Xbox 360 File Specifications Reference

# Introduction

This reference attempts to document the specifications of the custom data formats in use by the Xbox 360 console. This data has either been discovered through reverse engineering or from secondary sources in the Xbox 360 enthusiast community. Often, only the fields that are necessary for data access have been properly deciphered and in all cases the names of data fields are at best educated guesses. As far as the author is aware none of this information has been taken from or derived from primary sources and should not be considered definitive. For definitive specifications please consult Microsoft.

Contents

[Introduction 1](#_Toc283662853)

[XTAF 2](#_Toc283662854)

[Secure Transacted File System 4](#_Toc283662855)

[Xbox Database Format 8](#_Toc283662856)

[Account Block 10](#_Toc283662857)

[Appendix – Additional Tables 11](#_Toc283662858)

[References 13](#_Toc283662859)

XTAF

XTAF is the Xbox 360 file system, sometimes referred to as FATX. XTAF is the file system that is placed upon the raw hard drive of the Xbox 360. It is a derivative of the legacy File Allocation Table file system that was introduced in Microsoft DOS. XTAF uses a big-endian byte order as opposed to the little-endian FAT and many legacy configuration options of FAT have been removed. The community knowledge of XTAF is well advanced with the documentation available from the Free60 project being excellent as well as there being several open source implementations of XTAF of varying completeness to provide example code to help understanding.

The XTAF format contains three distinct sections, a 512 byte header, a file allocation table and a number of 0x4000 byte long clusters containing file data and directory information.

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Content | Description |
| 0x000 | 4 | XTAF | Magic identifier |
| 0x004 | 4 |  | Serial number of the file system, could also be 0 or -1 |
| 0x008 | 4 |  | Number of sectors per cluster |
| 0x00C | 4 | 1 | Number of copies of the file allocation table |
| 0x010 | 2 | 0 | Unknown |
| 0x012 | 0xFEE | 0xFF | Unused / unknown |

**XTAF Header Structure**

Files in XTAF are split into 0x4000 (16384) byte clusters that are not necessarily adjacent on disk. The File Allocation Table is used to determine where the next cluster of a file resides. The file allocation table is an array of cluster numbers that are either 4 bytes or 2 bytes wide depending on the size of the disk (2 bytes if the number of clusters is less than 0xFFF4 and 4 bytes otherwise). Each entry in the file allocation table provides the offset to the cluster that follows the current one with the special values 0xFFFFFFF and 0x00000000 to denote that this is the last cluster in a chain and that the cluster is unallocated respectively. The File Allocation Table is located at offset 0x1000 into the partition.

Immediately after the File Allocation Table is the root directory cluster. The root directory cluster contains a list of file records that represent the files and directories present in the root directory of the file system. Unlike other directories the root directory can only occupy one cluster and so accessing it does not require any File Allocation Table lookups. File records contain the name, size, starting cluster and access/modified/created time stamps of files and directories. If a file record is allocated with all bytes set to 0xFF this signifies the end of the directory.

|  |  |  |
| --- | --- | --- |
| Offset | Size | Description |
| 0x00 | 1 | length of the file name, or 0xE5 for deleted files |
| 0x01 | 1 | file flags |
| 0x02 | 0x2A | file name, padded with either 0x00 or 0xFF bytes |
| 0x2C | 4 | starting cluster of file, 0 for empty files |
| 0x30 | 4 | file size, 0 for directories |
| 0x34 | 2 | creation date |
| 0x36 | 2 | creation time |
| 0x38 | 2 | access date |
| 0x3A | 2 | access time |
| 0x3C | 2 | update date |
| 0x3E | 2 | update time |

**XTAF File Record Structure**

Unlike the FAT file system XTAF directories do not contain ‘.’ and ‘..’ records that point to the current directory and parent directory. The time stamps and file flags are in the same format as the FAT file system. The first cluster entry refers to both the index into the File Allocation Table of the cluster and the offset of the cluster on disk. Cluster numbers start at number 2 as the root directory cluster is considered cluster 1 before indexing into the File Allocation Table or calculating the offset of a cluster subtract 1 from the cluster number. To calculate the file system offset of a cluster multiply the cluster number by 0x4000 and add the offset of the root cluster.

