

The Forensic Playground File Format

1.0

William Woodruff

March 10, 2018

1 Meta-Overview

The Forensic Playground File Format (FPFF) is an open format designed to serve as a sandbox for forensics education and competition. It has three main goals:

1. **Resemblance.** FPFF is similar to many common binary formats, making it a good tool for familiarizing students with binary layouts and parsing.
2. **Uniqueness.** FPFF is different enough from real formats, preventing automatic analysis with tools like `binwalk`.
3. **Flexibility.** FPFF's specification is simple, making extension and modification straightforward.

FPFF was developed by [UMD-CSEC](#) for [CMSC389R: Ethical Hacking](#).

The specification and a reference implementation are available on [GitHub](#) under the MIT license.

Everything below this is fictional.

2 Overview

Developed internally at Briong by lead developer Mark Thompson, the Forensic Playground File Format (FPFF) is a standards-compliant container format. This document contains the official specification for FPFF 1.0.

3 Terminology

- *File* and *stream* are used interchangeably, to denote a source of data.
- A *word* is 32 bits, or 4 bytes.
- A *dword* is 64 bits, or 8 bytes.
- A *double* is a 64 bit IEEE754 floating-point number.
- The use of **must** in any condition indicates that an FPFF parser should fail immediately if the condition is not met.
- The use of **should** indicates a user expectation that an FPFF parser may choose not to enforce.

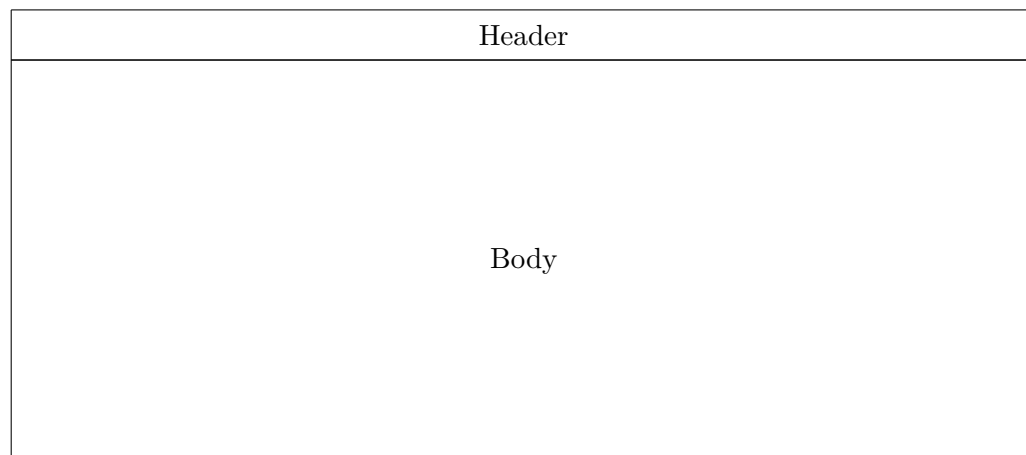
4 Specification

If FPFF data is read from a file, then that file **should** have a `.fpff` suffix.

All FPFF data is little-endian **except** for the content of section values, which may be whatever their format requires.

Unless otherwise specified, all integer fields are **unsigned**.

An FPFF file has two parts: a **header** and a **body**.



Each part is specified in detail below.

4.1 Header

The FPFF header begins at offset 0.

Its layout is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Magic (0xBEFEDADE)																															
Version (version)																															
Timestamp (timestamp)																															
Author (author)																															
Section count (nsects)																															

Each field is described below.

4.1.1 Magic

The magic field is one word.

A valid FPFF stream **must** begin with the FPFF magic bytes: 0xBEFEDADE. Any stream that does not begin with 0xBEFEDADE is not a valid FPFF stream.

4.1.2 Version

The version field is one word.

The stream's version **must** be 1, i.e. 0x1. Other versions are reserved for future FPFF specifications.

4.1.3 Timestamp

The timestamp field is one word.

The stream's timestamp **must** be a valid UNIX timestamp.¹

4.1.4 Author

The author field is a dword (8 bytes).

¹https://en.wikipedia.org/wiki/Unix_time

The stream's author **must** be ASCII-encoded². If the author is shorter than 8 bytes, then the field **must** be padded with null (0x0) bytes.

4.1.5 Section count

The section count field is one word.

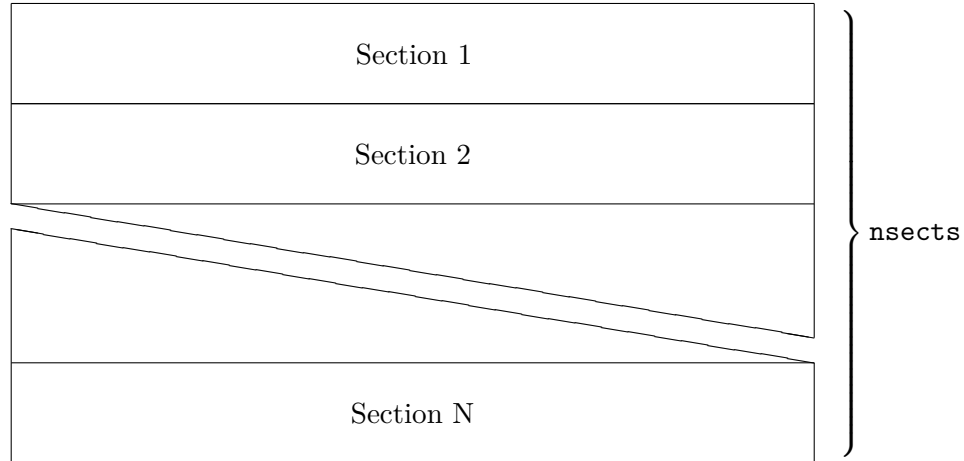
The stream's section count **must** be greater than 0.

