exFAT file system specification

08/26/2019 • 106 minutes to read •

In this article

1 Introduction

The exFAT file system is the successor to FAT32 in the FAT family of file systems. This specification describes the exFAT file system and provides all the information necessary for implementing the exFAT file system.

1.1 Design Goals

The exFAT file system has three central design goals (see list below).

1. *Retain the simplicity of FAT-based file systems.*

> Two of the strengths of FAT-based file systems are their relative simplicity and ease of implementation. In the spirit of its predecessors, implementers should find exFAT relatively simple and easy to implement.

2. *Enable very large files and storage devices.*

> The exFAT file system uses 64 bits to describe file size, thereby enabling applications which depend on very large files. The exFAT file system also allows for clusters as large

> as 32MB, effectively enabling very large storage devices.

3. *Incorporate extensibility for future innovation.*

> The exFAT file system incorporates extensibility into its design, enabling the file system to keep pace with innovations in storage and changes in usage.

1.2 Specific Terminology

In the context of this specification, certain terms (see Table 1) carry specific meaning for the design and implementation of the exFAT file system.

Table 1 Definition of Terms Which Carry Very Specific Meaning

Term	Definition
Shall	This specification uses the term “shall” to describe a behavior which is mandatory.
Should	This specification uses the term “should” to describe a behavior which it strongly recommends, but does not make mandatory.
May	This specification uses the term “may” to describe a behavior which is optional.
Mandatory	This term describes a field or structure which an implementation shall modify and shall interpret as this specification describes.
Optional	This term describes a field or structure which an implementation may or may not support. If an implementation supports a given optional field or structure, it shall modify and shall interpret the field or structure as this specification describes.
Undefined	This term describes field or structure contents which an implementation may modify as necessary (i.e. clear to zero when setting surrounding fields or structures) and shall not interpret to hold any specific meaning.
Reserved	This term describes field or structure contents which implementations: 1. Shall initialize to zero and should not use for any purpose 2. Should not interpret, except when computing checksums 3. Shall preserve across operations which modify surrounding fields or structures

1.3 Full Text of Common Acronyms

This specification uses acronyms in common use in the personal computer industry (see Table 2).

Table 2 Full Text of Common Acronyms

Acronym	Full Text
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input Output System
CPU	Central Processing Unit
exFAT	extensible File Allocation Table
FAT	File Allocation Table
FAT12	File Allocation Table, 12-bit cluster indices
FAT16	File Allocation Table, 16-bit cluster indices
FAT32	File Allocation Table, 32-bit cluster indices
GPT	GUID Partition Table
GUID	Globally Unique Identifier (see Section 10.1)
INT	Interrupt
MBR	Master Boot Record
texFAT	Transaction-safe exFAT
UTC	Coordinated Universal Time

1.4 Default Field and Structure Qualifiers

Fields and structures in this specification have the following qualifiers (see list below), unless the specification notes otherwise.

1. Are unsigned

2. Use decimal notation to describe values, where not otherwise noted; this specification uses the post-fix letter "h" to denote hexadecimal numbers and encloses GUIDs in curly braces
3. Are in little-endian format
4. Do not require a null-terminating character for strings

1.5 Windows CE and TexFAT

TexFAT is an extension to exFAT that adds transaction-safe operational semantics on top of the base file system. TexFAT is used by Windows CE. TexFAT requires the use of the two FATs and allocation bitmaps for use in transactions. It also defines several additional structures including padding descriptors and security descriptors.

2 Volume Structure

A volume is the set of all file system structures and data space necessary to store and retrieve user data. All exFAT volumes contain four regions (see Table 3).

Table 3 Volume Structure

Sub-region Name	Offset (sector)	Size (sectors)	Comments
Main Boot Region			
Main Boot Sector	0	1	This sub-region is mandatory and Section 3.1 defines its contents.
Main Extended Boot Sectors	1	8	This sub-region is mandatory and Section 3.2 defines its contents.
Main OEM Parameters	9	1	This sub-region is mandatory and Section 3.3 defines its contents.
Main Reserved	10	1	This sub-region is mandatory and its contents are reserved.

Sub-region Name	**Offset** **(sector)**	**Size** **(sectors)**	**Comments**
Main Boot Checksum	11	1	This sub-region is mandatory and Section 3.4 defines its contents.
Backup Boot Region			
Backup Boot Sector	12	1	This sub-region is mandatory and Section 3.1 defines its contents.
Backup Extended Boot Sectors	13	8	This sub-region is mandatory and Section 3.2 defines its contents.
Backup OEM Parameters	21	1	This sub-region is mandatory and Section 3.3 defines its contents.
Backup Reserved	22	1	This sub-region is mandatory and its contents are reserved.
Backup Boot Checksum	23	1	This sub-region is mandatory and Section 3.4 defines its contents.
FAT Region			
FAT Alignment	24	FatOffset – 24	This sub-region is mandatory and its contents, if any, are undefined. Note: the Main and Backup Boot Sectors both contain the FatOffset field.

Sub-region Name	Offset (sector)	Size (sectors)	Comments
First FAT	FatOffset	FatLength	This sub-region is mandatory and Section 4.1 defines its contents. Note: the Main and Backup Boot Sectors both contain the FatOffset and FatLength fields.
Second FAT	FatOffset + FatLength	FatLength * (NumberOfFats – 1)	This sub-region is mandatory and Section 4.1 defines its contents, if any. Note: the Main and Backup Boot Sectors both contain the FatOffset, FatLength, and NumberOfFats fields. The NumberOfFats field may only hold values 1 and 2.
Data Region			
Cluster Heap Alignment	FatOffset + FatLength * NumberOfFats	ClusterHeapOffset – (FatOffset + FatLength * NumberOfFats)	This sub-region is mandatory and its contents, if any, are undefined. Note: the Main and Backup Boot Sectors both contain the FatOffset, FatLength, NumberOfFats, and ClusterHeapOffset fields. The NumberOfFats field’s valid values are 1 and 2.
Cluster Heap	ClusterHeapOffset	ClusterCount * $2^{SectorsPerClusterShift}$	This sub-region is mandatory and Section 5.1 defines its contents. Note: the Main and Backup Boot Sectors both contain the ClusterHeapOffset, ClusterCount, and SectorsPerClusterShift fields.

Sub-region Name	Offset (sector)	Size (sectors)	Comments
Excess Space	ClusterHeapOffset + ClusterCount * $2^{SectorsPerClusterShift}$	VolumeLength – (ClusterHeapOffset + ClusterCount * $2^{SectorsPerClusterShift}$)	This sub-region is mandatory and its contents, if any, are undefined. Note: the Main and Backup Boot Sectors both contain the ClusterHeapOffset, ClusterCount, SectorsPerClusterShift, and VolumeLength fields.

3 Main and Backup Boot Regions

The Main Boot region provides all the necessary boot-strapping instructions, identifying information, and file system parameters to enable an implementation to perform the following:

1. Boot-strap a computer system from an exFAT volume.
2. Identify the file system on the volume as exFAT.
3. Discover the location of the exFAT file system structures.

The Backup Boot region is a backup of the Main Boot region. It aids recovery of the exFAT volume in the advent of the Main Boot region being in an inconsistent state. Except under infrequent circumstances, such as updating boot-strapping instructions, implementations should not modify the contents of the Backup Boot region.

3.1 Main and Backup Boot Sector Sub-regions

The Main Boot Sector contains code for boot-strapping from an exFAT volume and fundamental exFAT parameters which describe the volume structure (see Table 4). BIOS, MBR, or other boot-strapping agents may inspect this sector and may load and execute any boot-strapping instructions contained therein.

The Backup Boot Sector is a backup of the Main Boot Sector and has the same structure (see Table 4). The Backup Boot Sector may aid recovery operations; however, implementations shall treat the contents of the VolumeFlags and PercentInUse fields as stale.

Prior to using the contents of either the Main or Backup Boot Sector, implementations shall verify their contents by validating their respective Boot Checksum and ensuring all their fields are within their valid value range.

While the initial format operation will initialize the contents of both the Main and Backup Boot Sectors, implementations may update these sectors (and shall also update their respective Boot Checksum) as needed. However, implementations may update either the VolumeFlags or PercentInUse fields without updating their respective Boot Checksum (the checksum specifically excludes these two fields).

Table 4 Main and Backup Boot Sector Structure

Field Name	Offset (byte)	Size (bytes)	Comments
JumpBoot	0	3	This field is mandatory and Section 3.1.1 defines its contents.
FileSystemName	3	8	This field is mandatory and Section 3.1.2 defines its contents.
MustBeZero	11	53	This field is mandatory and Section 3.1.3 defines its contents.
PartitionOffset	64	8	This field is mandatory and Section 3.1.4 defines its contents.
VolumeLength	72	8	This field is mandatory and Section 3.1.5 defines its contents.
FatOffset	80	4	This field is mandatory and Section 3.1.6 defines its contents.
FatLength	84	4	This field is mandatory and Section 3.1.7 defines its contents.
ClusterHeapOffset	88	4	This field is mandatory and Section 3.1.8 defines its contents.
ClusterCount	92	4	This field is mandatory and Section 3.1.9 defines its contents.

Field Name	Offset (byte)	Size (bytes)	Comments
FirstClusterOfRootDirectory	96	4	This field is mandatory and Section 3.1.10 defines its contents.
VolumeSerialNumber	100	4	This field is mandatory and Section 3.1.11 defines its contents.
FileSystemRevision	104	2	This field is mandatory and Section 3.1.12 defines its contents.
VolumeFlags	106	2	This field is mandatory and Section 3.1.13 defines its contents.
BytesPerSectorShift	108	1	This field is mandatory and Section 3.1.14 defines its contents.
SectorsPerClusterShift	109	1	This field is mandatory and Section 3.1.15 defines its contents.
NumberOfFats	110	1	This field is mandatory and Section 3.1.16 defines its contents.
DriveSelect	111	1	This field is mandatory and Section 3.1.17 defines its contents.
PercentInUse	112	1	This field is mandatory and Section 3.1.18 defines its contents.
Reserved	113	7	This field is mandatory and its contents are reserved.

Field Name	Offset (byte)	Size (bytes)	Comments
BootCode	120	390	This field is mandatory and Section 3.1.19 defines its contents.
BootSignature	510	2	This field is mandatory and Section 3.1.20 defines its contents.
ExcessSpace	512	$2^{\text{BytesPerSectorShift}}$ – 512	This field is mandatory and its contents, if any, are undefined. Note: the Main and Backup Boot Sectors both contain the BytesPerSectorShift field.

3.1.1 JumpBoot Field

The JumpBoot field shall contain the jump instruction for CPUs common in personal computers, which, when executed, "jumps" the CPU to execute the boot-strapping instructions in the BootCode field.

The valid value for this field is (in order of low-order byte to high-order byte) EBh 76h 90h.

3.1.2 FileSystemName Field

The FileSystemName field shall contain the name of the file system on the volume.

The valid value for this field is, in ASCII characters, "EXFAT ", which includes three trailing white spaces.

3.1.3 MustBeZero Field

The MustBeZero field shall directly correspond with the range of bytes the packed BIOS parameter block consumes on FAT12/16/32 volumes.

The valid value for this field is 0, which helps to prevent FAT12/16/32 implementations from mistakenly mounting an exFAT volume.

3.1.4 PartitionOffset Field

The PartitionOffset field shall describe the media-relative sector offset of the partition which hosts the given exFAT volume. This field aids boot-strapping from the volume using extended INT 13h on personal computers.

All possible values for this field are valid; however, the value 0 indicates implementations shall ignore this field.

3.1.5 VolumeLength Field

The VolumeLength field shall describe the size of the given exFAT volume in sectors.

The valid range of values for this field shall be:

- At least $2^{20}/ 2^{BytesPerSectorShift}$, which ensures the smallest volume is no less than 1MB
- At most $2^{64}- 1$, the largest value this field can describe

However, if the size of the Excess Space sub-region is 0, then the value of this field is ClusterHeapOffset + $(2^{32}- 11) * 2^{SectorsPerClusterShift}$.

3.1.6 FatOffset Field

The FatOffset field shall describe the volume-relative sector offset of the First FAT. This field enables implementations to align the First FAT to the characteristics of the underlying storage media.

The valid range of values for this field shall be:

- At least 24, which accounts for the sectors the Main Boot and Backup Boot regions consume
- At most ClusterHeapOffset - (FatLength * NumberOfFats), which accounts for the sectors the Cluster Heap consumes

3.1.7 FatLength Field

The FatLength field shall describe the length, in sectors, of each FAT table (the volume may contain up to two FATs).

The valid range of values for this field shall be:

- At least (ClusterCount + 2) * 2^2/ $2^{BytesPerSectorShift}$rounded up to the nearest integer, which ensures each FAT has sufficient space for describing all the clusters in the Cluster Heap
- At most (ClusterHeapOffset - FatOffset) / NumberOfFats rounded down to the nearest integer, which ensures the FATs exist before the Cluster Heap

This field may contain a value in excess of its lower bound (as described above) to enable the Second FAT, if present, to also be aligned to the characteristics of the underlying storage media. The contents of the space which exceeds what the FAT itself requires, if any, are undefined.

3.1.8 ClusterHeapOffset Field

The ClusterHeapOffset field shall describe the volume-relative sector offset of the Cluster Heap. This field enables implementations to align the Cluster Heap to the characteristics of the underlying storage media.

The valid range of values for this field shall be:

- At least FatOffset + FatLength * NumberOfFats, to account for the sectors all the preceding regions consume
- At most 2^{32}- 1 or VolumeLength - (ClusterCount * $2^{SectorsPerClusterShift}$), whichever calculation is less

3.1.9 ClusterCount Field

The ClusterCount field shall describe the number of clusters the Cluster Heap contains.

The valid value for this field shall be the lesser of the following:

- (VolumeLength - ClusterHeapOffset) / $2^{SectorsPerClusterShift}$rounded down to the nearest integer, which is exactly the number of clusters which can fit between the beginning of the Cluster Heap and the end of the volume
- 2^{32}- 11, which is the maximum number of clusters a FAT can describe

The value of the ClusterCount field determines the minimum size of a FAT. To avoid extremely large FATs, implementations can control the number of clusters in the Cluster Heap by increasing the cluster size (via the SectorsPerClusterShift field). This specification recommends no more than 2^{24}- 2 clusters in the Cluster Heap. However, implementations shall be able to handle volumes with up to 2^{32}- 11 clusters in the Cluster Heap.

3.1.10 FirstClusterOfRootDirectory Field

The FirstClusterOfRootDirectory field shall contain the cluster index of the first cluster of the root directory. Implementations should make every effort to place the first cluster of the root directory in the first non-bad cluster after the clusters the Allocation Bitmap and Up-case Table consume.

The valid range of values for this field shall be:

- At least 2, the index of the first cluster in the Cluster Heap
- At most ClusterCount + 1, the index of the last cluster in the Cluster Heap

3.1.11 VolumeSerialNumber Field

The VolumeSerialNumber field shall contain a unique serial number. This assists implementations to distinguish among different exFAT volumes. Implementations should generate the serial number by combining the date and time of formatting the exFAT volume. The mechanism for combining date and time to form a serial number is implementation-specific.

All possible values for this field are valid.

3.1.12 FileSystemRevision Field

The FileSystemRevision field shall describe the major and minor revision numbers of the exFAT structures on the given volume.

The high-order byte is the major revision number and the low-order byte is the minor revision number. For example, if the high-order byte contains the value 01h and if the low-order byte contains the value 05h, then the FileSystemRevision field describes the revision number 1.05. Likewise, if the high-order byte contains the value 0Ah and if the low-order byte contains the value 0Fh, then the FileSystemRevision field describes the revision number 10.15.

The valid range of values for this field shall be:

- At least 0 for the low-order byte and 1 for the high-order byte

- At most 99 for the low-order byte and 99 for the high-order byte

The revision number of exFAT this specification describes is 1.00. Implementations of this specification should mount any exFAT volume with major revision number 1 and shall not mount any exFAT volume with any other major revision number. Implementations shall honor the minor revision number and shall not perform operations or create any file system structures not described in the given minor revision number's corresponding specification.

3.1.13 VolumeFlags Field

The VolumeFlags field shall contain flags which indicate the status of various file system structures on the exFAT volume (see Table 5).

Implementations shall not include this field when computing its respective Main Boot or Backup Boot region checksum. When referring to the Backup Boot Sector, implementations shall treat this field as stale.

Table 5 VolumeFlags Field Structure

Field Name	Offset (bit)	Size (bits)	Comments
ActiveFat	0	1	This field is mandatory and Section 3.1.13.1 defines its contents.
VolumeDirty	1	1	This field is mandatory and Section 3.1.13.2 defines its contents.
MediaFailure	2	1	This field is mandatory and Section 3.1.13.3 defines its contents.
ClearToZero	3	1	This field is mandatory and Section 3.1.13.4 defines its contents.
Reserved	4	12	This field is mandatory and its contents are reserved.