# Secure Transacted File System

Secure Transacted File System (STFS) is a contain file format used to store downloaded content as well as locally generated data such as saved games and user profiles. The format has built in controls to ensure the integrity of the data including a digital signature and tables containing SHA1 hashes of each 0x1000 bytes of data. STFS files can be broken down into three types, LIVE, PIRS and CON files. LIVE files contain downloaded content such as games and movies signed with an RSA key controlled by Microsoft and PIRS files are similar except are not delivered by the Xbox LIVE service such as system updates while CON files are created by the Xbox 360 locally and are signed by the console. This arrangement makes LIVE and PIRS resistant to user tampering and CON files resistant from corruption. STFS files are generally found on XTAF file systems.

STFS containers are the most complicated of the Xbox 360 file formats examined and are also the least well documented format. Information online is limited to a description of its header format and speculative descriptions of its verification techniques as well as two incomplete and contradictory open source projects. This specification should be considered the most preliminary. The best public information about this file format is the source code of an Xbox 360 modification library called X360 by DJ Shepherd

The STFS file starts with a long header of 0x971A bytes that includes the type of STFS file it is (LIVE, PIRS or CONS), the digital signature of the file, the metadata structure version and large amounts of metadata including up to two images and title/description text in multiple languages. After the header there are the first hash tables and a table with the file listing. After this point the data starts divided into 0x1000 byte blocks interspersed every 170 blocks with one or more tables containing additional hashes.

**STFS Header** (Free60.org, 2011)

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x4 | ascii string | Magic “CON “, PIRS or LIVE |
| 0x6 | 0x5 | bytes | Certificate Owner Console ID |
| 0xB | 0x14 | ascii string | Certificate Owner Console Part Number |
| 0x1F | 0x1 | byte | Certificate Owner Console Type (1 for devkit, 2 for retail) |
| 0x20 | 0x8 | ascii string | Certificate Date of Generation |
| 0x28 | 0x4 | bytes | Public Exponent |
| 0x2C | 0x80 | bytes | Public Modulus |
| 0xAC | 0x100 | bytes | Certificate Signature |
| 0x1AC | 0x80 | bytes | Signature |

Alternatively for LIVE and PIRS packages the header changes as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x4 | 0x100 | bytes | Package Signature |
| 0x104 | 0x128 | bytes | Padding |

**STFS Metadata** (Free60.org, 2011)

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x22C | 0x100 | license entries (see below) | Licensing Data (indicates package owner) |
| 0x32C | 0x14 | bytes | Content ID / Header SHA1 Hash |
| 0x340 | 0x4 | unsigned int | Entry ID |
| 0x344 | 0x4 | signed int | Content Type (see below) |
| 0x348 | 0x4 | signed int | Metadata Version (see below) |
| 0x34C | 0x8 | signed long | Content Size |
| 0x354 | 0x4 | unsigned int | Media ID |
| 0x358 | 0x4 | signed int | Version (system/title updates) |
| 0x35C | 0x4 | signed int | Base Version (system/title updates) |
| 0x360 | 0x4 | unsigned int | Title ID |
| 0x364 | 0x1 | Byte | Platform (xbox/gfwl?) |
| 0x365 | 0x1 | Byte | Executable Type |
| 0x366 | 0x1 | Byte | Disc Number |
| 0x367 | 0x1 | Byte | Disc In Set |
| 0x368 | 0x4 | unsigned int | Save Game ID |
| 0x36C | 0x5 | bytes | Console ID |
| 0x371 | 0x8 | bytes | Profile ID |
| 0x379 | 0x1 | Byte | Volume Descriptor Struct Size (usually 0x24) |
| 0x37A | 0x24 | STFS Volume Descriptor | File System Volume Descriptor |
| 0x39D | 0x4 | signed int | Data File Count |
| 0x3A1 | 0x8 | signed long | Data File Combined Size |
| 0x3A9 | 0x8 | bytes | Reserved |
| 0x3B1 | 0x4C | bytes | Padding |
| 0x3FD | 0x14 | bytes | Device ID |
| 0x411 | 0x18 \* 12 (0x900) | unicode string | Display Name |
| 0xD11 | 0x18 \* 12 (0x900) | unicode string | Display Description |
| 0x1611 | 0x80 | unicode string | Publisher Name |
| 0x1691 | 0x80 | unicode string | Title Name |
| 0x1711 | 0x1 | Byte | Transfer Flags (see below) |
| 0x1712 | 0x4 | signed int | Thumbnail Image Size |
| 0x1716 | 0x4 | signed int | Title Thumbnail Image Size |
| 0x171A | 0x4000 | Image | Thumbnail Image |
| 0x571A | 0x4000 | Image | Title Thumbnail Image |

