The SFrame Format

Version 2

Indu Bhagat

Table of Contents

1 Introduction

1.1 Overview

The SFrame stack trace information is provided in a loaded section, known as the `.sframe` section. When available, the `.sframe` section appears in a new segment of its own, PT_GNU_SFRAME.

The SFrame format is currently supported only for select ABIs, namely, AMD64 and AAPCS64.

A portion of the SFrame format follows an unaligned on-disk representation. Some data structures, however, (namely the SFrame header and the SFrame function descriptor entry) have elements at their natural boundaries. All data structures are packed, unless otherwise stated.

The contents of the SFrame section are stored in the target endianness, i.e., in the endianness of the system on which the section is targetted to be used. An SFrame section reader may use the magic number in the SFrame header to identify the endianness of the SFrame section.

Addresses in this specification are expressed in bytes.

The rest of this specification describes the current version of the format, `SFRAME_VERSION_2`, in detail. Additional sections outline the major changes made to each previously published version of the SFrame stack trace format.

The associated API to decode, probe and encode the SFrame section, provided via `libsframe`, is not accompanied here at this time. This will be added later.

This document is intended to be in sync with the C code in '`sframe.h`'. Please report discrepancies between the two, if any.

1.2 Changes from Version 1 to Version 2

The following is a list of the changes made to the SFrame stack trace format since Version 1 was published.

- SFrame Function Descriptor Entry encodes the size of the repetitive code blocks, e.g., pltN entries for which an FDE of type SFRAME_FDE_TYPE_PCMASK is used.
- SFrame Function Descriptor Entry includes an explicit padding of two bytes to ensure natural alignment for its data members.
- The above two imply that each SFrame Function Descriptor Entry has a fixed size of 20 bytes instead of its size of 17 bytes in SFrame format version 1.

2 SFrame section

The SFrame section consists of an SFrame header, starting with a preamble, and two other sub-sections, namely the SFrame Function Descriptor Entry (SFrame FDE) sub-section, and the SFrame Frame Row Entry (SFrame FRE) sub-section.

2.1 SFrame Preamble

The preamble is a 32-bit packed structure; the only part of the SFrame whose format cannot vary between versions.

```
typedef struct sframe_preamble
{
  uint16_t sfp_magic;
  uint8_t sfp_version;
  uint8_t sfp_flags;
} ATTRIBUTE_PACKED sframe_preamble;
```

Every element of the SFrame preamble is naturally aligned.

All values are stored in the endianness of the target system for which the SFrame section is intended. Further details:

Offset	Name	Description
0x00	`uint16_t sfp_magic`	The magic number for SFrame section: 0xdee2. Defined as a macro `SFRAME_MAGIC`.
0x02	`uint8_t sfp_version`	The version number of this SFrame section. See Section 2.1.2 [SFrame version], page 2, for the set of valid values. Current version is `SFRAME_VERSION_1`.
0x03	`uint8_t sfp_flags`	Flags (section-wide) for this SFrame section. See Section 2.1.3 [SFrame flags], page 3, for the set of valid values.

2.1.1 SFrame endianness

SFrame sections are stored in the target endianness of the system that consumes them. The SFrame library (`libsframe`) can, however, detect whether to endian-flip an SFrame section at decode time, by inspecting the `sfp_magic` field in the SFrame header (If it appears as 0xe2de, endian-flipping is needed).

2.1.2 SFrame version

The version of the SFrame format can be determined by inspecting `sfp_version`. The following versions are currently valid:

Version	Number	Description
`SFRAME_VERSION_1`	1	First version, obsolete.
`SFRAME_VERSION_2`	2	Current version, under development.

This document describes `SFRAME_VERSION_2`.

2.1.3 SFrame flags

The preamble contains bitflags in its `sfp_flags` field that describe various section-wide properties.

The following flags are currently defined.

Flag	Versions	Value	Meaning
`SFRAME_F_FDE_SORTED`	All	0x1	Function Descriptor Entries are sorted on PC.
`SFRAME_F_FRAME_POINTER`	All	0x2	Functions preserve frame-pointer.

Further flags may be added in future.

2.2 SFrame Header

The SFrame header is the first part of an SFrame section. It begins with the SFrame preamble. All parts of it other than the preamble (see Section 2.1 [SFrame Preamble], page 2) can vary between SFrame file versions. It contains things that apply to the section as a whole, and offsets to the various other sub-sections defined in the format. As with the rest of the SFrame section, all values are stored in the endianness of the target system.