3.1.13.1 ActiveFat Field

The ActiveFat field shall describe which FAT and Allocation Bitmap are active (and implementations shall use), as follows:

- 0, which means the First FAT and First Allocation Bitmap are active
- 1, which means the Second FAT and Second Allocation Bitmap are active and is possible only when the NumberOfFats field contains the value 2

Implementations shall consider the inactive FAT and Allocation Bitmap as stale. Only TexFAT-aware implementations shall switch the active FAT and Allocation Bitmaps (see Section 7.1).

3.1.13.2 VolumeDirty Field

The VolumeDirty field shall describe whether the volume is dirty or not, as follows:

- 0, which means the volume is probably in a consistent state
- 1, which means the volume is probably in an inconsistent state

Implementations should set the value of this field to 1 upon encountering file system metadata inconsistencies which they do not resolve. If, upon mounting a volume, the value of this field is 1, only implementations which resolve file system metadata inconsistencies may clear the value of this field to 0. Such implementations shall only clear the value of this field to 0 after ensuring the file system is in a consistent state.

If, upon mounting a volume, the value of this field is 0, implementations should set this field to 1 before updating file system metadata and clear this field to 0 afterwards, similar to the recommended write ordering described in Section 8.1.

3.1.13.3 MediaFailure Field

The MediaFailure field shall describe whether an implementation has discovered media failures or not, as follows:

- 0, which means the hosting media has not reported failures or any known failures are already recorded in the FAT as "bad" clusters
- 1, which means the hosting media has reported failures (i.e. has failed read or write operations)

An implementation should set this field to 1 when:

1. The hosting media fails access attempts to any region in the volume
2. The implementation has exhausted access retry algorithms, if any

If, upon mounting a volume, the value of this field is 1, implementations which scan the entire volume for media failures and record all failures as "bad" clusters in the FAT (or otherwise resolve media failures) may clear the value of this field to 0.

3.1.13.4 ClearToZero Field

The ClearToZero field does not have significant meaning in this specification.

The valid values for this field are:

- 0, which does not have any particular meaning
- 1, which means implementations shall clear this field to 0 prior to modifying any file system structures, directories, or files

3.1.14 BytesPerSectorShift Field

The BytesPerSectorShift field shall describe the bytes per sector expressed as log~2~(N), where N is the number of bytes per sector. For example, for 512 bytes per sector, the value of this field is 9.

The valid range of values for this field shall be:

- At least 9 (sector size of 512 bytes), which is the smallest sector possible for an exFAT volume
- At most 12 (sector size of 4096 bytes), which is the memory page size of CPUs common in personal computers

3.1.15 SectorsPerClusterShift Field

The SectorsPerClusterShift field shall describe the sectors per cluster expressed as log~2~(N), where N is number of sectors per cluster. For example, for 8 sectors per cluster, the value of this field is 3.

The valid range of values for this field shall be:

- At least 0 (1 sector per cluster), which is the smallest cluster possible
- At most 25 - BytesPerSectorShift, which evaluates to a cluster size of 32MB

3.1.16 NumberOfFats Field

The NumberOfFats field shall describe the number of FATs and Allocation Bitmaps the volume contains.

The valid range of values for this field shall be:

- 1, which indicates the volume only contains the First FAT and First Allocation Bitmap
- 2, which indicates the volume contains the First FAT, Second FAT, First Allocation Bitmap, and Second Allocation Bitmap; this value is only valid for TexFAT volumes

3.1.17 DriveSelect Field

The DriveSelect field shall contain the extended INT 13h drive number, which aids boot-strapping from this volume using extended INT 13h on personal computers.

All possible values for this field are valid. Similar fields in previous FAT-based file systems frequently contained the value 80h.

3.1.18 PercentInUse Field

The PercentInUse field shall describe the percentage of clusters in the Cluster Heap which are allocated.

The valid range of values for this field shall be:

- Between 0 and 100 inclusively, which is the percentage of allocated clusters in the Cluster Heap, rounded down to the nearest integer
- Exactly FFh, which indicates the percentage of allocated clusters in the Cluster Heap is not available

Implementations shall change the value of this field to reflect changes in the allocation of clusters in the Cluster Heap or shall change it to FFh.

Implementations shall not include this field when computing its respective Main Boot or Backup Boot region checksum. When referring to the Backup Boot Sector, implementations shall treat this field as stale.

3.1.19 BootCode Field

The BootCode field shall contain boot-strapping instructions. Implementations may populate this field with the CPU instructions necessary for boot-strapping a computer system. Implementations which don't provide boot-strapping instructions shall initialize each byte in this field to F4h (the halt instruction for CPUs common in personal computers) as part of their format operation.

3.1.20 BootSignature Field

The BootSignature field shall describe whether the intent of a given sector is for it to be a Boot Sector or not.

The valid value for this field is AA55h. Any other value in this field invalidates its respective Boot Sector. Implementations should verify the contents of this field prior to depending on any other field in its respective Boot Sector.

3.2 Main and Backup Extended Boot Sectors Sub-regions

Each sector of the Main Extended Boot Sectors has the same structure; however, each sector may hold distinct boot-strapping instructions (see Table 6). Boot-strapping agents, such as the boot-strapping instructions in the Main Boot Sector, alternate BIOS implementations, or an embedded system's firmware, may load these sectors and execute the instructions they contain.

The Backup Extended Boot Sectors is a backup of the Main Extended Boot Sectors and has the same structure (see Table 6).

Prior to executing the instructions of either the Main or Backup Extended Boot Sectors, implementations should verify their contents by ensuring each sector's ExtendedBootSignature field contains its prescribed value.

While the initial format operation will initialize the contents of both the Main and Backup Extended Boot Sectors, implementations may update these sectors (and shall also update their respective Boot Checksum) as needed.

Table 6 Extended Boot Sector Structure

	Offset	Size	
Field Name	(byte)	(bytes)	Comments

Field Name	Offset (byte)	Size (bytes)	Comments
ExtendedBootCode	0	$2^{BytesPerSectorShift} – 4$	This field is mandatory and Section 3.2.1 defines its contents. Note: the Main and Backup Boot Sectors both contain the BytesPerSectorShift field.
ExtendedBootSignature	$2^{BytesPerSectorShift} – 4$	4	This field is mandatory and Section 3.2.2 defines its contents. Note: the Main and Backup Boot Sectors both contain the BytesPerSectorShift field.

3.2.1 ExtendedBootCode Field

The ExtendedBootCode field shall contain boot-strapping instructions. Implementations may populate this field with the CPU instructions necessary for boot-strapping a computer system. Implementations which don't provide boot-strapping instructions shall initialize each byte in this field to 00h as part of their format operation.

3.2.2 ExtendedBootSignature Field

The ExtendedBootSignature field shall describe whether the intent of given sector is for it to be an Extended Boot Sector or not.

The valid value for this field is AA550000h. Any other value in this field invalidates its respective Main or Backup Extended Boot Sector. Implementations should verify the contents of this field prior to depending on any other field in its respective Extended Boot Sector.

3.3 Main and Backup OEM Parameters Sub-regions

The Main OEM Parameters sub-region contains ten parameters structures which may contain manufacturer-specific information (see Table 7). Each of the ten parameters structures derives from the Generic Parameters template (see Section 3.3.2). Manufacturers may derive their own custom parameters structures from the Generic Parameters template. This specification itself defines two parameters structures: Null Parameters (see Section 3.3.3) and Flash Parameters (see Section 3.3.4).

The Backup OEM Parameters is a backup of the Main OEM Parameters and has the same structure (see Table 7).

Prior to using the contents of either the Main or Backup OEM Parameters, implementations shall verify their contents by validating their respective Boot Checksum.

Manufacturers should populate the Main and Backup OEM Parameters with their own custom parameters structures, if any, and any other parameter structures. Subsequent format operations shall preserve the contents of the Main and Backup OEM Parameters.

Implementations may update the Main and Backup OEM Parameters as needed (and shall also update their respective Boot Checksum).

Table 7 OEM Parameters Structure

	Offset	**Size**	
Field Name	**(byte)**	**(bytes)**	**Comments**
Parameters[0]	0	48	This field is mandatory and Section 3.3.1 defines its contents.
. . .	. . .	. . .	. . .
Parameters[9]	432	48	This field is mandatory and Section 3.3.1 defines its contents.
Reserved	480	$2^{BytesPerSectorShift}$ – 480	This field is mandatory and its contents are reserved. Note: the Main and Backup Boot Sectors both contain the BytesPerSectorShift field.

3.3.1 Parameters[0] ... Parameters[9]

Each Parameters field in this array contains a parameters structure, which derives from the Generic Parameters template (see Section 3.3.2). Any unused Parameters field shall be described as containing a Null Parameters structure (see Section 3.3.3).

3.3.2 Generic Parameters Template

The Generic Parameters template provides the base definition of a parameters structure (see Table 8). All parameters structures derive from this template. Support for this Generic Parameters template is mandatory.

Table 8 Generic Parameters Template

	Offset	Size	
Field Name	**(byte)**	**(bytes)**	**Comments**
ParametersGuid	0	16	This field is mandatory and Section 3.3.2.1 defines its contents.
CustomDefined	16	32	This field is mandatory and the structures which derive from this template define its contents.

3.3.2.1 ParametersGuid Field

The ParametersGuid field shall describe a GUID, which determines the layout of the remainder of the given parameters structure.

All possible values for this field are valid; however, manufacturers should use a GUID-generating tool, such as GuidGen.exe, to select a GUID when deriving custom parameters structures from this template.

3.3.3 Null Parameters

The Null Parameters structure derives from the Generic Parameters template (see Section 3.3.2) and shall describe an unused Parameters field (see Table 9). When creating or updating the OEM Parameters structure, implementations shall populate unused Parameters fields with the Null Parameters structure. Also, when creating or updating the OEM Parameters structure, implementations should consolidate Null Parameters structures at the end of the array, thereby leaving all other Parameters structures at the beginning of the OEM Parameters structure.

Support for the Null Parameters structure is mandatory.

Table 9 Null Parameters Structure

Field Name	Offset (byte)	Size (bytes)	Comments
ParametersGuid	0	16	This field is mandatory and Section 3.3.3.1 defines its contents.
Reserved	16	32	This field is mandatory and its contents are reserved.

3.3.3.1 ParametersGuid Field

The ParametersGuid field shall conform to the definition provided by the Generic Parameters template (see Section 3.3.2.1).

The valid value for this field, in GUID notation, is {00000000-0000-0000-0000-000000000000}.

3.3.4 Flash Parameters

The Flash Parameter structure derives from the Generic Parameters template (see Section 3.3.2) and contains parameters for flash media (see Table 10). Manufacturers of flash-based storage devices may populate a Parameters field (preferably the Parameters[0] field) with this parameters structure. Implementations may use the information in the Flash Parameters structure to optimize access operations during reads/writes and for alignment of file system structures durning formatting of the media.

Support for the Flash Parameters structure is optional.

Table 10 Flash Parameters Structure

Field Name	Offset (byte)	Size (bytes)	Comments
ParametersGuid	0	16	This field is mandatory and Section 3.3.4.1 defines its contents.
EraseBlockSize	16	4	This field is mandatory and Section 3.3.4.2 defines its contents.

Field Name	Offset (byte)	Size (bytes)	Comments
PageSize	20	4	This field is mandatory and Section 3.3.4.3 defines its contents.
SpareSectors	24	4	This field is mandatory and Section 3.3.4.4 defines its contents.
RandomAccessTime	28	4	This field is mandatory and Section 3.3.4.5 defines its contents.
ProgrammingTime	32	4	This field is mandatory and Section 3.3.4.6 defines its contents.
ReadCycle	36	4	This field is mandatory and Section 3.3.4.7 defines its contents.
WriteCycle	40	4	This field is mandatory and Section 3.3.4.8 defines its contents.
Reserved	44	4	This field is mandatory and its contents are reserved.

All possible values for all Flash Parameters fields, except for the ParametersGuid field, are valid. However, the value 0 indicates the field is actually meaningless (implementations shall ignore the given field).

3.3.4.1 ParametersGuid Field

The ParametersGuid field shall conform to the definition provided in the Generic Parameters template (see Section 3.3.2.1).

The valid value for this field, in GUID notation, is {0A0C7E46-3399-4021-90C8-FA6D389C4BA2}.

3.3.4.2 EraseBlockSize Field

The EraseBlockSize field shall describe the size, in bytes, of the flash media's erase block.

3.3.4.3 PageSize Field

The PageSize field shall describe the size, in bytes of the flash media's page.

3.3.4.4 SpareSectors Field

The SpareSectors field shall describe the number of sectors the flash media has available for its internal sparing operations.

3.3.4.5 RandomAccessTime Field

The RandomAccessTime field shall describe the flash media's average random access time, in nanoseconds.

3.3.4.6 ProgrammingTime Field

The ProgrammingTime field shall describe the flash media's average programming time, in nanoseconds.

3.3.4.7 ReadCycle Field

The ReadCycle field shall describe the flash media's average read cycle time, in nanoseconds.

3.3.4.8 WriteCycle Field

The WriteCycle field shall describe the average write cycle time, in nanoseconds.

3.4 Main and Backup Boot Checksum Sub-regions

The Main and Backup Boot Checksums each contain a repeating pattern of the four-byte checksum of the contents of all other sub-regions in their respective Boot regions. The checksum calculation shall not include the VolumeFlags and PercentInUse fields in their respective Boot Sector (see Figure 1). The repeating pattern of the four-byte checksum fills its respective Boot Checksum sub-region from the beginning to the end of the sub-region.

Prior to using the contents of any of the other sub-regions in either the Main or Backup Boot regions, implementations shall verify their contents by validating their respective Boot Checksum.

While the initial format operation will populate both the Main and Backup Boot Checksums with the repeating checksum pattern, implementations shall update these sectors as the contents of the other sectors in their respective Boot regions change.

Figure 1 Boot Checksum Computation

```
UInt32  BootChecksum
(
    UCHAR * Sectors,        // points to an in-memory copy of the 11 sectors
       USHORT   BytesPerSector
)
{
       UInt32   NumberOfBytes = (UInt32)BytesPerSector * 11;
       UInt32   Checksum =      0;
       UInt32   Index;

       for (Index = 0; Index < NumberOfBytes; Index++)
       {
        if ((Index == 106) || (Index == 107) || (Index == 112))
              {
                     continue;
              }
              Checksum = ((Checksum&1) ? 0x80000000 : 0) + (Checksum>>1) +
(UInt32)Sectors[Index];
    }

       return Checksum;
}
```

4 File Allocation Table Region

The File Allocation Table (FAT) region may contain up to two FATs, one in the First FAT sub-region and another in the Second FAT sub-region. The NumberOfFats field describes how many FATs this region contains. The valid values for the NumberOfFats field are 1 and 2. Therefore, the First FAT sub-region always contains a FAT. If the NumberOfFats field is two, then the Second FAT sub-region also contains a FAT.

The ActiveFat field of the VolumeFlags field describes which FAT is active. Only the VolumeFlags field in the Main Boot Sector is current. Implementations shall treat the FAT which is not active as stale. Use of the inactive FAT and switching between FATs is implementation specific.

4.1 First and Second FAT Sub-regions

A FAT shall describe cluster chains in the Cluster Heap (see Table 11). A cluster chain is a series of clusters which provides space for recording the contents of files, directories, and other file system structures. A FAT represents a cluster chain as a singly-linked list of cluster indices. With the exception of the first two entries, every entry in a FAT represents exactly one cluster.

Table 11 File Allocation Table Structure

Field Name	Offset (byte)	Size (bytes)	Comments
FatEntry[0]	0	4	This field is mandatory and Section 4.1.1 defines its contents.
FatEntry[1]	4	4	This field is mandatory and Section 4.1.2 defines its contents.
FatEntry[2]	8	4	This field is mandatory and Section 4.1.3 defines its contents.
. . .	. . .	. . .	. . .
FatEntry[ClusterCount+1]	(ClusterCount + 1) * 4	4	This field is mandatory and Section 4.1.3 defines its contents. ClusterCount + 1 can never exceed FFFFFFF6h. Note: the Main and Backup Boot Sectors both contain the ClusterCount field.

Field Name	Offset (byte)	Size (bytes)	Comments
ExcessSpace	(ClusterCount + 2) * 4	(FatLength * $2^{BytesPerSectorShift}$) – ((ClusterCount + 2) * 4)	This field is mandatory and its contents, if any, are undefined. Note: the Main and Backup Boot Sectors both contain the ClusterCount, FatLength, and BytesPerSectorShift fields.

4.1.1 FatEntry[0] Field

The FatEntry[0] field shall describe the media type in the first byte (the lowest order byte) and shall contain FFh in the remaining three bytes.

The media type (the first byte) should be F8h.

4.1.2 FatEntry[1] Field

The FatEntry[1] field only exists due to historical precedence and does not describe anything of interest.

The valid value for this field is FFFFFFFFh. Implementations shall initialize this field to its prescribed value and should not use this field for any purpose. Implementations should not interpret this field and shall preserve its contents across operations which modify surrounding fields.