**STFS Metadata Version 2** (Free60.org, 2011)

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x3B1 | 0x10 | bytes | Series ID |
| 0x3C1 | 0x10 | bytes | Season ID |
| 0x3D1 | 0x2 | signed short | Season Number |
| 0x3D5 | 0x2 | signed short | Episode Number |
| 0x3D5 | 0x28 | bytes | Padding |
| 0x171A | 0x3D00 (thumbnail size) | image | Thumbnail Image |
| 0x541A | 0x300 (each 0x80 = different locale) | image | Additional Display Names |
| 0x571A | 0x3D00 (title thumbnail size) | image | Title Thumbnail Image |
| 0x941A | 0x300 (each 0x80 = different locale) | image | Additional Display Descriptions |

If the metadata version field is set to 2 the above changes are present in the metadata format. The STFS volume descriptor contains information about the location of the file listing table and the top level hash table.

**STFS Volume Descriptor** (Free60.org, 2011)

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x1 | byte | Reserved |
| 0x1 | 0x1 | byte | Block Seperation |
| 0x2 | 0x2 | signed short | File Table Block Count |
| 0x4 | 0x3 | signed int24 | File Table Block Number |
| 0x7 | 0x14 | bytes | Top Hash Table Hash |
| 0x1B | 0x4 | signed int | Total Allocated Block Count |
| 0x1F | 0x4 | signed int | Total Unallocated Block Count |

If bit 12, 13 and 15 of the Entry ID are on ((Entry ID + 0xFFF) & 0xF000) >> 0xC == 0xB) there are 2 hash tables every 0xAA (170) blocks, evidence suggests that this is the case in CON files. The second hash table contains almost identical information to the first and it is hypothesized that the duplication is to support transactional integrity. Hash tables contain 170 records each containing a SHA1 hash of the relevant block as well as a status byte and the block number of the following block.

**STFS Hash Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x14 | bytes | SHA1 Hash of Block |
| 0x14 | 0x01 | byte | Status (0x00, 0x40, 0x80, 0xC0) |
| 0x15 | 0x03 | 24 bit integer | Next Block Index |

**STFS Hash Status Values**

|  |  |
| --- | --- |
| Value | Information |
| 0x00 | Unused Block |
| 0x40 | Free Block (previously used) |
| 0x80 | Used Block |
| 0xC0 | Newly Allocated Block |

The file listing table contains a series of file listing structures that describe a file or directory inside the STFS archive. The File Listing structure contains both big-endian and little-endian values and the path indicator value is an offset into the File Listing table where -1 (0xFFFF) indicates the root directory. Time stamps in the File Listing table are the same as XTAF (and FAT). Bit 6 of byte 0x28 indicates whether or not the entry is a directory.

