if stripping this extra field, FWKCS
        preserves all versions of AV through PKZIP version 2.04g.

        FWKCS, and FWKCS Contents_Signature System, are
        trademarks of Frederick W. Kantor.

        (1) R. Rivest, RFC1321.TXT, MIT Laboratory for Computer
            Science and RSA Data Security, Inc., April 1992.
            ll.76-77: "The MD5 algorithm is being placed in the
            public domain for review and possible adoption as a
            standard."


       -Info-ZIP Unicode Path Extra Field:
        ==================================

        Stores the UTF-8 version of the entry path as stored in the
        local header and central directory header.
        (Last Revision 20070912)

        Value         Size        Description
        -----         ----        -----------
(UPath) 0x7075        Short       tag for this extra block type ("up")
        TSize         Short       total data size for this block
        Version       1 byte      version of this extra field, currently 1
        NameCRC32     4 bytes     File Name Field CRC32 Checksum
        UnicodeName   Variable    UTF-8 version of the entry File Name

        Currently Version is set to the number 1.  If there is a need
        to change this field, the version will be incremented.  Changes
        may not be backward compatible so this extra field should not be
        used if the version is not recognized.

        The NameCRC32 is the standard zip CRC32 checksum of the File Name
        field in the header.  This is used to verify that the header
        File Name field has not changed since the Unicode Path extra field
```

was created. This can happen if a utility renames the entry but does not update the UTF-8 path extra field. If the CRC check fails, this UTF-8 Path Extra Field should be ignored and the File Name field in the header used instead.

The UnicodeName is the UTF-8 version of the contents of the File Name field in the header. As UnicodeName is defined to be UTF-8, no UTF-8 byte order mark (BOM) is used. The length of this field is determined by subtracting the size of the previous fields from TSize. If both the File Name and Comment fields are UTF-8, the new General Purpose Bit Flag, bit 11 (Language encoding flag (EFS)), can be used to indicate that both the header File Name and Comment fields are UTF-8 and, in this case, the Unicode Path and Unicode Comment extra fields are not needed and should not be created. Note that, for backward compatibility, bit 11 should only be used if the native character set of the paths and comments being zipped up are already in UTF-8. The same method, either bit 11 or extra fields, should be used in both the local and central directory headers.

-Info-ZIP Unicode Comment Extra Field:
=====================================

Stores the UTF-8 version of the entry comment as stored in the central directory header.
(Last Revision 20070912)

	Value	Size	Description
(UCom)	0x6375	Short	tag for this extra block type ("uc")
	TSize	Short	total data size for this block
	Version	1 byte	version of this extra field, currently 1
	ComCRC32	4 bytes	Comment Field CRC32 Checksum
	UnicodeCom	Variable	UTF-8 version of the entry comment

Currently Version is set to the number 1. If there is a need to change this field, the version will be incremented. Changes may not be backward compatible so this extra field should not be used if the version is not recognized.

The ComCRC32 is the standard zip CRC32 checksum of the Comment field in the central directory header. This is used to verify that the comment field has not changed since the Unicode Comment extra field was created. This can happen if a utility changes the Comment field but does not update the UTF-8 Comment extra field. If the CRC check fails, this Unicode Comment extra field should be ignored and the Comment field in the header used.

The UnicodeCom field is the UTF-8 version of the entry comment field in the header. As UnicodeCom is defined to be UTF-8, no UTF-8 byte order mark (BOM) is used. The length of this field is determined by

subtracting the size of the previous fields from TSize. If both the File Name and Comment fields are UTF-8, the new General Purpose Bit Flag, bit 11 (Language encoding flag (EFS)), can be used to indicate both the header File Name and Comment fields are UTF-8 and, in this case, the Unicode Path and Unicode Comment extra fields are not needed and should not be created. Note that, for backward compatibility, bit 11 should only be used if the native character set of the paths and comments being zipped up are already in UTF-8. The same method, either bit 11 or extra fields, should be used in both the local and central directory headers.

-Info-ZIP New Unix Extra Field:
==========

Currently stores Unix UIDs/GIDs up to 32 bits.
(Last Revision 20080509)

	Value	Size	Description
(UnixN)	0x7875	Short	tag for this extra block type ("ux")
	TSize	Short	total data size for this block
	Version	1 byte	version of this extra field, currently 1
	UIDSize	1 byte	Size of UID field
	UID	Variable	UID for this entry
	GIDSize	1 byte	Size of GID field
	GID	Variable	GID for this entry

Currently Version is set to the number 1. If there is a need to change this field, the version will be incremented. Changes may not be backward compatible so this extra field should not be used if the version is not recognized.

UIDSize is the size of the UID field in bytes. This size should match the size of the UID field on the target OS.

UID is the UID for this entry in standard little endian format.

GIDSize is the size of the GID field in bytes. This size should match the size of the GID field on the target OS.

GID is the GID for this entry in standard little endian format.

If both the old 16-bit Unix extra field (tag 0x7855, Info-ZIP Unix) and this extra field are present, the values in this extra field supercede the values in that extra field.