²<https://en.wikipedia.org/wiki/ASCII>

4.2 Body

The FPFF body begins immediately after the header (offset `sizeof(header)`).

The body is a list of `nsects` sections:

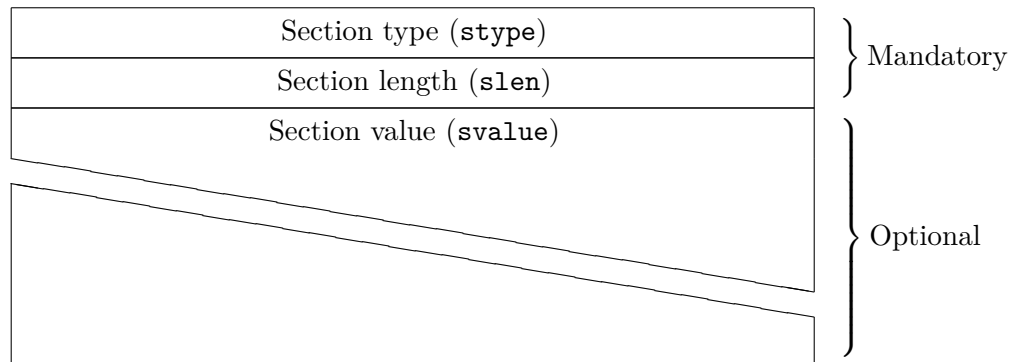


The layout of sections is described below.

4.2.1 Sections

Every section has *at least* two words: the section type (`stype`) and length (`slen`).

If `slen` is 0, then `svalue` **must** not exist. Thus, `slen` refers to the **value only** – the *total* length of the section in bytes is `slen + sizeof(stype) + sizeof(slen)`.



4.2.1.1 Section types

The `stype` field of a section indicates how to handle the section's value.

There are currently a fixed set of valid types:

- SECTION_ASCII (0x1)
- SECTION_UTF8 (0x2) – UTF-8-encoded text³.
- SECTION_WORDS (0x3) – Array of words.
- SECTION_DWORDS (0x4) – Array of dwords.
- SECTION_DOUBLES (0x5) – Array of doubles.
- SECTION_COORD (0x6) – (Latitude, longitude) tuple of doubles.
- SECTION_REFERENCE (0x7) – The index of another section.
- SECTION_PNG (0x8) – Embedded PNG image.

A section's type **must** be one of the above.

4.2.1.1.1 SECTION_ASCII

Sections of type SECTION_ASCII **must** contain `slen` bytes of ASCII-encoded text.

4.2.1.1.2 SECTION_UTF8

Sections of type SECTION_UTF8 **must** contain `slen` bytes of UTF-8-encoded text.

4.2.1.1.3 SECTION_WORDS

Sections of type SECTION_WORDS **must** contain `slen / 4` words.

4.2.1.1.4 SECTION_DWORDS

Sections of type SECTION_DWORDS **must** contain `slen / 8` dwords.

4.2.1.1.5 SECTION_DOUBLES

Sections of type SECTION_DOUBLES **must** contain `slen / 8` doubles.

4.2.1.1.6 SECTION_COORD

Sections of type SECTION_COORD **must** contain two doubles.

SECTION_COORD sections **must** have an `slen` of exactly 16.

The coordinates inside of a SECTION_COORD **should** be a valid (latitude, longitude) tuple.

³<https://en.wikipedia.org/wiki/UTF-8>

4.2.1.1.7 SECTION_REFERENCE

Sections of type SECTION_REFERENCE **must** contain one word.

SECTION_REFERENCE sections **must** have an `slen` of exactly 4.

The `svalue` of a SECTION_REFERENCE section **must** be a valid index in the range `[0, nsects - 1]`.

4.2.1.1.8 SECTION_PNG

Sections of type SECTION_PNG **must** contain `slen` bytes of PNG-encoded data⁴.

As a space-saving measure, a proper FPFF emitter **must** remove the PNG's file signature⁵. Thus, a proper FPFF parser **must** re-add the signature to produce the actual PNG.

4.2.1.2 Section length

As mentioned in 4.2.1, a section's length (`slen`) is the length of the section's value (`svalue`), **not** including the length of `stype` and `slen` themselves.

⁴https://en.wikipedia.org/wiki/Portable_Network_Graphics

⁵<http://www.libpng.org/pub/png/spec/1.2/PNG-Rationale.html#R.PNG-file-signature>