4.1.3 FatEntry[2] ... FatEntry[ClusterCount+1] Fields

Each FatEntry field in this array shall represent a cluster in the Cluster Heap. FatEntry[2] represents the first cluster in the Cluster Heap and FatEntry[ClusterCount+1] represents the last cluster in the Cluster Heap.

The valid range of values for these fields shall be:

- Between 2 and ClusterCount + 1, inclusively, which points to the next FatEntry in the given cluster chain; the given FatEntry shall not point to any FatEntry which precedes it in the given cluster chain

- Exactly FFFFFFF7h, which marks the given FatEntry's corresponding cluster as "bad"
- Exactly FFFFFFFFh, which marks the given FatEntry's corresponding cluster as the last cluster of a cluster chain; this is the only valid value for the last FatEntry of any given cluster chain

5 Data Region

The Data region contains the Cluster Heap, which provides managed space for file system structures, directories, and files.

5.1 Cluster Heap Sub-region

The Cluster Heap's structure is very simple (see Table 12); each consecutive series of sectors describes one cluster, as the SectorsPerClusterShift field defines. Importantly, the first cluster of the Cluster Heap has index two, which directly corresponds to the index of FatEntry[2].

In an exFAT volume, an Allocation Bitmap (see Section 7.1.5) maintains the record of the allocation state of all clusters. This is a significant difference from exFAT's predecessors (FAT12, FAT16, and FAT32), in which a FAT maintained a record of the allocation state of all clusters in the Cluster Heap.

Table 12 Cluster Heap Structure

	Offset	**Size**	
Field Name	**(sector)**	**(sectors)**	**Comments**
Cluster[2]	ClusterHeapOffset	$2^{\text{SectorsPerClusterShift}}$	This field is mandatory and Section 5.1.1 defines its contents. Note: the Main and Backup Boot Sectors both contain the ClusterHeapOffset and SectorsPerClusterShift fields.

Field Name	Offset (sector)	Size (sectors)	Comments
. . .	. . .	. . .	. . .
Cluster[ClusterCount+1]	ClusterHeapOffset + (ClusterCount – 1) * $2^{SectorsPerClusterShift}$	$2^{SectorsPerClusterShift}$	This field is mandatory and Section 5.1.1 defines its contents. Note: the Main and Backup Boot Sectors both contain the ClusterCount, ClusterHeapOffset, and SectorsPerClusterShift fields.

5.1.1 Cluster[2] ... Cluster[ClusterCount+1] Fields

Each Cluster field in this array is a series of contiguous sectors, whose size is defined by the SectorsPerClusterShift field.

6 Directory Structure

The exFAT file system uses a directory tree approach to manage the file system structures and files which exist in the Cluster Heap. Directories have a one-to-many relationship between parent and child in the directory tree.

The directory to which the FirstClusterOfRootDirectory field refers is the root of the directory tree. All other directories descend from the root directory in a singly-linked fashion.

Each directory consists of a series of directory entries (see Table 13).

One or more directory entries combine into a directory entry set which describes something of interest, such as a file system structure, sub-directory, or file.

Table 13 Directory Structure

Field Name	Offset (byte)	Size (byte)	Comments
DirectoryEntry[0]	0	32	This field is mandatory and Section 6.1 defines its contents.
. . .	. . .	. . .	. . .
DirectoryEntry[N–1]	(N – 1) * 32	32	This field is mandatory and Section 6.1 defines its contents. N, the number of DirectoryEntry fields, is the size, in bytes, of the cluster chain which contains the given directory, divided by the size of a DirectoryEntry field, 32 bytes.

6.1 DirectoryEntry[0] ... DirectoryEntry[N--1]

Each DirectoryEntry field in this array derives from the Generic DirectoryEntry template (see Section 6.2).

6.2 Generic DirectoryEntry Template

The Generic DirectoryEntry template provides the base definition for directory entries (see Table 14). All directory entry structures derive from this template and only Microsoft-defined directory entry structures are valid (exFAT does not have provisions for manufacturer-defined directory entry structures except as defined in section 7.8 and section 7.9). The ability to interpret the Generic DirectoryEntry template is mandatory.

Table 14 Generic DirectoryEntry Template

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 6.2.1 defines its contents.

Field Name	Offset (byte)	Size (byte)	Comments
CustomDefined	1	19	This field is mandatory and structures which derive from this template may define its contents.
FirstCluster	20	4	This field is mandatory and Section 6.2.2 defines its contents.
DataLength	24	8	This field is mandatory and Section 6.2.3 defines its contents.

6.2.1 EntryType Field

The EntryType field has three modes of usage which the value of the field defines (see list below).

- 00h, which is an end-of-directory marker and the following conditions apply:
  - All other fields in the given DirectoryEntry are actually reserved
  - All subsequent directory entries in the given directory also are end-of-directory markers
  - End-of-directory markers are only valid outside directory entry sets
  - Implementations may overwrite end-of-directory markers as necessary
- Between 01h and 7Fh inclusively, which is an unused-directory-entry marker and the following conditions apply:
  - All other fields in the given DirectoryEntry are actually undefined
  - Unused directory entries are only valid outside of directory entry sets
  - Implementations may overwrite unused directory entries as necessary
  - This range of values corresponds to the InUse field (see Section 6.2.1.4) containing the value 0
- Between 81h and FFh inclusively, which is a regular directory entry and the following conditions apply:

- The contents of the EntryType field (see Table 15) determine the layout of the remainder of the DirectoryEntry structure
- This range of values, and only this range of values, are valid inside a directory entry set
- This range of values directly corresponds to the InUse field (see Section 6.2.1.4) containing the value 1

To prevent modifications to the InUse field (see Section 6.2.1.4) erroneously resulting in an end-of-directory marker, the value 80h is invalid.

Table 15 Generic EntryType Field Structure

Field Name	Offset (bit)	Size (bits)	Comments
TypeCode	0	5	This field is mandatory and Section 6.2.1.1 defines its contents.
TypeImportance	5	1	This field is mandatory and Section 6.2.1.2 defines its contents.
TypeCategory	6	1	This field is mandatory and Section 6.2.1.3 defines its contents.
InUse	7	1	This field is mandatory and Section 6.2.1.4 defines its contents.

6.2.1.1 TypeCode Field

The TypeCode field partially describes the specific type of the given directory entry. This field, plus the TypeImportance and TypeCategory fields (see Sections 6.2.1.2 and 6.2.1.3, respectively) uniquely identify the type of the given directory entry.

All possible values of this field are valid, unless the TypeImportance and TypeCategory fields both contain the value 0; in that case, the value 0 is invalid for this field.

6.2.1.2 TypeImportance Field

The TypeImportance field shall describe the importance of the given directory entry.

The valid values for this field shall be:

- 0, which means the given directory entry is critical (see Sections 6.3.1.2.1 and 6.4.1.2.1 for critical primary and critical secondary directory entries, respectively)
- 1, which means the given directory entry is benign (see Sections 6.3.1.2.2 and 6.4.1.2.2 for benign primary and benign secondary directory entries, respectively)

6.2.1.3 TypeCategory Field

The TypeCategory field shall describe the category of the given directory entry.

The valid values for this field shall be:

- 0, which means the given directory entry is primary (see Section 6.3)
- 1, which means the given directory entry is secondary (see Section 6.4)

6.2.1.4 InUse Field

The InUse field shall describe whether the given directory entry in use or not.

The valid values for this field shall be:

- 0, which means the given directory entry is not in use; this means the given structure actually is an unused directory entry
- 1, which means the given directory entry is in use; this means the given structure is a regular directory entry

6.2.2 FirstCluster Field

The FirstCluster field shall contain the index of the first cluster of an allocation in the Cluster Heap associated with the given directory entry.

The valid range of values for this field shall be:

- Exactly 0, which means no cluster allocation exists
- Between 2 and ClusterCount + 1, which is the range of valid cluster indices

Structures which derive from this template may redefine both the FirstCluster and DataLength fields, if a cluster allocation is not compatible with the derivative structure.

6.2.3 DataLength Field

The DataLength field describes the size, in bytes, of the data the associated cluster allocation contains.

The valid range of value for this field is:

- At least 0; if the FirstCluster field contains the value 0, then this field's only valid value is 0
- At most ClusterCount * $2^{\text{SectorsPerClusterShift}} * 2^{\text{BytesPerSectorShift}}$

Structures which derive from this template may redefine both the FirstCluster and DataLength fields, if a cluster allocation is not possible for the derivative structure.

6.3 Generic Primary DirectoryEntry Template

The first directory entry in a directory entry set shall be a primary directory entry. All subsequent directory entries, if any, in the directory entry set shall be secondary directory entries (see Section 6.4).

The ability to interpret the Generic Primary DirectoryEntry template is mandatory.

All primary directory entry structures derive from the Generic Primary DirectoryEntry template (see Table 16), which derives from the Generic DirectoryEntry template (see Section 6.2).

Table 16 Generic Primary DirectoryEntry Template

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 6.3.1 defines its contents.
SecondaryCount	1	1	This field is mandatory and Section 6.3.2 defines its contents.
SetChecksum	2	2	This field is mandatory and Section 6.3.3 defines its contents.
GeneralPrimaryFlags	4	2	This field is mandatory and Section 6.3.4 defines its contents.

Field Name	Offset (byte)	Size (byte)	Comments
CustomDefined	6	14	This field is mandatory and structures which derive from this template define its contents.
FirstCluster	20	4	This field is mandatory and Section 6.3.5 defines its contents.
DataLength	24	8	This field is mandatory and Section 6.3.6 defines its contents.

6.3.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1).

6.3.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.1).

6.3.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.2).

6.3.1.2.1 Critical Primary Directory Entries

Critical primary directory entries contain information which is critical to the proper management of an exFAT volume. Only the root directory contains critical primary directory entries (File directory entries are an exception, see Section 7.4).

The definition of critical primary directory entries correlates to the major exFAT revision number. Implementations shall support all critical primary directory entries and shall only record the critical primary directory entry structures this specification defines.

6.3.1.2.2 Benign Primary Directory Entries

Benign primary directory entries contain additional information which may be useful for managing an exFAT volume. Any directory may contain benign primary directory entries.

The definition of benign primary directory entries correlates to the minor exFAT revision number. Support for any benign primary directory entry this specification, or any subsequent specification, defines is optional. An unrecognized benign primary directory entry renders the entire directory entry set as unrecognized (beyond the definition of the applicable directory entry templates).

6.3.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.3).

For this template, the valid value for this field shall be 0.

6.3.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.4).

6.3.2 SecondaryCount Field

The SecondaryCount field shall describe the number of secondary directory entries which immediately follow the given primary directory entry. These secondary directory entries, along with the given primary directory entry, comprise the directory entry set.

The valid range of values for this field shall be:

- At least 0, which means this primary directory entry is the only entry in the directory entry set
- At most 255, which means the next 255 directory entries and this primary directory entry comprise the directory entry set

Critical primary directory entry structures which derive from this template may redefine both the SecondaryCount and SetChecksum fields.

6.3.3 SetChecksum Field

The SetChecksum field shall contain the checksum of all directory entries in the given directory entry set. However, the checksum excludes this field (see Figure 2). Implementations shall verify the contents of this field are valid prior to using any other directory entry in the given directory entry set.

Critical primary directory entry structures which derive from this template may redefine both the SecondaryCount and SetChecksum fields.

Figure 2 EntrySetChecksum Computation

Copy

```
UInt16  EntrySetChecksum
(
    UCHAR * Entries,        // points to an in-memory copy of the directory
entry set
       UCHAR    SecondaryCount
)
{
       UInt16   NumberOfBytes = ((UInt16)SecondaryCount + 1) * 32;
    UInt16  Checksum =      0;
    UInt16  Index;

       for (Index = 0; Index < NumberOfBytes; Index++)
    {
            if ((Index == 2) || (Index == 3))
              {
                    continue;
            }
            Checksum = ((Checksum&1) ? 0x8000 : 0) + (Checksum>>1) +
(UInt16)Entries[Index];
       }

       return Checksum;
}
```

6.3.4 GeneralPrimaryFlags Field

The GeneralPrimaryFlags field contains flags (see Table 17).

Critical primary directory entry structures which derive from this template may redefine this field.

Table 17 Generic GeneralPrimaryFlags Field Structure

Field Name	Offset (bit)	Size (bits)	Comments
AllocationPossible	0	1	This field is mandatory and Section 6.3.4.1 defines its contents.
NoFatChain	1	1	This field is mandatory and Section 6.3.4.2 defines its contents.
CustomDefined	2	14	This field is mandatory and structures which derive from this template may define this field.

6.3.4.1 AllocationPossible Field

The AllocationPossible field shall describe whether or not an allocation in the Cluster Heap is possible for the given directory entry.

The valid values for this field shall be:

- 0, which means an associated allocation of clusters is not possible and the FirstCluster and DataLength fields are actually undefined (structures which derive from this template may redefine those fields)

- 1, which means an associated allocation of clusters is possible and the FirstCluster and DataLength fields are as defined

6.3.4.2 NoFatChain Field

The NoFatChain field shall indicate whether or not the active FAT describes the given allocation's cluster chain.

The valid values for this field shall be:

- 0, which means the corresponding FAT entries for the allocation's cluster chain are valid and implementations shall interpret them; if the AllocationPossible field contains the value 0, or if the AllocationPossible field contains the value 1 and the FirstCluster field contains the value 0, then this field's only valid value is 0

- 1, which means the associated allocation is one contiguous series of clusters; the corresponding FAT entries for the clusters are invalid and implementations shall not interpret them; implementations may use the following equation to calculate the size of the associated allocation: DataLength / ($2^{SectorsPerClusterShift} * 2^{BytesPerSectorShift}$) rounded up to the nearest integer

If critical primary directory entry structures which derive from this template redefine the GeneralPrimaryFlags field, then the corresponding FAT entries for any associated allocation's cluster chain are valid.

6.3.5 FirstCluster Field

The FirstCluster field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.2).

If the NoFatChain bit is 1 then FirstCluster must point to a valid cluster in the cluster heap.

Critical primary directory entry structures which derive from this template may redefine the FirstCluster and DataLength fields. Other structures which derive from this template may redefine the FirstCluster and DataLength fields only if the AllocationPossible field contains the value 0.

6.3.6 DataLength Field

The DataLength field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.3).

If the NoFatChain bit is 1 then DataLength must not be zero. If the FirstCluster field is zero, then DataLength must also be zero.

Critical primary directory entry structures which derive from this template may redefine the FirstCluster and DataLength fields. Other structures which derive from this template may redefine the FirstCluster and DataLength fields only if the AllocationPossible field contains the value 0.

6.4 Generic Secondary DirectoryEntry Template

The central purpose of secondary directory entries is to provide additional information about a directory entry set. The ability to interpret the Generic Secondary DirectoryEntry template is mandatory.

The definition of both critical and benign secondary directory entries correlates to the minor exFAT revision number. Support for any critical or benign secondary directory entry this specification, or subsequent specifications, defines is optional.

All secondary directory entry structures derive from the Generic Secondary DirectoryEntry template (see Table 18), which derives from the Generic DirectoryEntry template (see Section 6.2).

Table 18 Generic Secondary DirectoryEntry Template

	Offset	Size	
Field Name	**(byte)**	**(byte)**	**Comments**
EntryType	0	1	This field is mandatory and Section 6.4.1 defines its contents.
GeneralSecondaryFlags	1	1	This field is mandatory and Section 6.4.2 defines its contents.
CustomDefined	2	18	This field is mandatory and structures which derive from this template define its contents.
FirstCluster	20	4	This field is mandatory and Section 6.4.3 defines its contents.
DataLength	24	8	This field is mandatory and Section 6.4.4 defines its contents.

6.4.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1)

6.4.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.1).

6.4.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.2).

6.4.1.2.1 Critical Secondary Directory Entries

Critical secondary directory entries contain information which is critical to the proper management of its containing directory entry set. While support for any specific critical secondary directory entry is optional, an unrecognized critical directory entry renders the entire directory entry set as unrecognized (beyond the definition of the applicable directory entry templates).

However, if a directory entry set contains at least one critical secondary directory entry which an implementation does not recognize, then the implementation shall at most interpret the templates of the directory entries in the directory entry set and not the data any allocation associated with any directory entry in the directory entry set contains (File directory entries are an exception, see Section 7.4).

6.4.1.2.2 Benign Secondary Directory Entries

Benign secondary directory entries contain additional information which may be useful for managing its containing directory entry set. Support for any specific benign secondary directory entry is optional. Unrecognized benign secondary directory entries do not render the entire directory entry set as unrecognized.

Implementations may ignore any benign secondary entry it does not recognize.

6.4.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.3).

For this template, the valid value for this field is 1.

6.4.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.1.4).

6.4.2 GeneralSecondaryFlags Field

The GeneralSecondaryFlags field contains flags (see Table 19).

Table 19 Generic GeneralSecondaryFlags Field Structure

Field Name	Offset (bit)	Size (bits)	Comments

Field Name	Offset (bit)	Size (bits)	Comments
AllocationPossible	0	1	This field is mandatory and Section 6.4.2.1 defines its contents.
NoFatChain	1	1	This field is mandatory and Section 6.4.2.2 defines its contents.
CustomDefined	2	6	This field is mandatory and structures which derive from this template may define this field.