**STFS File Listing** (Free60.org, 2011)

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x28 | ascii string | File name, null-padded |
| 0x28 | 0x1 | byte | Length of file name, plus flags |
| 0x29 | 0x3 | signed int24 | Number of blocks allocated for file (little endian) |
| 0x2C | 0x3 | signed int24 | Copy of 0x29 |
| 0x2F | 0x3 | signed int24 | Starting block number of file (little endian) |
| 0x32 | 0x2 | signed short | Path indicator (big endian) |
| 0x34 | 0x4 | unsigned int | Size of file in bytes (big endian) |
| 0x38 | 0x4 | signed int | Update date/time stamp of file |
| 0x3C | 0x4 | signed int | Access date/time stamp of file |

Hash tables are interspersed with data so that it is not trivial to convert a block number into an offset in the file. Block numbers refer to data blocks only and do not increment for hash table blocks. The following code segment demonstrates how to adjust a block number to take into account embedded hash tables. The variable table\_size\_shift is 1 if ((Entry ID + 0xFFF) & 0xF000) >> 0xC == 0xB) is True and 0 otherwise. The return value can be multiplied by 0x1000 and then added to 0xC000 to find the location of the block on disk.

block\_adjust = 0

if block\_num >= 0xAA:

block\_adjust += ((block\_num // 0xAA)) + 1 << table\_size\_shift

if block\_num > 0x70E4:

block\_adjust += ((block\_num // 0x70E4) + 1)<< table\_size\_shift

return block\_adjust + block\_num

Every 0xAA blocks there is a hash table containing the hashes of the next 0xAA blocks. Every 0x70E4 (0xAA \* 0xAA) blocks there is a hash table presumably containing hashes of the previous 0xAA hash blocks. Finally, every 0x4AF768 (0xAA \* 0xAA \* 0xAA) there is a table presumably containing the hashes of the 0x70E4 hash tables.

# Xbox Database Format

The Xbox Database Format (XDBF) is a generic container format for storing records and files. XDBF is the format of Gamer Profile Data (GPD) files which are used to store information relevant to a single user including settings, information about the games played, achievement information for each game and image resources. XDBF is also used by Statistics, Presence, Achievement files (SPA) which are embedded in game software bundles and are used to generate GPD files for each user that plays the game. GPD and SPA files are both usually found inside STFS containers. XDBF files have features to simplify the process of syncing them with remote servers.

The XDBF format is well understood by the Xbox 360 enthusiast community and is documented by the Free60 project. The file format is composed of a header, a table of record entries, a table of free space, and the data area. These regions are adjacent and their sizes can be calculated from header information. The data area is not divided into blocks or clusters and entries are contiguous inside the XDBF area and are specified by a start offset and a length. The XDBF header specifies the length of the entry and free space tables as well as how many records in those tables have been used.

**XDBF Header** (Free60.org, 2010)

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x4 | ascii string | Magic (0x58444246) “XDBF” |
| 0x4 | 0x4 | unsigned int | Version (0x10000) |
| 0x8 | 0x4 | unsigned int | Entry Table Length (in number of entries) |
| 0xC | 0x4 | unsigned int | Entry Count |
| 0x10 | 0x4 | unsigned int | Free Space Table Length (in number of entries) |
| 0x14 | 0x4 | unsigned int | Free Space Table Entry Count |

**XDBF Entry Table** (Free60.org, 2010)

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x2 | unsigned short | Namespace |
| 0x2 | 0x8 | unsigned long | ID |
| 0xA | 0x4 | unsigned int | Offset |
| 0xE | 0x4 | unsigned int | Length |

Offset is not the offset from the start of the file but rather the offset from the end of the free space table. Namespaces describe the type of entry in the XDBF file and vary depending on the particular type of XDBF file. GPD files contain achievement records, image records, setting records, title records, strings and achievement security records. String entries are UTF-16 big-endian strings and image entries are PNG files both of the length specified in the entry record. Other strings in achievement and title entries are also UTF-16 big-endian. The Setting entry structure is less well documented. In a Setting entry the Setting ID field determines the size and structure of the payload and the Content ID field determines which setting this entry corresponds to.