The two sub-sections tile the SFrame section: each section runs from the offset given until the start of the next section. An explicit length is given for the last sub-section, the SFrame Frame Row Entry (SFrame FRE) sub-section.

```
typedef struct sframe_header
{
  sframe_preamble sfh_preamble;
  uint8_t sfh_abi_arch;
  int8_t sfh_cfa_fixed_fp_offset;
  int8_t sfh_cfa_fixed_ra_offset;
  uint8_t sfh_auxhdr_len;
  uint32_t sfh_num_fdes;
  uint32_t sfh_num_fres;
  uint32_t sfh_fre_len;
  uint32_t sfh_fdeoff;
  uint32_t sfh_freoff;
} ATTRIBUTE_PACKED sframe_header;
```

Every element of the SFrame header is naturally aligned.

The sub-section offsets, namely `sfh_fdeoff` and `sfh_freoff`, in the SFrame header are relative to the *end* of the SFrame header; they are each an offset in bytes into the SFrame section where the SFrame FDE sub-section and the SFrame FRE sub-section respectively start.

SFrame header allows specifying explicitly the fixed offsets from CFA, if any, from which FP or RA may be recovered. For example, in AMD64, the stack offset of the return address is `CFA - 8`. Since this offset is in close vicinity with the CFA in most ABIs, `sfh_cfa_fixed_fp_offset` and `sfh_cfa_fixed_ra_offset` are limited to signed 8-bit integers.

SFrame format has made some provisions for supporting more ABIs/architectures in the future. The `sframe_header` structure provides an unsigned 8-bit integral field to denote the size of an auxiliary SFrame header. The auxiliary SFrame header follows right after the `sframe_header` structure. As for the offset calculations, the *end* of SFrame header must be the end of the auxiliary SFrame header, if the latter is present.

Putting it all together:

Offset	Name	Description
0x00	`sframe_preamble sfh_preamble`	The SFrame preamble. See Section 2.1 [SFrame Preamble], page 2.
0x04	`uint8_t sfh_abi_arch`	The ABI/arch identifier. See Section 2.2.1 [SFrame ABI/arch identifier], page 5.
0x05	`int8_t sfh_cfa_fixed_fp_offset`	The CFA fixed FP offset, if any.
0x06	`int8_t sfh_cfa_fixed_ra_offset`	The CFA fixed RA offset, if any.
0x07	`uint8_t sfh_auxhdr_len`	Size in bytes of the auxiliary header that follows the `sframe_header` structure.
0x08	`uint32_t sfh_num_fdes`	The number of SFrame FDEs in the section.
0xc	`uint32_t sfh_num_fres`	The number of SFrame FREs in the section.
0x10	`uint32_t sfh_fre_len`	The length in bytes of the SFrame FRE sub-section.
0x14	`uint32_t sfh_fdeoff`	The offset in bytes of the SFrame FDE sub-section. This sub-section contains `sfh_num_fdes` number of fixed-length array elements. The array element is of type SFrame function desciptor entry, each providing a high-level function description for backtracing. See Section 2.3 [SFrame Function Descriptor Entries], page 5.
0x18	`uint32_t sfh_freoff`	The offset in bytes of the SFrame FRE sub-section, the core of the SFrame section, which describes the stack trace information using variable-length array elements. See Section 2.4 [SFrame Frame Row Entries], page 8.

2.2.1 SFrame ABI/arch identifier

SFrame header identifies the ABI/arch of the target system for which the executable and hence, the stack trace information contained in the SFrame section, is intended. There are currently three identifiable ABI/arch values in the format.

ABI/arch Identifier	Value	Description
`SFRAME_ABI_AARCH64_ENDIAN_BIG`	1	AARCH64 big-endian
`SFRAME_ABI_AARCH64_ENDIAN_LITTLE`	2	AARCH64 little-endian
`SFRAME_ABI_AMD64_ENDIAN_LITTLE`	3	AMD64 little-endian

The presence of an explicit identification of ABI/arch in SFrame may allow stack trace generators to make certain ABI-specific decisions.