6.4.2.1 AllocationPossible Field

The AllocationPossible field shall have the same definition as the same-named field in the Generic Primary DirectoryEntry template (see Section 6.3.4.1).

6.4.2.2 NoFatChain Field

The NoFatChain field shall have the same definition as the same-named field in the Generic Primary DirectoryEntry template (see Section 6.3.4.2).

6.4.3 FirstCluster Field

The FirstCluster field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.2).

If the NoFatChain bit is 1 then FirstCluster must point to a valid cluster in the cluster heap.

6.4.4 DataLength Field

The DataLength field shall conform to the definition provided in the Generic DirectoryEntry template (see Section 6.2.3).

If the NoFatChain bit is 1 then DataLength must not be zero. If the FirstCluster field is zero, then DataLength must also be zero.

7 Directory Entry Definitions

Revision 1.00 of the exFAT file system defines the following directory entries:

- Critical primary
  - Allocation Bitmap (Section 7.1)
  - Up-case Table (Section 7.2)
  - Volume Label (Section 7.3)
  - File (Section 7.4)
- Benign primary
  - Volume GUID (Section 7.5)
  - TexFAT Padding (Section 7.10)
- Critical secondary
  - Stream Extension (Section 7.6)
  - File Name (Section 7.7)
- Benign secondary
  - Vendor Extension (Section 7.8)
  - Vendor Allocation (Section 7.9)

7.1 Allocation Bitmap Directory Entry

In the exFAT file system, a FAT does not describe the allocation state of clusters; rather, an Allocation Bitmap does. Allocation Bitmaps exist in the Cluster Heap (see Section 7.1.5) and have corresponding critical primary directory entries in the root directory (see Table 20).

The NumberOfFats field determines the number of valid Allocation Bitmap directory entries in the root directory. If the NumberOfFats field contains the value 1, then the only valid number of Allocation Bitmap directory entries is 1. Further, the one Allocation Bitmap directory entry is only valid if it describes the First Allocation Bitmap (see Section 7.1.2.1). If the NumberOfFats field contains the value 2, then the only valid number of Allocation Bitmap directory entries is 2. Further, the two Allocation Bitmap directory entries are only valid if one describes the First Allocation Bitmap and the other describes the Second Allocation Bitmap.

Table 20 Allocation Bitmap DirectoryEntry Structure

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 7.1.1 defines its contents.
BitmapFlags	1	1	This field is mandatory and Section 7.1.2 defines its contents.
Reserved	2	18	This field is mandatory and its contents are reserved.
FirstCluster	20	4	This field is mandatory and Section 7.1.3 defines its contents.
DataLength	24	8	This field is mandatory and Section 7.1.4 defines its contents.

7.1.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1).

7.1.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.1).

For an Allocation Bitmap directory entry, the valid value for this field is 1.

7.1.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.2).

For an Allocation Bitmap directory entry, the valid value for this field is 0.

7.1.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.3).

7.1.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.4).

7.1.2 BitmapFlags Field

The BitmapFlags field contains flags (see Table 21).

Table 21 BitmapFlags Field Structure

	Offset	**Size**	
Field Name	**(bit)**	**(bits)**	**Comments**
BitmapIdentifier	0	1	This field is mandatory and Section 7.1.2.1 defines its contents.
Reserved	1	7	This field is mandatory and its contents are reserved.

7.1.2.1 BitmapIdentifier Field

The BitmapIdentifier field shall indicate which Allocation Bitmap the given directory entry describes. Implementations shall use the First Allocation Bitmap in conjunction with the First FAT and shall use the Second Allocation Bitmap in conjunction with the Second FAT. The ActiveFat field describes which FAT and Allocation Bitmap are active.

The valid values for this field shall be:

- 0, which means the given directory entry describes the First Allocation Bitmap
- 1, which means the given directory entry describes the Second Allocation Bitmap and is possible only when NumberOfFats contains the value 2

7.1.3 FirstCluster Field

The FirstCluster field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.5).

This field contains the index of the first cluster of the cluster chain, as the FAT describes, which hosts the Allocation Bitmap.

7.1.4 DataLength Field

The DataCluster field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.6).

7.1.5 Allocation Bitmap

An Allocation Bitmap records the allocation state of the clusters in the Cluster Heap. Each bit in an Allocation Bitmap indicates whether its corresponding cluster is available for allocation or not.

An Allocation Bitmap represents clusters from lowest to highest index (see Table 22). For historical reasons, the first cluster has index 2. Note: the first bit in the bitmap is the lowest-order bit of the first byte.

Table 22 Allocation Bitmap Structure

	Offset	**Size**	
Field Name	**(bit)**	**(bits)**	**Comments**
BitmapEntry[2]	0	1	This field is mandatory and Section 7.1.5.1 defines its contents.
. . .	. . .	. . .	. . .
BitmapEntry[ClusterCount+1]	ClusterCount - 1	1	This field is mandatory and Section 7.1.5.1 defines its contents. Note: the Main and Backup Boot Sectors both contain the ClusterCount field.

Field Name	Offset (bit)	Size (bits)	Comments
Reserved	ClusterCount	(DataLength * 8) – ClusterCount	This field is mandatory and its contents, if any, are reserved. Note: the Main and Backup Boot Sectors both contain the ClusterCount field.

7.1.5.1 BitmapEntry[2] ... BitmapEntry[ClusterCount+1] Fields

Each BitmapEntry field in this array represents a cluster in the Cluster Heap. BitmapEntry[2] represents the first cluster in the Cluster Heap and BitmapEntry[ClusterCount+1] represents the last cluster in the Cluster Heap.

The valid values for these fields shall be:

- 0, which describes the corresponding cluster as available for allocation
- 1, which describes the corresponding cluster as not available for allocation (a cluster allocation may already consume the corresponding cluster or the active FAT may describe the corresponding cluster as bad)

7.2 Up-case Table Directory Entry

The Up-case Table defines the conversion from lower-case to upper-case characters. This is important due to the File Name directory entry (see Section 7.7) using Unicode characters and the exFAT file system being case insensitive and case preserving. The Up-case Table exists in the Cluster Heap (see Section 7.2.5) and has a corresponding critical primary directory entry in the root directory (see Table 23). The valid number of Up-case Table directory entries is 1.

Due to the relationship between the Up-case Table and file names, implementations should not modify the Up-case Table, except as a result of format operations.

Table 23 Up-case Table DirectoryEntry Structure

Field Name	Offset (byte)	Size (byte)	Comments

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 7.2.1 defines its contents.
Reserved1	1	3	This field is mandatory and its contents are reserved.
TableChecksum	4	4	This field is mandatory and Section 7.2.2 defines its contents.
Reserved2	8	12	This field is mandatory and its contents are reserved.
FirstCluster	20	4	This field is mandatory and Section 7.2.3 defines its contents.
DataLength	24	8	This field is mandatory and Section 7.2.4 defines its contents.

7.2.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1).

7.2.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.1).

For the Up-case Table directory entry, the valid value for this field is 2.

7.2.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.2).

For the Up-case Table directory entry, the valid value for this field is 0.

7.2.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.3).

7.2.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.4).

7.2.2 TableChecksum Field

The TableChecksum field contains the checksum of the Up-case Table (which the FirstCluster and DataLength fields describe). Implementations shall verify the contents of this field are valid prior to using the Up-case Table.

Figure 3 TableChecksum Computation

```
UInt32  TableChecksum
(
       UCHAR *  Table,      // points to an in-memory copy of the up-case table
       UInt64   DataLength
)
{
       UInt32   Checksum =  0;
       UInt64   Index;

       for (Index = 0; Index < DataLength; Index++)
       {
            Checksum = ((Checksum&1) ? 0x80000000 : 0) + (Checksum>>1) +
(UInt32)Table[Index];
    }

       return Checksum;
}
```

7.2.3 FirstCluster Field

The FirstCluster field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.5).

This field contains the index of the first cluster of the cluster chain, as the FAT describes, which hosts the Up-case Table.

7.2.4 DataLength Field

The DataCluster field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.6).

7.2.5 Up-case Table

An up-case table is a series of Unicode character mappings. A character mapping consists of a 2-byte field, with the index of the field in the up-case table representing the Unicode character to be up-cased, and the 2-byte field representing the up-cased Unicode character.

The first 128 Unicode characters have mandatory mappings (see Table 24). An up-case table which has any other character mapping for any of the first 128 Unicode characters is invalid.

Implementations which only support characters from the mandatory mapping range may ignore the mappings of the rest of the up-case table. Such implementations shall only use characters from the mandatory mapping range when creating or renaming files (via the File Name directory entry, see Section 7.7). When up-casing existing file names, such implementations shall not up-case characters from the non-mandatory mapping range, but shall leave them intact in the resulting up-cased file name (this is a partial up-casing). When comparing file names, such implementations shall treat file names which differ from the name under comparison only by Unicode characters from the non-mandatory mapping range as equivalent. While such file names are only potentially equivalent, such implementations cannot ensure the fully up-cased file name does not collide with the name under comparison.

Table 24 Mandatory First 128 Up-case Table Entries

Table Index	**Table Entries**							
	+ 0	**+ 1**	**+ 2**	**+ 3**	**+ 4**	**+ 5**	**+ 6**	**+ 7**
0000h	0000h	0001h	0002h	0003h	0004h	0005h	0006h	0007
0008h	0008h	0009h	000Ah	000Bh	000Ch	000Dh	000Eh	000F
0010h	0010h	0011h	0012h	0013h	0014h	0015h	0016h	0017
0018h	0018h	0019h	001Ah	001Bh	001Ch	001Dh	001Eh	001F

Table Index	**Table Entries**							
0020h	0020h	0021h	0022h	0023h	0024h	0025h	0026h	0027
0028h	0028h	0029h	002Ah	002Bh	002Ch	002Dh	002Eh	002F
0030h	0030h	0031h	0032h	0033h	0034h	0035h	0036h	0037
0038h	0038h	0039h	003Ah	003Bh	003Ch	003Dh	003Eh	003F
0040h	0040h	0041h	0042h	0043h	0044h	0045h	0046h	0047
0048h	0048h	0049h	004Ah	004Bh	004Ch	004Dh	004Eh	004F
0050h	0050h	0051h	0052h	0053h	0054h	0055h	0056h	0057
0058h	0058h	0059h	005Ah	005Bh	005Ch	005Dh	005Eh	005F
0060h	0060h	**0041h**	**0042h**	**0043h**	**0044h**	**0045h**	**0046h**	**0047**
0068h	**0048h**	**0049h**	**004Ah**	**004Bh**	**004Ch**	**004Dh**	**004Eh**	**004F**
0070h	**0050h**	**0051h**	**0052h**	**0053h**	**0054h**	**0055h**	**0056h**	**0057**
0078h	**0058h**	**0059h**	**005Ah**	007Bh	007Ch	007Dh	007Eh	007F

(Note: entries with non-identity up-case mappings are in bold)

Upon formatting a volume, implementations may generate an up-case table in a compressed format using identity-mapping compression, since a large portion of the Unicode character space has no concept of case (which means the "lower-case" and "upper-case" characters are equivalent). Implementations compress an up-case table by representing a series of identity mappings with the value FFFFh followed with the number of identity mappings.

For example, an implementation may represent the first 100 (64h) character mappings with the following eight entries of a compressed up-case table:

> FFFFh, 0061h, 0041h, 0042h, 0043h

The first two entries indicate the first 97 (61h) characters (from 0000h to 0060h) have identity mappings. The subsequent characters, 0061h through 0063h, map to characters 0041h through 0043h, respectively.

The ability to provide a compressed up-case table upon formatting a volume is optional. However, the ability to interpret both an uncompressed and a compressed up-case table is mandatory. The value of the TableChecksum field always conforms to how the up-case table exists on the volume, which may be in either compressed or uncompressed format.

7.2.5.1 Recommended Up-case Table

When formatting a volume, implementations should record the recommended up-case table in compressed format (see Table 25), for which the value of the TableChecksum field is E619D30Dh.

If an implementation defines its own up-case table, either compressed or uncompressed, then that table shall cover the complete Unicode character range (from character codes 0000h to FFFFh inclusive).

Table 25 Recommended Up-case Table in Compressed Format

Raw Offset	**Compressed Table Entries**						
	+ 0	**+ 1**	**+ 2**	**+ 3**	**+ 4**	**+ 5**	**+ 6**
0000h	0000h	0001h	0002h	0003h	0004h	0005h	0006h
0008h	0008h	0009h	000Ah	000Bh	000Ch	000Dh	000Eh
0010h	0010h	0011h	0012h	0013h	0014h	0015h	0016h
0018h	0018h	0019h	001Ah	001Bh	001Ch	001Dh	001Eh
0020h	0020h	0021h	0022h	0023h	0024h	0025h	0026h
0028h	0028h	0029h	002Ah	002Bh	002Ch	002Dh	002Eh
0030h	0030h	0031h	0032h	0033h	0034h	0035h	0036h

Raw Offset	Compressed Table Entries						
0038h	0038h	0039h	003Ah	003Bh	003Ch	003Dh	003Eh
0040h	0040h	0041h	0042h	0043h	0044h	0045h	0046h
0048h	0048h	0049h	004Ah	004Bh	004Ch	004Dh	004Eh
0050h	0050h	0051h	0052h	0053h	0054h	0055h	0056h
0058h	0058h	0059h	005Ah	005Bh	005Ch	005Dh	005Eh
0060h	0060h	0041h	0042h	0043h	0044h	0045h	0046h
0068h	0048h	0049h	004Ah	004Bh	004Ch	004Dh	004Eh
0070h	0050h	0051h	0052h	0053h	0054h	0055h	0056h
0078h	0058h	0059h	005Ah	007Bh	007Ch	007Dh	007Eh
0080h	0080h	0081h	0082h	0083h	0084h	0085h	0086h
0088h	0088h	0089h	008Ah	008Bh	008Ch	008Dh	008Eh
0090h	0090h	0091h	0092h	0093h	0094h	0095h	0096h
0098h	0098h	0099h	009Ah	009Bh	009Ch	009Dh	009Eh
00A0h	00A0h	00A1h	00A2h	00A3h	00A4h	00A5h	00A6h
00A8h	00A8h	00A9h	00AAh	00ABh	00ACh	00ADh	00AEh
00B0h	00B0h	00B1h	00B2h	00B3h	00B4h	00B5h	00B6h
00B8h	00B8h	00B9h	00BAh	00BBh	00BCh	00BDh	00BEh
00C0h	00C0h	00C1h	00C2h	00C3h	00C4h	00C5h	00C6h
00C8h	00C8h	00C9h	00CAh	00CBh	00CCh	00CDh	00CEh

Raw Offset	Compressed Table Entries						
00D0h	00D0h	00D1h	00D2h	00D3h	00D4h	00D5h	00D6h
00D8h	00D8h	00D9h	00DAh	00DBh	00DCh	00DDh	00DEh
00E0h	00C0h	00C1h	00C2h	00C3h	00C4h	00C5h	00C6h
00E8h	00C8h	00C9h	00CAh	00CBh	00CCh	00CDh	00CEh
00F0h	00D0h	00D1h	00D2h	00D3h	00D4h	00D5h	00D6h
00F8h	00D8h	00D9h	00DAh	00DBh	00DCh	00DDh	00DEh
0100h	0100h	0100h	0102h	0102h	0104h	0104h	0106h
0108h	0108h	0108h	010Ah	010Ah	010Ch	010Ch	010Eh
0110h	0110h	0110h	0112h	0112h	0114h	0114h	0116h
0118h	0118h	0118h	011Ah	011Ah	011Ch	011Ch	011Eh
0120h	0120h	0120h	0122h	0122h	0124h	0124h	0126h
0128h	0128h	0128h	012Ah	012Ah	012Ch	012Ch	012Eh
0130h	0130h	0131h	0132h	0132h	0134h	0134h	0136h
0138h	0138h	0139h	0139h	013Bh	013Bh	013Dh	013Dh
0140h	013Fh	0141h	0141h	0143h	0143h	0145h	0145h
0148h	0147h	0149h	014Ah	014Ah	014Ch	014Ch	014Eh
0150h	0150h	0150h	0152h	0152h	0154h	0154h	0156h
0158h	0158h	0158h	015Ah	015Ah	015Ch	015Ch	015Eh
0160h	0160h	0160h	0162h	0162h	0164h	0164h	0166h