**GPD Namespaces**

|  |  |
| --- | --- |
| Value | Description |
| 1 | Achievement |
| 2 | Image |
| 3 | Setting |
| 4 | Title |
| 5 | String |
| 6 | Achievement Security |

**Achievement Entries**

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x4 | unsigned int | Magic (0x1C) |
| 0x4 | 0x4 | unsigned int | Achievement ID |
| 0x8 | 0x4 | unsigned int | Image ID |
| 0xC | 0x4 | signed int | Gamerscore |
| 0x10 | 0x4 | unsigned int | Flags (see below) |
| 0x14 | 0x8 | signed long | Unlock Time |
| 0x18 | null terminated | unicode string | Name |
| 0x18 + Name length | null terminated | unicode string | Locked Description |
| 0x18 + Name length + Locked Description length | null terminated | unicode string | Unlocked Description |

**Title Entries**

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x4 | unsigned int | Title ID |
| 0x4 | 0x4 | signed int | Achievement Count |
| 0x8 | 0x4 | signed int | Achievement Unlocked Count |
| 0xC | 0x4 | signed int | Gamerscore Total |
| 0x10 | 0x4 | signed int | Gamerscore Unlocked |
| 0x14 | 0x8 | signed long | Unknown |
| 0x1C | 0x4 | signed int | Unknown |
| 0x20 | 0x8 | signed long | Last Played Time |
| 0x28 | null terminated | unicode string | Title Name |

**Setting Entries**

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x8 | bytes | Content ID |
| 0x8 | 0x4 | signed int | Setting ID |
| 0xC | Variable | Bytes | Data |

**Setting ID**

|  |  |  |
| --- | --- | --- |
| Value | Description | Data |
| 0 | Context | Int |
| 1 | Unsigned Integer | Unsigned Integer |
| 2 | Long | 64 bit Integer |
| 3 | Double | Double |
| 4 | String | 32 bit Integer length followed by UTF-16 BE text |
| 5 | Float | Float |
| 6 | Binary | 32 bit Integer length followed by binary data |
| 7 | Timestamp | 64 bit Microsoft File Time timestamp |

# Account Block

The Account Block is a file inside a STFS archive that describes a Xbox 360 Profile. The Account Block is 404 bytes long and is encrypted with RC4 and HMAC-SHA1. The RC4 key is the first 16 bytes of the HMAC-SHA1 digest of the first 16 bytes of the Account file encrypted with the key E1BC159C73B1EAE9AB3170F3AD47EBF3 .

Very little information was available about the structure of this file and most of the following information about the decrypted Account Block has been derived from reverse engineering. It is worth emphasising that the layout and purpose of many fields of the Account Block is still unknown.

**Account Block**

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Length | Type | Information |
| 0x0 | 0x1 | byte | Account Type (0x20 = Live) |
| 0x1 | 0x4 | bytes | Account Passcode |
| 0x10 | 0x1E | UTF-16-BE | GamerTag |
| 0x30 | 0x8 | bytes | XUID (Live Only) |
| 0x39 | 0x1 | byte | Account Level (0x30 = Silver, 0x60 = Gold) |
| 0x3C | 0x4 | ASCII | Console Type (PROD, PART) |