2.3 SFrame FDE

The SFrame Function Descriptor Entry sub-section is a sorted array of fixed-length SFrame function descriptor entries (SFrame FDEs). Each SFrame FDE is a packed structure which contains information to describe a function's stack trace information at a high-level.

```
typedef struct sframe_func_desc_entry
{
  int32_t sfde_func_start_address;
  uint32_t sfde_func_size;
  uint32_t sfde_func_start_fre_off;
  uint32_t sfde_func_num_fres;
  uint8_t sfde_func_info;
  uint8_t sfde_func_rep_size;
  uint16_t sfde_func_padding2;
} ATTRIBUTE_PACKED sframe_func_desc_entry;
```

Every element of the SFrame function descriptor entry is naturally aligned.

`sfde_func_start_fre_off` is the offset to the first SFrame FRE for the function. This offset is relative to the *end of the SFrame FDE* sub-section (unlike the offsets in the SFrame header, which are relative to the *end* of the SFrame header).

`sfde_func_info` is the "info word", containing information on the FRE type and the FDE type for the function See Section 2.3.1 [The SFrame FDE info word], page 6.

Following table describes each component of the SFrame FDE structure:

Offset	Name	Description
0x00	`int32_t sfde_func_start_address`	Signed 32-bit integral field denoting the virtual memory address of the described function.

0x04	`uint32_t sfde_func_size`	Unsigned 32-bit integral field specifying the size of the function in bytes.
0x08	`uint32_t sfde_func_start_fre_off`	Unsigned 32-bit integral field specifying the offset in bytes of the function's first SFrame FRE in the SFrame section.
0x0c	`uint32_t sfde_func_num_fres`	Unsigned 32-bit integral field specifying the total number of SFrame FREs used for the function.
0x10	`uint8_t sfde_func_info`	Unsigned 8-bit integral field specifying the SFrame FDE info word. See Section 2.3.1 [The SFrame FDE info word], page 6.
0x11	`uint8_t sfde_func_rep_size`	Unsigned 8-bit integral field specifying the size of the repetitive code block for which an SFrame FDE of type SFRAME_FDE_TYPE_PCMASK is used. For example, in AMD64, the size of a pltN entry is 16 bytes.
0x12	`uint16_t sfde_func_padding2`	Padding of 2 bytes. Currently unused bytes.

2.3.1 The SFrame FDE info word

The info word is a bitfield split into three parts. From MSB to LSB:

Bit offset	Name	Description
7–6	`unused`	Unused bits.
5	`pauth_key`	Specify which key is used for signing the return addresses in the SFrame FDE. Two possible values: SFRAME_AARCH64_PAUTH_KEY_A (0), or SFRAME_AARCH64_PAUTH_KEY_B (1).
4	`fdetype`	Specify the SFrame FDE type. Two possible values: SFRAME_FDE_TYPE_PCMASK (1), or SFRAME_FDE_TYPE_PCINC (0). See Section 2.3.2 [The SFrame FDE types], page 7.
0–3	`fretype`	Choice of three SFrame FRE types. See Section 2.3.3 [The SFrame FRE types], page 7.

2.3.2 The SFrame FDE types

SFrame format defines two types of FDE entries. The choice of which SFrame FDE type to use is made based on the instruction patterns in the relevant program stub.

An SFrame FDE of type `SFRAME_FDE_TYPE_PCINC` is an indication that the PCs in the FREs should be treated as increments in bytes. This is used fo the the bulk of the executable code of a program, which contains instructions with no specific pattern.

In contrast, an SFrame FDE of type `SFRAME_FDE_TYPE_PCMASK` is an indication that the PCs in the FREs should be treated as masks. This type is useful for the cases where a small pattern of instructions in a program stub is used repeatedly for a specific functionality. Typical usecases are pltN entries and trampolines.

Name of SFrame FDE type	**Value**	**Description**
SFRAME_FDE_TYPE_PCINC	0	Unwinders perform a (PC >= FRE_START_ADDR) to look up a matching FRE.
SFRAME_FDE_TYPE_PCMASK	1	Unwinders perform a (PC % REP_BLOCK_SIZE >= FRE_START_ADDR) to look up a matching FRE. REP_BLOCK_SIZE is the size in bytes of the repeating block of program instructions.

2.3.3 The SFrame FRE types

A real world application can have functions of size big and small. SFrame format defines three types of SFrame FRE entries to represent the stack trace information for such a variety of function sizes. These representations vary in the number of bits needed to encode the start address offset in the SFrame FRE.