Raw Offset	Compressed Table Entries						
0168h	0168h	0168h	016Ah	016Ah	016Ch	016Ch	016Eh
0170h	0170h	0170h	0172h	0172h	0174h	0174h	0176h
0178h	0178h	0179h	0179h	017Bh	017Bh	017Dh	017Dh
0180h	0243h	0181h	0182h	0182h	0184h	0184h	0186h
0188h	0187h	0189h	018Ah	018Bh	018Bh	018Dh	018Eh
0190h	0190h	0191h	0191h	0193h	0194h	01F6h	0196h
0198h	0198h	0198h	023Dh	019Bh	019Ch	019Dh	0220h
01A0h	01A0h	01A0h	01A2h	01A2h	01A4h	01A4h	01A6h
01A8h	01A7h	01A9h	01AAh	01ABh	01ACh	01ACh	01AEh
01B0h	01AFh	01B1h	01B2h	01B3h	01B3h	01B5h	01B5h
01B8h	01B8h	01B8h	01BAh	01BBh	01BCh	01BCh	01BEh
01C0h	01C0h	01C1h	01C2h	01C3h	01C4h	01C5h	01C4h
01C8h	01C8h	01C7h	01CAh	01CBh	01CAh	01CDh	01CDh
01D0h	01CFh	01D1h	01D1h	01D3h	01D3h	01D5h	01D5h
01D8h	01D7h	01D9h	01D9h	01DBh	01DBh	018Eh	01DEh
01E0h	01E0h	01E0h	01E2h	01E2h	01E4h	01E4h	01E6h
01E8h	01E8h	01E8h	01EAh	01EAh	01ECh	01ECh	01EEh
01F0h	01F0h	01F1h	01F2h	01F1h	01F4h	01F4h	01F6h
01F8h	01F8h	01F8h	01FAh	01FAh	01FCh	01FCh	01FEh

Raw Offset	Compressed Table Entries						
0200h	0200h	0200h	0202h	0202h	0204h	0204h	0206h
0208h	0208h	0208h	020Ah	020Ah	020Ch	020Ch	020Eh
0210h	0210h	0210h	0212h	0212h	0214h	0214h	0216h
0218h	0218h	0218h	021Ah	021Ah	021Ch	021Ch	021Eh
0220h	0220h	0221h	0222h	0222h	0224h	0224h	0226h
0228h	0228h	0228h	022Ah	022Ah	022Ch	022Ch	022Eh
0230h	0230h	0230h	0232h	0232h	0234h	0235h	0236h
0238h	0238h	0239h	2C65h	023Bh	023Bh	023Dh	2C66h
0240h	0240h	0241h	0241h	0243h	0244h	0245h	0246h
0248h	0248h	0248h	024Ah	024Ah	024Ch	024Ch	024Eh
0250h	0250h	0251h	0252h	0181h	0186h	0255h	0189h
0258h	0258h	018Fh	025Ah	0190h	025Ch	025Dh	025Eh
0260h	0193h	0261h	0262h	0194h	0264h	0265h	0266h
0268h	0197h	0196h	026Ah	2C62h	026Ch	026Dh	026Eh
0270h	0270h	0271h	019Dh	0273h	0274h	019Fh	0276h
0278h	0278h	0279h	027Ah	027Bh	027Ch	2C64h	027Eh
0280h	01A6h	0281h	0282h	01A9h	0284h	0285h	0286h
0288h	01AEh	0244h	01B1h	01B2h	0245h	028Dh	028Eh
0290h	0290h	0291h	01B7h	0293h	0294h	0295h	0296h

Raw Offset	Compressed Table Entries						
0298h	0298h	0299h	029Ah	029Bh	029Ch	029Dh	029Eh
02A0h	02A0h	02A1h	02A2h	02A3h	02A4h	02A5h	02A6h
02A8h	02A8h	02A9h	02AAh	02ABh	02ACh	02ADh	02AEh
02B0h	02B0h	02B1h	02B2h	02B3h	02B4h	02B5h	02B6h
02B8h	02B8h	02B9h	02BAh	02BBh	02BCh	02BDh	02BEh
02C0h	02C0h	02C1h	02C2h	02C3h	02C4h	02C5h	02C6h
02C8h	02C8h	02C9h	02CAh	02CBh	02CCh	02CDh	02CEh
02D0h	02D0h	02D1h	02D2h	02D3h	02D4h	02D5h	02D6h
02D8h	02D8h	02D9h	02DAh	02DBh	02DCh	02DDh	02DEh
02E0h	02E0h	02E1h	02E2h	02E3h	02E4h	02E5h	02E6h
02E8h	02E8h	02E9h	02EAh	02EBh	02ECh	02EDh	02EEh
02F0h	02F0h	02F1h	02F2h	02F3h	02F4h	02F5h	02F6h
02F8h	02F8h	02F9h	02FAh	02FBh	02FCh	02FDh	02FEh
0300h	0300h	0301h	0302h	0303h	0304h	0305h	0306h
0308h	0308h	0309h	030Ah	030Bh	030Ch	030Dh	030Eh
0310h	0310h	0311h	0312h	0313h	0314h	0315h	0316h
0318h	0318h	0319h	031Ah	031Bh	031Ch	031Dh	031Eh
0320h	0320h	0321h	0322h	0323h	0324h	0325h	0326h
0328h	0328h	0329h	032Ah	032Bh	032Ch	032Dh	032Eh

Raw Offset	Compressed Table Entries						
0330h	0330h	0331h	0332h	0333h	0334h	0335h	0336h
0338h	0338h	0339h	033Ah	033Bh	033Ch	033Dh	033Eh
0340h	0340h	0341h	0342h	0343h	0344h	0345h	0346h
0348h	0348h	0349h	034Ah	034Bh	034Ch	034Dh	034Eh
0350h	0350h	0351h	0352h	0353h	0354h	0355h	0356h
0358h	0358h	0359h	035Ah	035Bh	035Ch	035Dh	035Eh
0360h	0360h	0361h	0362h	0363h	0364h	0365h	0366h
0368h	0368h	0369h	036Ah	036Bh	036Ch	036Dh	036Eh
0370h	0370h	0371h	0372h	0373h	0374h	0375h	0376h
0378h	0378h	0379h	037Ah	03FDh	03FEh	03FFh	037Eh
0380h	0380h	0381h	0382h	0383h	0384h	0385h	0386h
0388h	0388h	0389h	038Ah	038Bh	038Ch	038Dh	038Eh
0390h	0390h	0391h	0392h	0393h	0394h	0395h	0396h
0398h	0398h	0399h	039Ah	039Bh	039Ch	039Dh	039Eh
03A0h	03A0h	03A1h	03A2h	03A3h	03A4h	03A5h	03A6h
03A8h	03A8h	03A9h	03AAh	03ABh	0386h	0388h	0389h
03B0h	03B0h	0391h	0392h	0393h	0394h	0395h	0396h
03B8h	0398h	0399h	039Ah	039Bh	039Ch	039Dh	039Eh
03C0h	03A0h	03A1h	03A3h	03A3h	03A4h	03A5h	03A6h

Raw Offset	**Compressed Table Entries**						
03C8h	03A8h	03A9h	03AAh	03ABh	038Ch	038Eh	038Fh
03D0h	03D0h	03D1h	03D2h	03D3h	03D4h	03D5h	03D6h
03D8h	03D8h	03D8h	03DAh	03DAh	03DCh	03DCh	03DEh
03E0h	03E0h	03E0h	03E2h	03E2h	03E4h	03E4h	03E6h
03E8h	03E8h	03E8h	03EAh	03EAh	03ECh	03ECh	03EEh
03F0h	03F0h	03F1h	03F9h	03F3h	03F4h	03F5h	03F6h
03F8h	03F7h	03F9h	03FAh	03FAh	03FCh	03FDh	03FEh
0400h	0400h	0401h	0402h	0403h	0404h	0405h	0406h
0408h	0408h	0409h	040Ah	040Bh	040Ch	040Dh	040Eh
0410h	0410h	0411h	0412h	0413h	0414h	0415h	0416h
0418h	0418h	0419h	041Ah	041Bh	041Ch	041Dh	041Eh
0420h	0420h	0421h	0422h	0423h	0424h	0425h	0426h
0428h	0428h	0429h	042Ah	042Bh	042Ch	042Dh	042Eh
0430h	0410h	0411h	0412h	0413h	0414h	0415h	0416h
0438h	0418h	0419h	041Ah	041Bh	041Ch	041Dh	041Eh
0440h	0420h	0421h	0422h	0423h	0424h	0425h	0426h
0448h	0428h	0429h	042Ah	042Bh	042Ch	042Dh	042Eh
0450h	0400h	0401h	0402h	0403h	0404h	0405h	0406h
0458h	0408h	0409h	040Ah	040Bh	040Ch	040Dh	040Eh

Raw Offset	Compressed Table Entries						
0460h	0460h	0460h	0462h	0462h	0464h	0464h	0466h
0468h	0468h	0468h	046Ah	046Ah	046Ch	046Ch	046Eh
0470h	0470h	0470h	0472h	0472h	0474h	0474h	0476h
0478h	0478h	0478h	047Ah	047Ah	047Ch	047Ch	047Eh
0480h	0480h	0480h	0482h	0483h	0484h	0485h	0486h
0488h	0488h	0489h	048Ah	048Ah	048Ch	048Ch	048Eh
0490h	0490h	0490h	0492h	0492h	0494h	0494h	0496h
0498h	0498h	0498h	049Ah	049Ah	049Ch	049Ch	049Eh
04A0h	04A0h	04A0h	04A2h	04A2h	04A4h	04A4h	04A6h
04A8h	04A8h	04A8h	04AAh	04AAh	04ACh	04ACh	04AEh
04B0h	04B0h	04B0h	04B2h	04B2h	04B4h	04B4h	04B6h
04B8h	04B8h	04B8h	04BAh	04BAh	04BCh	04BCh	04BEh
04C0h	04C0h	04C1h	04C1h	04C3h	04C3h	04C5h	04C5h
04C8h	04C7h	04C9h	04C9h	04CBh	04CBh	04CDh	04CDh
04D0h	04D0h	04D0h	04D2h	04D2h	04D4h	04D4h	04D6h
04D8h	04D8h	04D8h	04DAh	04DAh	04DCh	04DCh	04DEh
04E0h	04E0h	04E0h	04E2h	04E2h	04E4h	04E4h	04E6h
04E8h	04E8h	04E8h	04EAh	04EAh	04ECh	04ECh	04EEh
04F0h	04F0h	04F0h	04F2h	04F2h	04F4h	04F4h	04F6h

Raw Offset	Compressed Table Entries						
04F8h	04F8h	04F8h	04FAh	04FAh	04FCh	04FCh	04FEh
0500h	0500h	0500h	0502h	0502h	0504h	0504h	0506h
0508h	0508h	0508h	050Ah	050Ah	050Ch	050Ch	050Eh
0510h	0510h	0510h	0512h	0512h	0514h	0515h	0516h
0518h	0518h	0519h	051Ah	051Bh	051Ch	051Dh	051Eh
0520h	0520h	0521h	0522h	0523h	0524h	0525h	0526h
0528h	0528h	0529h	052Ah	052Bh	052Ch	052Dh	052Eh
0530h	0530h	0531h	0532h	0533h	0534h	0535h	0536h
0538h	0538h	0539h	053Ah	053Bh	053Ch	053Dh	053Eh
0540h	0540h	0541h	0542h	0543h	0544h	0545h	0546h
0548h	0548h	0549h	054Ah	054Bh	054Ch	054Dh	054Eh
0550h	0550h	0551h	0552h	0553h	0554h	0555h	0556h
0558h	0558h	0559h	055Ah	055Bh	055Ch	055Dh	055Eh
0560h	0560h	0531h	0532h	0533h	0534h	0535h	0536h
0568h	0538h	0539h	053Ah	053Bh	053Ch	053Dh	053Eh
0570h	0540h	0541h	0542h	0543h	0544h	0545h	0546h
0578h	0548h	0549h	054Ah	054Bh	054Ch	054Dh	054Eh
0580h	0550h	0551h	0552h	0553h	0554h	0555h	0556h
0588h	17F6h	2C63h	1D7Eh	1D7Fh	1D80h	1D81h	1D82h

Raw Offset	Compressed Table Entries						
0590h	1D84h	1D85h	1D86h	1D87h	1D88h	1D89h	1D8Ah
0598h	1D8Ch	1D8Dh	1D8Eh	1D8Fh	1D90h	1D91h	1D92h
05A0h	1D94h	1D95h	1D96h	1D97h	1D98h	1D99h	1D9Ah
05A8h	1D9Ch	1D9Dh	1D9Eh	1D9Fh	1DA0h	1DA1h	1DA2h
05B0h	1DA4h	1DA5h	1DA6h	1DA7h	1DA8h	1DA9h	1DAAh
05B8h	1DACh	1DADh	1DAEh	1DAFh	1DB0h	1DB1h	1DB2h
05C0h	1DB4h	1DB5h	1DB6h	1DB7h	1DB8h	1DB9h	1DBAh
05C8h	1DBCh	1DBDh	1DBEh	1DBFh	1DC0h	1DC1h	1DC2h
05D0h	1DC4h	1DC5h	1DC6h	1DC7h	1DC8h	1DC9h	1DCAh
05D8h	1DCCh	1DCDh	1DCEh	1DCFh	1DD0h	1DD1h	1DD2h
05E0h	1DD4h	1DD5h	1DD6h	1DD7h	1DD8h	1DD9h	1DDAh
05E8h	1DDCh	1DDDh	1DDEh	1DDFh	1DE0h	1DE1h	1DE2h
05F0h	1DE4h	1DE5h	1DE6h	1DE7h	1DE8h	1DE9h	1DEAh
05F8h	1DECh	1DEDh	1DEEh	1DEFh	1DF0h	1DF1h	1DF2h
0600h	1DF4h	1DF5h	1DF6h	1DF7h	1DF8h	1DF9h	1DFAh
0608h	1DFCh	1DFDh	1DFEh	1DFFh	1E00h	1E00h	1E02h
0610h	1E04h	1E04h	1E06h	1E06h	1E08h	1E08h	1E0Ah
0618h	1E0Ch	1E0Ch	1E0Eh	1E0Eh	1E10h	1E10h	1E12h
0620h	1E14h	1E14h	1E16h	1E16h	1E18h	1E18h	1E1Ah

Raw Offset	Compressed Table Entries						
0628h	1E1Ch	1E1Ch	1E1Eh	1E1Eh	1E20h	1E20h	1E22h
0630h	1E24h	1E24h	1E26h	1E26h	1E28h	1E28h	1E2Ah
0638h	1E2Ch	1E2Ch	1E2Eh	1E2Eh	1E30h	1E30h	1E32h
0640h	1E34h	1E34h	1E36h	1E36h	1E38h	1E38h	1E3Ah
0648h	1E3Ch	1E3Ch	1E3Eh	1E3Eh	1E40h	1E40h	1E42h
0650h	1E44h	1E44h	1E46h	1E46h	1E48h	1E48h	1E4Ah
0658h	1E4Ch	1E4Ch	1E4Eh	1E4Eh	1E50h	1E50h	1E52h
0660h	1E54h	1E54h	1E56h	1E56h	1E58h	1E58h	1E5Ah
0668h	1E5Ch	1E5Ch	1E5Eh	1E5Eh	1E60h	1E60h	1E62h
0670h	1E64h	1E64h	1E66h	1E66h	1E68h	1E68h	1E6Ah
0678h	1E6Ch	1E6Ch	1E6Eh	1E6Eh	1E70h	1E70h	1E72h
0680h	1E74h	1E74h	1E76h	1E76h	1E78h	1E78h	1E7Ah
0688h	1E7Ch	1E7Ch	1E7Eh	1E7Eh	1E80h	1E80h	1E82h
0690h	1E84h	1E84h	1E86h	1E86h	1E88h	1E88h	1E8Ah
0698h	1E8Ch	1E8Ch	1E8Eh	1E8Eh	1E90h	1E90h	1E92h
06A0h	1E94h	1E94h	1E96h	1E97h	1E98h	1E99h	1E9Ah
06A8h	1E9Ch	1E9Dh	1E9Eh	1E9Fh	1EA0h	1EA0h	1EA2h
06B0h	1EA4h	1EA4h	1EA6h	1EA6h	1EA8h	1EA8h	1EAAh
06B8h	1EACh	1EACh	1EAEh	1EAEh	1EB0h	1EB0h	1EB2h

Raw Offset	Compressed Table Entries						
06C0h	1EB4h	1EB4h	1EB6h	1EB6h	1EB8h	1EB8h	1EBAh
06C8h	1EBCh	1EBCh	1EBEh	1EBEh	1EC0h	1EC0h	1EC2h
06D0h	1EC4h	1EC4h	1EC6h	1EC6h	1EC8h	1EC8h	1ECAh
06D8h	1ECCh	1ECCh	1ECEh	1ECEh	1ED0h	1ED0h	1ED2h
06E0h	1ED4h	1ED4h	1ED6h	1ED6h	1ED8h	1ED8h	1EDAh
06E8h	1EDCh	1EDCh	1EDEh	1EDEh	1EE0h	1EE0h	1EE2h
06F0h	1EE4h	1EE4h	1EE6h	1EE6h	1EE8h	1EE8h	1EEAh
06F8h	1EECh	1EECh	1EEEh	1EEEh	1EF0h	1EF0h	1EF2h
0700h	1EF4h	1EF4h	1EF6h	1EF6h	1EF8h	1EF8h	1EFAh
0708h	1EFCh	1EFDh	1EFEh	1EFFh	1F08h	1F09h	1F0Ah
0710h	1F0Ch	1F0Dh	1F0Eh	1F0Fh	1F08h	1F09h	1F0Ah
0718h	1F0Ch	1F0Dh	1F0Eh	1F0Fh	1F18h	1F19h	1F1Ah
0720h	1F1Ch	1F1Dh	1F16h	1F17h	1F18h	1F19h	1F1Ah
0728h	1F1Ch	1F1Dh	1F1Eh	1F1Fh	1F28h	1F29h	1F2Ah
0730h	1F2Ch	1F2Dh	1F2Eh	1F2Fh	1F28h	1F29h	1F2Ah
0738h	1F2Ch	1F2Dh	1F2Eh	1F2Fh	1F38h	1F39h	1F3Ah
0740h	1F3Ch	1F3Dh	1F3Eh	1F3Fh	1F38h	1F39h	1F3Ah
0748h	1F3Ch	1F3Dh	1F3Eh	1F3Fh	1F48h	1F49h	1F4Ah
0750h	1F4Ch	1F4Dh	1F46h	1F47h	1F48h	1F49h	1F4Ah