# Appendix – Additional Tables

**STFS Content Types** (Free60.org, 2011)

|  |  |
| --- | --- |
| Value | Description |
| 0xD0000 | Arcade Title |
| 0x9000 | Avatar Item |
| 0x40000 | Cache File |
| 0x2000000 | Community Game |
| 0x80000 | Game Demo |
| 0x20000 | Gamer Picture |
| 0xA0000 | Game Title |
| 0xC0000 | Game Trailer |
| 0x400000 | Game Video |
| 0x4000 | Installed Game |
| 0xB0000 | Installer |
| 0x2000 | IPTV Pause Buffer |
| 0xF0000 | License Store |
| 0x2 | Marketplace Content |
| 0x100000 | Movie |
| 0x300000 | Music Video |
| 0x500000 | Podcast Video |
| 0x10000 | Profile |
| 0x3 | Publisher |
| 0x1 | Saved Game |
| 0x50000 | Storage Download |
| 0x30000 | Theme |
| 0x200000 | TV |
| 0x90000 | Video |
| 0x600000 | Viral Video |
| 0x70000 | Xbox Download |
| 0x5000 | Xbox Original Game |
| 0x60000 | Xbox Saved Game |
| 0x1000 | Xbox 360 Title |
| 0x5000 | Xbox Title |
| 0xE0000 | XNA |

**GPD Content ID** (Shepherd DJ, 2010)

|  |  |
| --- | --- |
| GPDID | Description |
| 0x10040004 | GamerZone |
| 0x10040005 | Region |
| 0x10040006 | Gamerscore |
| 0x10040007 | Presence State (Unknown) |
| 0x10040008 | Camera |
| 0x5004000B | Reputation |
| 0x1004000C | Mute Setting |
| 0x1004000D | Voice Output Speakers |
| 0x1004000E | Voice Volume Setting |
| 0x4064000F | Gamer Picture Reference |
| 0x40640010 | Personal Picture Reference |
| 0x402C0011 | Motto |
| 0x10040012 | Titles Played |
| 0x10040013 | Achievements Unlocked |
| 0x10040015 | Difficulty Setting |
| 0x10040018 | Control Sensitivity |
| 0x1004001D | Preferred Color 1 |
| 0x1004001E | Preferred Color 2 |
| 0x10040022 | Auto Aim |
| 0x10040024 | Auto Center |
| 0x10040024 | Action Movement Control |
| 0x10040038 | Gamerscore Earned On Title |
| 0x10040039 | Achievements Unlocked on Title |
| 0x1004003A | User Tier (Unknown) |
| 0x1004003B | Has Messanger Account |
| 0x1004003C | Messanger Auto Signin |
| 0x1004003D | Save Live Password |
| 0x1004003E | Public Friends List |
| 0x1004003F | Service Type (Unknown) |
| 0x41040040 | Account Name |
| 0x40520041 | Account Location |
| 0x41900042 | Gamercard URL |
| 0x43E80043 | Account Bio |
| 0x10000000 | Sync ID Table |
| 0x20000000 | Sync Record |
| 0x10042004 | Xbox.com Favorite Game (1) |
| 0x10042005 | Xbox.com Favorite Game (2) |
| 0x10042006 | Xbox.com Favorite Game (3) |
| 0x10042007 | Xbox.com Favorite Game (4) |
| 0x10042008 | Xbox.com Favorite Game (5) |
| 0x10042009 | Xbox.com Favorite Game (6) |
| 0x1004200A | Xbox.com Platforms Owned |
| 0x1004200B | Xbox.com Connection Speed |
| 0x700803F4 | User Crux Last Change Time (Unknown) |

# References

Free60.org. (2011, 01). *GPD*. Retrieved 01 24, 2011, from Free60 Wiki: http://free60.org/GPD

Free60.org. (2011, 01). *STFS*. Retrieved 01 24, 2011, from Free60 Wiki: http://free60.org/STFS

Free60.org. (2010, 08). *XDBF*. Retrieved 01 24, 2011, from Free60 Wiki: http://free60.org/XDBF

Free60.org. (2010, 03). *XTAF*. Retrieved 01 24, 2011, from Free60 Wiki: http://free60.org/XTAF

Shepherd DJ. (2010, 03). *Programs*. Retrieved 01 24, 2011, from SkunkieButt's Blog: http://skunkiebutt.com/?page\_id=362

TheFallen93. (2010, 09). *XAM Systemlink Patch*. Retrieved 01 24, 2011, from TheFallen93's Blog: http://webcache.googleusercontent.com/search?q=cache:http://thefallen93.com/Public/Xbox/xam\_systemlink.txt