.streamdb File Extension

> .streamdb stands for stream database. The .streamdb files contain the majority of game data in DOOM Eternal.

> As a general rule, any struct with `_t` appended to the end is an **actual struct used by the game engine**. Any other struct names have been created by the wiki author for organization/convenience purposes.

Signature

DOOM Eternal ".streamdb" files can be identified by the first 8 bytes of the file header, which is always: `0x50A5C2292EF3C761`.

Internally, the game engine references a 2nd type of stream database, which would be identified by a slightly different file signature: `0x4FA5C2292EF3C761` - however, this signature has not been observed in any files used in DOOM Eternal.

File Structure

The .streamdb file consists of 3 parts. An `INDEX` section, which is a list of all the files contained within, followed by a `PREFETCH` section, and finally the `DATA` section, which contains data referenced by the index.

```
struct STREAM_DB_FILE
{
    INDEX index;
    PREFETCH prefetch;
    DATA data;
};
```

The `INDEX` contains a list of hashed IDs rather than plaintext names. All files contained within the .streamdb are stored in a **headerless** format, and are usually compressed via Oodle Kraken or Oodle Leviathan compression technology.

> Files embedded in the .streamdb are impossible to identify by looking at the .streamdb alone. Instead, they are referenced via data contained in **.resources** files.

Index Section

The .streamdb `INDEX` structure is of varying length. It consists of a 32-byte header, followed by a variable number of 16-byte entries. The overall structure is described as follows:

```
struct INDEX
{
streamDatabaseHeader_t header;// File signature + metadata
streamDatabaseEntry2_t streamdbEntries[];// One 16-byte entry for each header.numEntries
};
```

The `streamDatabaseHeader_t` struct is a 32-byte sequence:

```
struct streamDatabaseHeader_t
{
uint64 magic;// 50 A5 C2 29 2E F3 C7 61
uint32 headerLength;
uint32 pad0;// null padding
uint32 pad1;// null padding
uint32 pad2;// null padding
uint32 numEntries;// Total entries, not including prefetch IDs
uint32 flags;// Always 3
};
```

The `streamDatabaseEntry2 t` struct is a 16-byte sequence. It will be repeated `n` times, where `n = streamDatabaseHeader t.numEntries`. Therefore, the total length of the `INDEX` section can be calculated as `length = 32 + (16 * streamDatabaseHeader_t.numEntries)`

```
struct streamDatabaseEntry2_t
{
uint64 identity;// Shuffled version of .resources ID
uint32 offset16; // Multiply by 16 for data offset within .streamdb
uint32 length;// Size of the file in .streamdb (usually compressed)
};
```

After the last entry, the `INDEX` section ends and the `PREFETCH` section begins.

Prefetch Section

The .streamdb `PREFETCH` structure is of varying length. It consists of a 16-byte header, followed (optionally) by either one or two 16-byte prefetchBlocks, and a number of 8-byte prefetchIDs.

The overall `PREFETCH` structure is described as follows:

```
struct PREFETCH
{
    streamDatabasePrefetchHeader_t prefetchHeader;□// Prefetch section header
    streamDatabasePrefetchBlock_t prefetchBlock[];□// (Optional) Between 0-2 prefetch "blocks"
□uint64 prefetchID[];□□□□□□// (Optional) 0 or more prefetch file IDs.
};
```

The `streamDatabasePrefetchHeader_t` struct is a 16-byte sequence. It is always present in the .streamdb file, even if this .streamdb does not contain any prefetch entries.

```
struct streamDatabasePrefetchHeader_t
{
□uint32 numPrefetchBlocks;
□uint32 totalLength;□□□□□// Total length of prefetch header, blocks, entries
};
```

If `streamDatabasePrefetchHeader_t.numPrefetchBlocks = 0`, then the PREFETCH section ends here. Otherwise, it is followed by the number of `streamDatabasePrefetchBlock_t` structs specified (which is always an integer between `0` and `2`).

```
struct streamDatabasePrefetchBlock_t
{
□uint64 name;□□□□□□// Hash of "AI" or "FirstPerson"
□uint32 firstItemIndex;□□□□// Offset relative to end of prefetch blocks
□uint32 numItems;□□□□□// Num prefetch entries in this block
};
```

> The "name" is a hash of either the word `AI` or `FirstPerson`. A value of `5891933081285280768` is a hash of the word `AI`, and a value of `6801151928053439575` is a hash of the word `FirstPerson`.

Finally, the `PREFETCH` section ends with an array of `uint64 prefetchIDs[]` - each of these IDs will match a `streamDatabaseEntry2 t.identity` from the `INDEX` section above. The number of `prefetchID` is the specified by `streamDatabasePrefetchBlock_t.numItems`.

Data Section

The `DATA` section begins at the offset given in `streamDatabaseHeader_t.headerLength`. This section is simply a series of compressed files. The starting offset and the length (in bytes) of each file is given by a `streamDatabaseEntry2_t` in `INDEX` section.

There is often some null padding at the end of each compressed file. This is because the starting offset must be evenly divisible by 16 (because `streamDatabaseEntry2_t.offset16` is multiplied by `16` for the file offset - presumably to allow these offsets to be stored as uint32 rather than uint64).

The compressed files commonly begin with the bytes `8C 06` or `CC 06` which identifies Oodle Kraken compression.

010 Editor Template

A 010 Editor template for use with Doom Eternal's .streamdb files can be found here:

https://github.com/brongo/eternal-010-templates/blob/main/templates/DoomEternalStreamDB.bt

---

Revision #10
Created 12 September 2021 13:51:02 by SamPT
Updated 13 September 2021 11:32:42 by SamPT