Raw Offset	Compressed Table Entries						
0758h	1F4Ch	1F4Dh	1F4Eh	1F4Fh	1F50h	1F59h	1F52h
0760h	1F54h	1F5Dh	1F56h	1F5Fh	1F58h	1F59h	1F5Ah
0768h	1F5Ch	1F5Dh	1F5Eh	1F5Fh	1F68h	1F69h	1F6Ah
0770h	1F6Ch	1F6Dh	1F6Eh	1F6Fh	1F68h	1F69h	1F6Ah
0778h	1F6Ch	1F6Dh	1F6Eh	1F6Fh	1FBAh	1FBBh	1FC8h
0780h	1FCAh	1FCBh	1FDAh	1FDBh	1FF8h	1FF9h	1FEAh
0788h	1FFAh	1FFBh	1F7Eh	1F7Fh	1F88h	1F89h	1F8Ah
0790h	1F8Ch	1F8Dh	1F8Eh	1F8Fh	1F88h	1F89h	1F8Ah
0798h	1F8Ch	1F8Dh	1F8Eh	1F8Fh	1F98h	1F99h	1F9Ah
07A0h	1F9Ch	1F9Dh	1F9Eh	1F9Fh	1F98h	1F99h	1F9Ah
07A8h	1F9Ch	1F9Dh	1F9Eh	1F9Fh	1FA8h	1FA9h	1FAAh
07B0h	1FACh	1FADh	1FAEh	1FAFh	1FA8h	1FA9h	1FAAh
07B8h	1FACh	1FADh	1FAEh	1FAFh	1FB8h	1FB9h	1FB2h
07C0h	1FB4h	1FB5h	1FB6h	1FB7h	1FB8h	1FB9h	1FBAh
07C8h	1FBCh	1FBDh	1FBEh	1FBFh	1FC0h	1FC1h	1FC2h
07D0h	1FC4h	1FC5h	1FC6h	1FC7h	1FC8h	1FC9h	1FCAh
07D8h	1FC3h	1FCDh	1FCEh	1FCFh	1FD8h	1FD9h	1FD2h
07E0h	1FD4h	1FD5h	1FD6h	1FD7h	1FD8h	1FD9h	1FDAh
07E8h	1FDCh	1FDDh	1FDEh	1FDFh	1FE8h	1FE9h	1FE2h

Raw Offset	Compressed Table Entries						
07F0h	1FE4h	1FECh	1FE6h	1FE7h	1FE8h	1FE9h	1FEAh
07F8h	1FECh	1FEDh	1FEEh	1FEFh	1FF0h	1FF1h	1FF2h
0800h	1FF4h	1FF5h	1FF6h	1FF7h	1FF8h	1FF9h	1FFAh
0808h	1FF3h	1FFDh	1FFEh	1FFFh	2000h	2001h	2002h
0810h	2004h	2005h	2006h	2007h	2008h	2009h	200Ah
0818h	200Ch	200Dh	200Eh	200Fh	2010h	2011h	2012h
0820h	2014h	2015h	2016h	2017h	2018h	2019h	201Ah
0828h	201Ch	201Dh	201Eh	201Fh	2020h	2021h	2022h
0830h	2024h	2025h	2026h	2027h	2028h	2029h	202Ah
0838h	202Ch	202Dh	202Eh	202Fh	2030h	2031h	2032h
0840h	2034h	2035h	2036h	2037h	2038h	2039h	203Ah
0848h	203Ch	203Dh	203Eh	203Fh	2040h	2041h	2042h
0850h	2044h	2045h	2046h	2047h	2048h	2049h	204Ah
0858h	204Ch	204Dh	204Eh	204Fh	2050h	2051h	2052h
0860h	2054h	2055h	2056h	2057h	2058h	2059h	205Ah
0868h	205Ch	205Dh	205Eh	205Fh	2060h	2061h	2062h
0870h	2064h	2065h	2066h	2067h	2068h	2069h	206Ah
0878h	206Ch	206Dh	206Eh	206Fh	2070h	2071h	2072h
0880h	2074h	2075h	2076h	2077h	2078h	2079h	207Ah

Raw Offset	Compressed Table Entries						
0888h	207Ch	207Dh	207Eh	207Fh	2080h	2081h	2082h
0890h	2084h	2085h	2086h	2087h	2088h	2089h	208Ah
0898h	208Ch	208Dh	208Eh	208Fh	2090h	2091h	2092h
08A0h	2094h	2095h	2096h	2097h	2098h	2099h	209Ah
08A8h	209Ch	209Dh	209Eh	209Fh	20A0h	20A1h	20A2h
08B0h	20A4h	20A5h	20A6h	20A7h	20A8h	20A9h	20AAh
08B8h	20ACh	20ADh	20AEh	20AFh	20B0h	20B1h	20B2h
08C0h	20B4h	20B5h	20B6h	20B7h	20B8h	20B9h	20BAh
08C8h	20BCh	20BDh	20BEh	20BFh	20C0h	20C1h	20C2h
08D0h	20C4h	20C5h	20C6h	20C7h	20C8h	20C9h	20CAh
08D8h	20CCh	20CDh	20CEh	20CFh	20D0h	20D1h	20D2h
08E0h	20D4h	20D5h	20D6h	20D7h	20D8h	20D9h	20DAh
08E8h	20DCh	20DDh	20DEh	20DFh	20E0h	20E1h	20E2h
08F0h	20E4h	20E5h	20E6h	20E7h	20E8h	20E9h	20EAh
08F8h	20ECh	20EDh	20EEh	20EFh	20F0h	20F1h	20F2h
0900h	20F4h	20F5h	20F6h	20F7h	20F8h	20F9h	20FAh
0908h	20FCh	20FDh	20FEh	20FFh	2100h	2101h	2102h
0910h	2104h	2105h	2106h	2107h	2108h	2109h	210Ah
0918h	210Ch	210Dh	210Eh	210Fh	2110h	2111h	2112h

Raw Offset	Compressed Table Entries						
0920h	2114h	2115h	2116h	2117h	2118h	2119h	211Ah
0928h	211Ch	211Dh	211Eh	211Fh	2120h	2121h	2122h
0930h	2124h	2125h	2126h	2127h	2128h	2129h	212Ah
0938h	212Ch	212Dh	212Eh	212Fh	2130h	2131h	2132h
0940h	2134h	2135h	2136h	2137h	2138h	2139h	213Ah
0948h	213Ch	213Dh	213Eh	213Fh	2140h	2141h	2142h
0950h	2144h	2145h	2146h	2147h	2148h	2149h	214Ah
0958h	214Ch	214Dh	2132h	214Fh	2150h	2151h	2152h
0960h	2154h	2155h	2156h	2157h	2158h	2159h	215Ah
0968h	215Ch	215Dh	215Eh	215Fh	2160h	2161h	2162h
0970h	2164h	2165h	2166h	2167h	2168h	2169h	216Ah
0978h	216Ch	216Dh	216Eh	216Fh	2160h	2161h	2162h
0980h	2164h	2165h	2166h	2167h	2168h	2169h	216Ah
0988h	216Ch	216Dh	216Eh	216Fh	2180h	2181h	2182h
0990h	2183h	FFFFh	034Bh	24B6h	24B7h	24B8h	24B9h
0998h	24BBh	24BCh	24BDh	24BEh	24BFh	24C0h	24C1h
09A0h	24C3h	24C4h	24C5h	24C6h	24C7h	24C8h	24C9h
09A8h	24CBh	24CCh	24CDh	24CEh	24CFh	FFFFh	0746h
09B0h	2C01h	2C02h	2C03h	2C04h	2C05h	2C06h	2C07h

Raw Offset	Compressed Table Entries						
09B8h	2C09h	2C0Ah	2C0Bh	2C0Ch	2C0Dh	2C0Eh	2C0Fh
09C0h	2C11h	2C12h	2C13h	2C14h	2C15h	2C16h	2C17h
09C8h	2C19h	2C1Ah	2C1Bh	2C1Ch	2C1Dh	2C1Eh	2C1Fh
09D0h	2C21h	2C22h	2C23h	2C24h	2C25h	2C26h	2C27h
09D8h	2C29h	2C2Ah	2C2Bh	2C2Ch	2C2Dh	2C2Eh	2C5Fh
09E0h	2C60h	2C62h	2C63h	2C64h	2C65h	2C66h	2C67h
09E8h	2C69h	2C69h	2C6Bh	2C6Bh	2C6Dh	2C6Eh	2C6Fh
09F0h	2C71h	2C72h	2C73h	2C74h	2C75h	2C75h	2C77h
09F8h	2C79h	2C7Ah	2C7Bh	2C7Ch	2C7Dh	2C7Eh	2C7Fh
0A00h	2C80h	2C82h	2C82h	2C84h	2C84h	2C86h	2C86h
0A08h	2C88h	2C8Ah	2C8Ah	2C8Ch	2C8Ch	2C8Eh	2C8Eh
0A10h	2C90h	2C92h	2C92h	2C94h	2C94h	2C96h	2C96h
0A18h	2C98h	2C9Ah	2C9Ah	2C9Ch	2C9Ch	2C9Eh	2C9Eh
0A20h	2CA0h	2CA2h	2CA2h	2CA4h	2CA4h	2CA6h	2CA6h
0A28h	2CA8h	2CAAh	2CAAh	2CACh	2CACh	2CAEh	2CAEh
0A30h	2CB0h	2CB2h	2CB2h	2CB4h	2CB4h	2CB6h	2CB6h
0A38h	2CB8h	2CBAh	2CBAh	2CBCh	2CBCh	2CBEh	2CBEh
0A40h	2CC0h	2CC2h	2CC2h	2CC4h	2CC4h	2CC6h	2CC6h
0A48h	2CC8h	2CCAh	2CCAh	2CCCh	2CCCh	2CCEh	2CCEh

Raw Offset	Compressed Table Entries						
0A50h	2CD0h	2CD2h	2CD2h	2CD4h	2CD4h	2CD6h	2CD6h
0A58h	2CD8h	2CDAh	2CDAh	2CDCh	2CDCh	2CDEh	2CDEh
0A60h	2CE0h	2CE2h	2CE2h	2CE4h	2CE5h	2CE6h	2CE7h
0A68h	2CE9h	2CEAh	2CEBh	2CECh	2CEDh	2CEEh	2CEFh
0A70h	2CF1h	2CF2h	2CF3h	2CF4h	2CF5h	2CF6h	2CF7h
0A78h	2CF9h	2CFAh	2CFBh	2CFCh	2CFDh	2CFEh	2CFFh
0A80h	10A1h	10A2h	10A3h	10A4h	10A5h	10A6h	10A7h
0A88h	10A9h	10AAh	10ABh	10ACh	10ADh	10AEh	10AFh
0A90h	10B1h	10B2h	10B3h	10B4h	10B5h	10B6h	10B7h
0A98h	10B9h	10BAh	10BBh	10BCh	10BDh	10BEh	10BFh
0AA0h	10C1h	10C2h	10C3h	10C4h	10C5h	FFFFh	D21Bh
0AA8h	FF22h	FF23h	FF24h	FF25h	FF26h	FF27h	FF28h
0AB0h	FF2Ah	FF2Bh	FF2Ch	FF2Dh	FF2Eh	FF2Fh	FF30h
0AB8h	FF32h	FF33h	FF34h	FF35h	FF36h	FF37h	FF38h
0AC0h	FF3Ah	FF5Bh	FF5Ch	FF5Dh	FF5Eh	FF5Fh	FF60h
0AC8h	FF62h	FF63h	FF64h	FF65h	FF66h	FF67h	FF68h
0AD0h	FF6Ah	FF6Bh	FF6Ch	FF6Dh	FF6Eh	FF6Fh	FF70h
0AD8h	FF72h	FF73h	FF74h	FF75h	FF76h	FF77h	FF78h
0AE0h	FF7Ah	FF7Bh	FF7Ch	FF7Dh	FF7Eh	FF7Fh	FF80h

Raw Offset	Compressed Table Entries						
0AE8h	FF82h	FF83h	FF84h	FF85h	FF86h	FF87h	FF88h
0AF0h	FF8Ah	FF8Bh	FF8Ch	FF8Dh	FF8Eh	FF8Fh	FF90h
0AF8h	FF92h	FF93h	FF94h	FF95h	FF96h	FF97h	FF98h
0B00h	FF9Ah	FF9Bh	FF9Ch	FF9Dh	FF9Eh	FF9Fh	FFA0h
0B08h	FFA2h	FFA3h	FFA4h	FFA5h	FFA6h	FFA7h	FFA8h
0B10h	FFAAh	FFABh	FFACh	FFADh	FFAEh	FFAFh	FFB0h
0B18h	FFB2h	FFB3h	FFB4h	FFB5h	FFB6h	FFB7h	FFB8h
0B20h	FFBAh	FFBBh	FFBCh	FFBDh	FFBEh	FFBFh	FFC0h
0B28h	FFC2h	FFC3h	FFC4h	FFC5h	FFC6h	FFC7h	FFC8h
0B30h	FFCAh	FFCBh	FFCCh	FFCDh	FFCEh	FFCFh	FFD0h
0B38h	FFD2h	FFD3h	FFD4h	FFD5h	FFD6h	FFD7h	FFD8h
0B40h	FFDAh	FFDBh	FFDCh	FFDDh	FFDEh	FFDFh	FFE0h
0B48h	FFE2h	FFE3h	FFE4h	FFE5h	FFE6h	FFE7h	FFE8h
0B50h	FFEAh	FFEBh	FFECh	FFEDh	FFEEh	FFEFh	FFF0h
0B58h	FFF2h	FFF3h	FFF4h	FFF5h	FFF6h	FFF7h	FFF8h
0B60h	FFFAh	FFFBh	FFFCh	FFFDh	FFFEh	FFFFh	

7.3 Volume Label Directory Entry

The Volume Label is a Unicode string which enables end users to distinguish their storage volumes. In the exFAT file system, the Volume Label exists as a critical primary directory entry in the root directory (see Table 26). The valid number of Volume Label directory entries ranges from 0 to 1.

Table 26 Volume Label DirectoryEntry Structure

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 7.3.1 defines its contents.
CharacterCount	1	1	This field is mandatory and Section 7.3.2 defines its contents.
VolumeLabel	2	22	This field is mandatory and Section 7.3.3 defines its contents.
Reserved	24	8	This field is mandatory and its contents are reserved.

7.3.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1).

7.3.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.1).

For the Volume Label directory entry, the valid value for this field is 3.

7.3.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.2).

For the Volume Label directory entry, the valid value for this field is 0.

7.3.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.3).

7.3.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.4).

7.3.2 CharacterCount Field

The CharacterCount field shall contain the length of the Unicode string the VolumeLabel field contains.

The valid range of values for this field shall be:

- At least 0, which means the Unicode string is 0 characters long (which is the equivalent of no volume label)
- At most 11, which means the Unicode string is 11 characters long

7.3.3 VolumeLabel Field

The VolumeLabel field shall contain a Unicode string, which is the user-friendly name of the volume. The VolumeLabel field has the same set of invalid characters as the FileName field of the File Name directory entry (see Section 7.7.3).

7.4 File Directory Entry

File directory entries describe files and directories. They are critical primary directory entries and any directory may contain zero or more File directory entries (see Table 27). For a File directory entry to be valid, exactly one Stream Extension directory entry and at least one File Name directory entry must immediately follow the File directory entry (see Sections 7.6 and 7.7, respectively).

Table 27 File DirectoryEntry

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 7.4.1 defines its contents.

Field Name	Offset (byte)	Size (byte)	Comments
SecondaryCount	1	1	This field is mandatory and Section 7.4.2 defines its contents.
SetChecksum	2	2	This field is mandatory and Section 7.4.3 defines its contents.
FileAttributes	4	2	This field is mandatory and Section 7.4.4 defines its contents.
Reserved1	6	2	This field is mandatory and its contents are reserved.
CreateTimestamp	8	4	This field is mandatory and Section 7.4.5 defines its contents.
LastModifiedTimestamp	12	4	This field is mandatory and Section 7.4.6 defines its contents.
LastAccessedTimestamp	16	4	This field is mandatory and Section 7.4.7 defines its contents.
Create10msIncrement	20	1	This field is mandatory and Section 7.4.5 defines its contents.
LastModified10msIncrement	21	1	This field is mandatory and Section 7.4.6 defines its contents.
CreateUtcOffset	22	1	This field is mandatory and Section 7.4.5 defines its contents.
LastModifiedUtcOffset	23	1	This field is mandatory and Section 7.4.6 defines its contents.
LastAccessedUtcOffset	24	1	This field is mandatory and Section 7.4.7 defines its contents.
Reserved2	25	7	This field is mandatory and its contents are reserved.

7.4.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1).

7.4.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.1).

For a File directory entry, the valid value for this field is 5.

7.4.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.2).

For a File directory entry, the valid value for this field is 0.

7.4.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.3).

7.4.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.4).

7.4.2 SecondaryCount Field

The SecondaryCount field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.2).

7.4.3 SetChecksum Field

The SetChecksum field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.3).

7.4.4 FileAttributes Field

The FileAttributes field contains flags (see Table 28).

Table 28 FileAttributes Field Structure

Field Name	Offset (bit)	Size (bits)	Comments
ReadOnly	0	1	This field is mandatory and conforms to the MS-DOS definition.
Hidden	1	1	This field is mandatory and conforms to the MS-DOS definition.
System	2	1	This field is mandatory and conforms to the MS-DOS definition.
Reserved1	3	1	This field is mandatory and its contents are reserved.
Directory	4	1	This field is mandatory and conforms to the MS-DOS definition.
Archive	5	1	This field is mandatory and conforms to the MS-DOS definition.
Reserved2	6	10	This field is mandatory and its contents are reserved.

7.4.5 CreateTimestamp, Create10msIncrement, and CreateUtcOffset Fields

In combination, the CreateTimestamp and CreateTime10msIncrement fields shall describe the local date and time the given file/directory was created. The CreateUtcOffset field describes the offset of local date and time from UTC. Implementations shall set these fields upon creation of the given directory entry set.

These fields shall conform to the definitions of the Timestamp, 10msIncrement, and UtcOffset fields (see Sections 7.4.8, 7.4.9, and 7.4.10, respectively).

7.4.6 LastModifiedTimestamp, LastModified10msIncrement, and LastModifiedUtcOffset Fields

In combination, the LastModifiedTimestamp and LastModifiedTime10msIncrement fields shall describe the local date and time the contents of any of the clusters associated with the given Stream Extension directory entry were last modified. The LastModifiedUtcOffset field describes the offset of local date and time from UTC. Implementations shall update these fields:

1. After modifying the contents of any of the clusters associated with the given Stream Extension directory entry (except for contents which exist beyond the point the ValidDataLength field describes)

2. Upon changing the values of either the ValidDataLength or DataLength fields

These fields shall conform to the definitions of the Timestamp, 10msIncrement, and UtcOffset fields (see Sections 7.4.8, 7.4.9, and 7.4.10, respectively).

7.4.7 LastAccessedTimestamp and LastAccessedUtcOffset Fields

The LastAccessedTimestamp field shall describe the local date and time the contents of any of the clusters associated with the given Stream Extension directory entry were last accessed. The LastAccessedUtcOffset field describes the offset of local date and time from UTC. Implementations shall update these fields:

1. After modifying the contents of any of the clusters associated with the given Stream Extension directory entry (except for contents which exist beyond the ValidDataLength)

2. Upon changing the values of either the ValidDataLength or DataLength fields

Implementations should update these fields after reading the contents of any of the clusters associated with the given Stream Extension directory entry.

These fields shall conform to the definitions of the Timestamp and UtcOffset fields (see Sections 7.4.8 and 7.4.10, respectively).

7.4.8 Timestamp Fields

Timestamp fields describe both local date and time, down to a two-second resolution (see Table 29).

Table 29 Timestamp Field Structure

	Offset	Size	
Field Name	(bit)	(bits)	Comments

Field Name	Offset (bit)	Size (bits)	Comments
DoubleSeconds	0	5	This field is mandatory and Section 7.4.8.1 defines its contents.
Minute	5	6	This field is mandatory and Section 7.4.8.2 defines its contents.
Hour	11	5	This field is mandatory and Section 7.4.8.3 defines its contents.
Day	16	5	This field is mandatory and Section 7.4.8.4 defines its contents.
Month	21	4	This field is mandatory and Section 7.4.8.5 defines its contents.
Year	25	7	This field is mandatory and Section 7.4.8.6 defines its contents.

7.4.8.1 DoubleSeconds Field

The DoubleSeconds field shall describe the seconds portion of the Timestamp field, in two-second multiples.

The valid range of values for this field shall be:

- 0, which represents 0 seconds
- 29, which represents 58 seconds

7.4.8.2 Minute Field

The Minute field shall describe the minutes portion of the Timestamp field.

The valid range of values for this field shall be:

- 0, which represents 0 minutes
- 59, which represents 59 minutes

7.4.8.3 Hour Field

The Hour field shall describe the hours portion of the Timestamp field.

The valid range of values for this field shall be:

- 0, which represents 00:00 hours
- 23, which represents 23:00 hours

7.4.8.4 Day Field

The Day field shall describe the day portion of the Timestamp field.

The valid range of values for this field shall be:

- 1, which is the first day of the given month
- The last day of the given month (the given month defines the number of valid days)

7.4.8.5 Month Field

The Month field shall describe the month portion of the Timestamp field.

The valid range of values for this field shall be:

- At least 1, which represents January
- At most 12, which represents December

7.4.8.6 Year Field

The Year field shall describe the year portion of the Timestamp field, relative to the year 1980. This field represents the year 1980 with the value 0 and the year 2107 with the value 127.

All possible values for this field are valid.

7.4.9 10msIncrement Fields

10msIncrement fields shall provide additional time resolution to their corresponding Timestamp fields in ten-millisecond multiples.

The valid range of values for these fields shall be:

- At least 0, which represents 0 milliseconds
- At most 199, which represents 1990 milliseconds

7.4.10 UtcOffset Fields

UtcOffset fields (see Table 30) shall describe the offset from UTC to the local date and time their corresponding Timestamp and 10msIncrement fields describe. The offset from UTC to the local date and time includes the effects of time zones and other date-time adjustments, such as daylight saving and regional summer time changes.

Table 30 UtcOffset Field Structure

	Offset	Size	
Field Name	**(bit)**	**(bits)**	**Comments**
OffsetFromUtc	0	7	This field is mandatory and Section 7.4.10.1 defines its contents.
OffsetValid	7	1	This field is mandatory and Section 7.4.10.2 defines its contents.

7.4.10.1 OffsetFromUtc Field

The OffsetFromUtc field shall describe the offset from UTC of the local date and time the related Timestamp and 10msIncrement fields contains. This field describes the offset from UTC in 15 minute intervals (see Table 31).

Table 31 Meaning of the Values of the OffsetFromUtc Field

Value	**Signed Decimal Equivalent**	**Description**
3Fh	63	Local date and time is UTC + 15:45
3Eh	62	Local date and time is UTC + 15:30
. . .	. . .	. . .

Value	Signed Decimal Equivalent	Description
01h	1	Local date and time is UTC + 00:15
00h	0	Local date and time is UTC
7Fh	-1	Local date and time is UTC – 00:15
. . .	. . .	. . .
41h	-63	Local date and time is UTC – 15:45
40h	-64	Local date and time is UTC – 16:00

As the table above indicates, all possible values for this field are valid. However, implementations should only record the value 00h for this field when:

1. Local date and time are actually the same as UTC, in which case the value of the OffsetValid field shall be 1

2. Local date and time are not known, in which case the value of the OffsetValid field shall be 1 and implementations shall consider UTC to be local date and time

3. UTC is not known, in which case the value of the OffsetValid field shall be 0

If the local date and time offset from UTC happens to not be a multiple of 15 minute intervals, then implementations shall record 00h in the OffsetFromUtc field and shall consider UTC to be local date and time.

7.4.10.2 OffsetValid Field

The OffsetValid field shall describe whether the contents of the OffsetFromUtc field are valid or not, as follows:

- 0, which means the contents of the OffsetFromUtc field are invalid

  > and shall be 00h

- 1, which means the contents of the OffsetFromUtc field are valid

Implementations should only set this field to the value 0 when UTC is not available for computing the value of the OffsetFromUtc field. If this field contains the value 0, then implementations shall treat the Timestamp and 10msIncrement fields as having the same UTC offset as the current local date and time.

7.5 Volume GUID Directory Entry

The Volume GUID directory entry contains a GUID which enables implementations to uniquely and programmatically distinguish volumes. The Volume GUID exists as a benign primary directory entry in the root directory (see Table 32). The valid number of Volume GUID directory entries ranges from 0 to 1.

Table 32 Volume GUID DirectoryEntry

	Offset	Size	
Field Name	**(byte)**	**(byte)**	**Comments**
EntryType	0	1	This field is mandatory and Section 7.5.1 defines its contents.
SecondaryCount	1	1	This field is mandatory and Section 7.5.2 defines its contents.
SetChecksum	2	2	This field is mandatory and Section 7.5.3 defines its contents.
GeneralPrimaryFlags	4	2	This field is mandatory and Section 7.5.4 defines its contents.
VolumeGuid	6	16	This field is mandatory and Section 7.5.5 defines its contents.
Reserved	22	10	This field is mandatory and its contents are reserved.

7.5.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1).

7.5.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.1).

For the Volume GUID directory entry, the valid value for this field is 0.

7.5.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.2).

For the Volume GUID directory entry, the valid value for this field is 1.

7.5.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.3).

7.5.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.1.4).

7.5.2 SecondaryCount Field

The SecondaryCount field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.2).

For the Volume GUID directory entry, the valid value for this field is 0.

7.5.3 SetChecksum Field

The SetChecksum field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.3).

7.5.4 GeneralPrimaryFlags Field

The GeneralPrimaryFlags field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.4) and defines the contents of the CustomDefined field to be reserved.

7.5.4.1 AllocationPossible Field

The AllocationPossible field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.4).

For the Volume GUID directory entry, the valid value for this field is 0.

7.5.4.2 NoFatChain Field

The NoFatChain field shall conform to the definition provided in the Generic Primary DirectoryEntry template (see Section 6.3.4).

7.5.5 VolumeGuid Field

The VolumeGuid field shall contain a GUID which uniquely identifies the given volume.

All possible values for this field are valid, except the null GUID, which is {00000000-0000-0000-0000-000000000000}.

7.6 Stream Extension Directory Entry

The Stream Extension directory entry is a critical secondary directory entry in File directory entry sets (see Table 33). The valid number of Stream Extension directory entries in a File directory entry set is 1. Further, this directory entry is valid only if it immediately follows the File directory entry.

Table 33 Stream Extension DirectoryEntry

	Offset	**Size**	
Field Name	**(byte)**	**(byte)**	**Comments**
EntryType	0	1	This field is mandatory and Section 7.6.1 defines its contents.
GeneralSecondaryFlags	1	1	This field is mandatory and Section 7.6.2 defines its contents.
Reserved1	2	1	This field is mandatory and its contents are reserved.
NameLength	3	1	This field is mandatory and Section 7.6.3 defines its contents.

Field Name	Offset (byte)	Size (byte)	Comments
NameHash	4	2	This field is mandatory and Section 7.6.4 defines its contents.
Reserved2	6	2	This field is mandatory and its contents are reserved.
ValidDataLength	8	8	This field is mandatory and Section 7.6.5 defines its contents.
Reserved3	16	4	This field is mandatory and its contents are reserved.
FirstCluster	20	4	This field is mandatory and Section 7.6.6 defines its contents.
DataLength	24	8	This field is mandatory and Section 7.6.7 defines its contents.

7.6.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1).

7.6.1.1 ypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.1).

For the Stream Extension directory entry, the valid value for this field is 0.

7.6.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.2).

For the Stream Extension directory entry, the valid value for this field is 0.

7.6.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.3).

7.6.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.4).

7.6.2 GeneralSecondaryFlags Field

The GeneralSecondaryFlags field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2) and defines the contents of the CustomDefined field to be reserved.

7.6.2.1 AllocationPossible Field

The AllocationPossible field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.1).

For the Stream Extension directory entry, the valid value for this field is 1.

7.6.2.2 NoFatChain Field

The NoFatChain field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.2).

7.6.3 NameLength Field

The NameLength field shall contain the length of the Unicode string the subsequent File Name directory entries (see Section 7.7) collectively contain.

The valid range of values for this field shall be:

- At least 1, which is the shortest possible file name
- At most 255, which is the longest possible file name

The value of the NameLength field also affects the number File Name Directory Entries (see Section 7.7).

7.6.4 NameHash Field

The NameHash field shall contain a 2-byte hash (see Figure 4) of the up-cased file name. This enables implementations to perform a quick comparison when searching for a file by name. Importantly, the NameHash provides a sure verification of a mismatch. Implementations shall verify all NameHash matches with a comparison of the up-cased file name.

Figure 4 NameHash Computation

```
UInt16  NameHash
(
    WCHAR * FileName,       // points to an in-memory copy of the up-cased file
name
       UCHAR    NameLength
)
{
       UCHAR *  Buffer =        (UCHAR *)FileName;
       UInt16   NumberOfBytes = (UInt16)NameLength * 2;
       UInt16   Hash =          0;
       UInt16   Index;

       for (Index = 0; Index < NumberOfBytes; Index++)
       {
              Hash = ((Hash&1) ? 0x8000 : 0) + (Hash>>1) +
(UInt16)Buffer[Index];
    }

    return Hash;
}
```

7.6.5 ValidDataLength Field

The ValidDataLength field shall describe how far into the data stream user data has been written. Implementations shall update this field as they write data further out into the data stream. On the storage media, the data between the valid data length and the data length of the data stream is undefined. Implementations shall return zeroes for read operations beyond the valid data length.

If the corresponding File directory entry describes a directory, then the only valid value for this field is equal to the value of the DataLength field. Otherwise, the range of valid values for this field shall be:

- At least 0, which means no user data has been written out to the data stream
- At most DataLength, which means user data has been written out to the entire length of the data stream

7.6.6 FirstCluster Field

The FirstCluster field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.3).

This field shall contain the index of the first cluster of the data stream, which hosts the user data.

7.6.7 DataLength Field

The DataLength field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.4).

If the corresponding File directory entry describes a directory, then the valid value for this field is the entire size of the associated allocation, in bytes, which may be 0. Further, for directories, the maximum value for this field is 256MB.

7.7 File Name Directory Entry

File Name directory entries are critical secondary directory entries in File directory entry sets (see Table 34). The valid number of File Name directory entries in a File directory entry set is NameLength / 15, rounded up to the nearest integer. Further, File Name directory entries are valid only if they immediately follow the Stream Extension directory entry as a consecutive series. File Name directory entries combine to form the file name for the File directory entry set.

All children of a given directory entry shall have unique File Name Directory Entry Sets. That is to say there can be no duplicate file or directory names after up-casing within any one directory.

Table 34 File Name DirectoryEntry

	Offset	Size	
Field Name	**(byte)**	**(byte)**	**Comments**

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 7.7.1 defines its contents.
GeneralSecondaryFlags	1	1	This field is mandatory and Section 7.7.2 defines its contents.
FileName	2	30	This field is mandatory and Section 7.7.3 defines its contents.

7.7.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1).

7.7.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.1).

For the Stream Extension directory entry, the valid value for this field is 1.

7.7.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.2).

For the Stream Extension directory entry, the valid value for this field is 0.

7.7.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.3).

7.7.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.4).

7.7.2 GeneralSecondaryFlags Field

The GeneralSecondaryFlags field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2) and defines the contents of the CustomDefined field to be reserved.

7.7.2.1 AllocationPossible Field

The AllocationPossible field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.1).

For the Stream Extension directory entry, the valid value for this field is 0.

7.7.2.2 NoFatChain Field

The NoFatChain field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.2).

7.7.3 FileName Field

The FileName field shall contain a Unicode string, which is a portion of the file name. In the order File Name directory entries exist in a File directory entry set, FileName fields concatenate to form the file name for the File directory entry set. Given the length of the FileName field, 15 characters, and the maximum number of File Name directory entries, 17, the maximum length of the final, concatenated file name is 255.

The concatenated file name has the same set of illegal characters as other FAT-based file systems (see Table 35). Implementations should set the unused characters of FileName fields to the value 0000h.

Table 35 Invalid FileName Characters

Character Code	Description	Character Code	Description	Character Code	Description
0000h	Control code	0001h	Control code	0002h	Control code
0003h	Control code	0004h	Control code	0005h	Control code

Character Code	Description	Character Code	Description	Character Code	Description
0006h	Control code	0007h	Control code	0008h	Control code
0009h	Control code	000Ah	Control code	000Bh	Control code
000Ch	Control code	000Dh	Control code	000Eh	Control code
000Fh	Control code	0010h	Control code	0011h	Control code
0012h	Control code	0013h	Control code	0014h	Control code
0015h	Control code	0016h	Control code	0017h	Control code
0018h	Control code	0019h	Control code	001Ah	Control code
001Bh	Control code	001Ch	Control code	001Dh	Control code
001Eh	Control code	001Fh	Control code	0022h	Quotation mark
002Ah	Asterisk	002Fh	Forward slash	003Ah	Colon
003Ch	Less-than sign	003Eh	Greater-than sign	003Fh	Question mark
005Ch	Back slash	007Ch	Vertical bar		

The file names "." and ".." have the special meaning of "this directory" and "containing directory", respectively. Implementations shall not record either of these reserved file names in the FileName field. However, implementations may generate these two file names in directory listings to refer to the directory being listed and the containing directory.

Implementations may wish to restrict file and directory names to just the ASCII character set. If so they should limit their character use to the range of valid characters in the first 128 Unicode entries. They must still store file and directory names in Unicode on the volume and translate to/from ASCII/Unicode when interfacing with the user.

7.8 Vendor Extension Directory Entry

The Vendor Extension directory entry is a benign secondary directory entry in File directory entry sets (see Table 36). A File directory entry set may contain any number of Vendor Extension directory entries, up to the limit of secondary directory entries, less the number of other secondary directory entries. Further, Vendor Extension directory entries are valid only if they do not precede the required Stream Extension and File Name directory entries.

Vendor Extension directory entries enable vendors to have unique, vendor-specific directory entries in individual File directory entry sets via the VendorGuid field (see Table 36). Unique directory entries effectively enable vendors to extend the exFAT file system. Vendors may define the contents of the VendorDefined field (see Table 36). Vendor implementations may maintain the contents of the VendorDefined field and may provide vendor-specific functionality.

Implementations which do not recognize the GUID of a Vendor Extension directory entry shall treat the directory entry the same as any other unrecognized benign secondary directory entry (see Section 8.2).

Table 36 Vendor Extension DirectoryEntry

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 7.8.1 defines its contents.
GeneralSecondaryFlags	1	1	This field is mandatory and Section 7.8.2 defines its contents.
VendorGuid	2	16	This field is mandatory and Section 7.8.3 defines its contents.
VendorDefined	18	14	This field is mandatory and vendors may define its contents.

7.8.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1).

7.8.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.1).

For the Vendor Extension directory entry, the valid value for this field is 0.

7.8.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.2).

For the Vendor Extension directory entry, the valid value for this field is 1.

7.8.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.3).

7.8.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.4).

7.8.2 GeneralSecondaryFlags Field

The GeneralSecondaryFlags field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2) and defines the contents of the CustomDefined field to be reserved.

7.8.2.1 AllocationPossible Field

The AllocationPossible field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.1).

For the Vendor Extension directory entry, the valid value for this field is 0.

7.8.2.2 NoFatChain Field

The NoFatChain field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.2).

7.8.3 VendorGuid Field

The VendorGuid field shall contain a GUID which uniquely identifies the given Vendor Extension.

All possible values for this field are valid, except the null GUID, which is {00000000-0000-0000-0000-000000000000}. However, vendors should use a GUID-generating tool, such as GuidGen.exe, to select a GUID when defining their extensions.

The value of this field determines the vendor-specific structure of the VendorDefined field.

7.9 Vendor Allocation Directory Entry

The Vendor Allocation directory entry is a benign secondary directory entry in File directory entry sets (see Table 37). A File directory entry set may contain any number of Vendor Allocation directory entries, up to the limit of secondary directory entries, less the number of other secondary directory entries. Further, Vendor Allocation directory entries are valid only if they do not precede the required Stream Extension and File Name directory entries.

Vendor Allocation directory entries enable vendors to have unique, vendor-specific directory entries in individual File directory entry sets via the VendorGuid field (see Table 37). Unique directory entries effectively enable vendors to extend the exFAT file system. Vendors may define the contents of the associated clusters, if any exist. Vendor implementations may maintain the contents of the associated clusters, if any, and may provide vendor-specific functionality.

Implementations which do not recognize the GUID of a Vendor Allocation directory entry shall treat the directory entry the same as any other unrecognized benign secondary directory entry (see Section 8.2).

Table 37 Vendor Allocation DirectoryEntry

Field Name	Offset (byte)	Size (byte)	Comments

Field Name	Offset (byte)	Size (byte)	Comments
EntryType	0	1	This field is mandatory and Section 7.9.1 defines its contents.
GeneralSecondaryFlags	1	1	This field is mandatory and Section 7.9.2 defines its contents.
VendorGuid	2	16	This field is mandatory and Section 7.9.3 defines its contents.
VendorDefined	18	2	This field is mandatory and vendors may define its contents.
FirstCluster	20	4	This field is mandatory and Section 7.9.4 defines its contents.
DataLength	24	8	This field is mandatory and Section 7.9.5 defines its contents.

7.9.1 EntryType Field

The EntryType field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1).

7.9.1.1 TypeCode Field

The TypeCode field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.1).

For the Vendor Allocation directory entry, the valid value for this field is 1.

7.9.1.2 TypeImportance Field

The TypeImportance field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.2).

For the Vendor Allocation directory entry, the valid value for this field is 1.

7.9.1.3 TypeCategory Field

The TypeCategory field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.3).

7.9.1.4 InUse Field

The InUse field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.1.4).

7.9.2 GeneralSecondaryFlags Field

The GeneralSecondaryFlags field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2) and defines the contents of the CustomDefined field to be reserved.

7.9.2.1 AllocationPossible Field

The AllocationPossible field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.1).

For the Vendor Allocation directory entry, the valid value for this field is 1.

7.9.2.2 NoFatChain Field

The NoFatChain field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.2.2).

7.9.3 VendorGuid Field

The VendorGuid field shall contain a GUID which uniquely identifies the given Vendor Allocation.

All possible values for this field are valid, except the null GUID, which is {00000000-0000-0000-0000-000000000000}. However, vendors should use a GUID-generating tool, such as GuidGen.exe, to select a GUID when defining their extensions.

The value of this field determines the vendor-specific structure of the contents of the associated clusters, if any exist.

7.9.4 FirstCluster Field

The FirstCluster field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.3).

7.9.5 DataLength Field

The DataLength field shall conform to the definition provided in the Generic Secondary DirectoryEntry template (see Section 6.4.4).

7.10 TexFAT Padding Directory Entry

This specification, exFAT Revision 1.00 File System Basic Specification, does not define the TexFAT Padding directory entry. However, its type code is 1 and its type importance is 1. Implementations of this specification shall treat TexFAT Padding directory entries the same as any other unrecognized benign primary directory entries, implementations shall not move TexFAT Padding directory entries.

8 Implementation Notes

8.1 Recommended Write Ordering

Implementations should ensure the volume is as resilient as possible to power faults and other unavoidable failures. When creating new directory entries or modifying cluster allocations, implementations should generally follow this writing order:

1. Set the value of the VolumeDirty field to 1
2. Update the active FAT, if necessary
3. Update the active Allocation Bitmap
4. Create or update the directory entry, if necessary
5. Clear the value of the VolumeDirty field to 0, if its value prior to the first step was 0

When deleting directory entries or freeing cluster allocations, implementations should follow this writing order:

1. Set the value of the VolumeDirty field to 1
2. Delete or update the directory entry, if necessary
3. Update the active FAT, if necessary

4. Update the active Allocation Bitmap

5. Clear the value of the VolumeDirty field to 0, if its value prior to the first step was 0

8.2 Implications of Unrecognized Directory Entries

Future exFAT specifications of the same major revision number, 1, and minor revision number higher than 0, may define new benign primary, critical secondary, and benign secondary directory entries. Only exFAT specifications of a higher major revision number may define new critical primary directory entries. Implementations of this specification, exFAT Revision 1.00 File System Basic Specification, should be able to mount and access any exFAT volume of major revision number 1 and any minor revision number. This presents scenarios in which an implementation may encounter directory entries which it does not recognize. The following describe implications of these scenarios:

1. The presence of unrecognized critical primary directory entries in the root directory renders the volume invalid. The presence of any critical primary directory entry, except File directory entries, in any non-root directory, renders the hosting directory invalid.

2. Implementations shall not modify unrecognized benign primary directory entries or their associated cluster allocations. However, when deleting a directory, and only when deleting a directory, implementations shall delete unrecognized benign primary directory entries and free all associated cluster allocations, if any.

3. Implementations shall not modify unrecognized critical secondary directory entries or their associated cluster allocations. The presence of one or more unrecognized critical secondary directory entries in a directory entry set renders the entire directory entry set unrecognized. When deleting a directory entry set which contains one or more unrecognized critical secondary directory entries, implementations shall free all cluster allocations, if any, associated with unrecognized critical secondary directory entries. Further, if the directory entry set describes a directory, implementations may:

    - Traverse into the directory

    - Enumerate the directory entries it contains

    - Delete contained directory entries

    - Move contained directory entries to a different directory

    However, implementations shall not:

- Modify contained directory entries, except delete, as noted
- Create new contained directory entries
- Open contained directory entries, except traverse and enumerate, as noted

4. Implementations shall not modify unrecognized benign secondary directory entries or their associated cluster allocations. Implementations should ignore unrecognized benign secondary directory entries. When deleting a directory entry set, implementations shall free all cluster allocations, if any, associated with unrecognized benign secondary directory entries.

9 File System Limits

9.1 Sector Size Limits

The BytesPerSectorShift field defines the lower and upper sector size limits (which evaluates to **lower limit: 512 bytes; upper limit: 4,096 bytes**).

9.2 Cluster Size Limits

The SectorsPerClusterShift field defines the lower and upper cluster size limits (**lower limit: 1 sector; upper limit: 25 -- BytesPerSectorShift sectors**, which evaluates to 32MB).

9.3 Cluster Heap Size Limits

The Cluster Heap shall contain at least enough space to host the following basic file system structures: the root directory, all Allocation Bitmaps, and the Up-case Table.

The lower Cluster Heap size limit is a function of the lower size limit of each of the basic file system structures which reside in the Cluster Heap. Even given the smallest possible cluster (512 bytes), each of the basic file system structures needs no more than one cluster. Therefore, the **lower limit is: 2 + NumberOfFats clusters**, which evaluates to either 3 or 4 clusters, depending on the value the NumberOfFats field.

The upper Cluster Heap size limit is a simple function of the maximum possible number of clusters, which the ClusterCount field defines (**upper limit: 2^{32}- 11 clusters**). Regardless of the cluster size, such a cluster heap has enough space to at least host the basic file system structures.

9.4 Volume Size Limits

The VolumeLength field defines the lower and upper volume size limits (lower limit: **2^{20}/ $2^{BytesPerSectorShift}$sectors**, which evaluates to 1MB; **upper limit: 2^{64}- 1 sectors**, which, given the largest possible sector size, evaluates to approximately 64ZB). However, this specification recommends no more than 2^{24}- 2 clusters in the Cluster Heap (see Section 3.1.9). Therefore, the recommended upper limit of a volume is: ClusterHeapOffset + (2^{24}- 2) * $2^{SectorsPerClusterShift}$. Given the largest possible cluster size, 32MB, and assuming ClusterHeapOffset is 96MB (enough space for the Main and Backup Boot regions and only the First FAT), the recommended upper limit of a volume evaluates to approximately 512TB.

9.5 Directory Size Limits

The DataLength field of the Stream Extension directory entry defines the lower and upper directory size limits (**lower limit: 0 bytes; upper limit: 256MB**). This means a directory may host up to 8,388,608 directory entries (each directory entry consumes 32 bytes). Given the smallest possible File directory entry set, three directory entries, a directory may host up to 2,796,202 files.

10 Appendix

10.1 Globally Unique Identifiers (GUIDs)

A GUID is the Microsoft implementation of a universally unique identifier. A GUID is a 128-bit value consisting of one group of 8 hexadecimal digits, followed by three groups of 4 hexadecimal digits each, and followed by one group of 12 hexadecimal digits, for example {6B29FC40-CA47-1067-B31D-00DD010662DA}, (see Table 38).

Table 38 GUID Structure

Field Name	Offset (byte)	Size (bytes)	Comments
Data1	0	4	This field is mandatory and contains the four bytes from the first group of the GUID (6B29FC40h from the example).
Data2	4	2	This field is mandatory and contains the two bytes from the second group of the GUID (CA47h from the example).

Field Name	Offset (byte)	Size (bytes)	Comments
Data3	6	2	This field is mandatory and contains the two bytes from the third group of the GUID (1067h from the example).
Data4[0]	8	1	This field is mandatory and contains the most significant byte from fourth group of the GUID (B3h from the example).
Data4[1]	9	1	This field is mandatory and contains the least significant byte from the fourth group of the GUID (1Dh from the example).
Data4[2]	10	1	This field is mandatory and contains the first byte from the fifth group of the GUID (00h from the example).
Data4[3]	11	1	This field is mandatory and contains the second byte from the fifth group of the GUID (DDh from the example).
Data4[4]	12	1	This field is mandatory and contains the third byte from the fifth group of the GUID (01h from the example).
Data4[5]	13	1	This field is mandatory and contains the fourth byte from the fifth group of the GUID (06h from the example).
Data4[6]	14	1	This field is mandatory and contains the fifth byte from the fifth group of the GUID (62h from the example).
Data4[7]	15	1	This field is mandatory and contains the sixth byte from the fifth group of the GUID (DAh from the example).

10.2 Partition Tables

To ensure interoperability of exFAT volumes in a broad set of usage scenarios, implementations should use partition type 07h for MBR partitioned storage and partition GUID {EBD0A0A2-B9E5-4433-87C0-68B6B72699C7} for GPT partitioned storage.

11 Documentation Change History

Table 39 describes the history of releases of, corrections to, additions to, removals from, and clarifications of this document.

Table 39 Documentation Change History

Date	Description of Change
08-Jan-2008	First release of the Basic Specification, which includes: Section 1, Introduction Section 2, Volume Structure Section 3, Main and Backup Boot Regions Section 4, File Allocation Table Region Section 5, Data Region Section 6, Directory Structure Section 7, Directory Entry Definitions Section 8, Implementation Notes Section 9, File System Limits Section 10, Appendix

Date	Description of Change
08-Jun-2008	Second release of the Basic Specification, which includes the following changes: Addition of Section 11, Documentation Change History Addition of the Vendor Extension and Vendor Allocation directory entries in Sections 7.8 and 7.9 Addition of the recommended up-case table in Sections 7.2.5 and 7.2.5.1 Addition of the UtcOffset fields in Section 7.4 and addition of the UTC acronym in Section 1.3 Correction of the size of the CustomDefined field in Table 19 Correction of the valid range of NameLength values in Section 7.6.3 Correction and clarification of the Timestamp and 10msIncrement fields in Section 7.4 Clarification of the Null Parameters structure in Section 3.3 Clarification of the meaning of the values of the NoFatChain field in Section 6.3.4.2 Clarification of the meaning of the values of the DataLength field in Section 6.2.3 Clarification of the VolumeDirty field in Section 3.1.13.2 and recommended write ordering in Section 8.1 Clarification of the MediaFailure field in Section 3.1.13.3

Date	Description of Change
01-Oct-2008	Third release of the Basic Specification, which includes the following changes: Addition of SHALL, SHOULD and MAY to field explanations Addition of UTC definition in Table 2 Section 1.3 Modified sections 1.5, to ensure alignment with the TexFAT specification document. Clarified the restriction that only Microsoft may define the layout of Directory Entries in Section 6.2 Added clarification that FirstCluster Field shall be zero if the DataLength is zero and NoFatChain is set to Section 6.3.5 and Section 6.4.3 Clarified requirements for valid file directory entries in Section 7.4 Added requirement for unique file and directory names to Section 7.7 Added implementation note for ASCII to the end of Section 7.7.3
01-Jan-2009	Fourth release of the Basic Specification, which includes the following changes: Removed references to Windows CE Access Control entries Added clarification to Section 7.2.5.1 to explicitly require a full up-case table
02-Sep-2009	Fifth release of the Basic Specification, which includes the following changes: Document formatting changes to allow better PDF conversion

Date	Description of Change
24-Feb-2010	Sixth release of the Basic Specification, which includes the following changes: Amended incorrect statement: “FirstCluster Field shall be zero if the DataLength is zero and NoFatChain is set” in Section 6.3.5 and Section 6.4.3 to “If the NoFatChain bit is 1 then FirstCluster must point to a valid cluster in the cluster heap“ to clarify that there must be valid allocation if the NoFatChain bit is set. Added “If the NoFatChain bit is 1 then DataLength must not be zero. If the FirstCluster field is zero, then DataLength must also be zero” to Section 6.3.6 and Section 6.4.4 to clarify that there must be valid allocation if the NoFatChain bit is set. Updated copyright notice to 2010
26-Aug-2019	Seventh release of the Basic Specification, which includes the following changes: Updated legal terms pertaining to the specification, including: Removal of Microsoft Confidential notice Removal of Microsoft Corporation Technical Documentation License Agreement section Updated copyright notice to 2019

Is this page helpful?

👍 Yes 👎 No