The following constants are defined and used to identify the SFrame FRE types:

Name	**Value**	**Description**
`SFRAME_FRE_TYPE_ADDR1`	0	The start address offset (in bytes) of the SFrame FRE is an unsigned 8-bit value.
`SFRAME_FRE_TYPE_ADDR2`	1	The start address offset (in bytes) of the SFrame FRE is an unsigned 16-bit value.
`SFRAME_FRE_TYPE_ADDR4`	2	The start address offset (in bytes) of the SFrame FRE is an unsigned 32-bit value.

A single function must use the same type of SFrame FRE throughout. An identifier to reflect the chosen SFrame FRE type is stored in the See Section 2.3.1 [The SFrame FDE info word], page 6.

2.4 SFrame FRE

The SFrame Frame Row Entry sub-section contains the core of the stack trace information.

An SFrame Frame Row Entry is a self-sufficient record containing SFrame stack trace information for a range of contiguous addresses, starting at the specified offset from the start of the function. Each SFrame Frame Row Entry is followed by S*N bytes, where:

- `S` is the size of the stack frame offset for the FRE, and
- `N` is the number of stack frame offsets in the FRE

The stack offsets, following the FRE, are interpreted in order as follows:

- The first offset is always used to locate the CFA, by interpreting it as: CFA = `BASE_REG` `+` offset1.
- If RA is being tracked, the second offset is always used to locate the RA, by interpreting it as: RA = CFA `+` offset2. If RA is *not* being tracked *and* FP is being tracked, the second offset will be used to locate the FP, by interpreting it as: FP = CFA `+` offset2.
- If both RA and FP are being tracked, the third offset will be used to locate the FP, by interpreting it as FP = CFA `+` offset3.

The entities `S`, `N` and `BASE_REG` are identified using the SFrame FRE info word, a.k.a. the `sframe_fre_info` See Section 2.4.1 [The SFrame FRE info word], page 9.

Following are the definitions of the allowed SFrame FRE:

```
typedef struct sframe_frame_row_entry_addr1
{
  uint8_t sfre_start_address;
  sframe_fre_info sfre_info;
} ATTRIBUTE_PACKED sframe_frame_row_entry_addr1;

typedef struct sframe_frame_row_entry_addr2
{
  uint16_t sfre_start_address;
  sframe_fre_info sfre_info;
} ATTRIBUTE_PACKED sframe_frame_row_entry_addr2;

typedef struct sframe_frame_row_entry_addr4
{
  uint32_t sfre_start_address;
  sframe_fre_info sfre_info;
} ATTRIBUTE_PACKED sframe_frame_row_entry_addr4;
```

For ensuring compactness, SFrame frame row entries are stored unaligned on disk. Appropriate mechanisms need to be employed, as necessary, by the serializing and deserializing entities, if unaligned accesses need to be avoided.

`sfre_start_address` is an unsigned 8-bit/16-bit/32-bit integral field identifies the start address of the range of program counters, for which the SFrame FRE applies. The value encoded in the `sfre_start_address` field is the offset in bytes of the start address of the SFrame FRE, from the start address of the function.

Further FRE types may be added in future.

2.4.1 The SFrame FRE info word

The SFrame FRE info word is a bitfield split into four parts. From MSB to LSB:

Bit offset	Name	Description
7	`fre_mangled_ra_p`	Indicate whether the return address is mangled with any authorization bits (signed RA).
5-6	`fre_offset_size`	Size of stack offsets in bytes. Valid values are: SFRAME_FRE_OFFSET_1B, SFRAME_FRE_OFFSET_2B, and SFRAME_FRE_OFFSET_4B.
1-4	`fre_offset_count`	A value of upto 3 is allowed to track all three of CFA, FP and RA.
0	`fre_cfa_base_reg_id`	Distinguish between SP or FP based CFA recovery.

Name	Value	Description
`SFRAME_FRE_OFFSET_1B`	0	All stack offsets following the fixed-length FRE structure are 1 byte long.
`SFRAME_FRE_OFFSET_2B`	1	All stack offsets following the fixed-length FRE structure are 2 bytes long.
`SFRAME_FRE_OFFSET_4B`	2	All stack offsets following the fixed-length FRE structure are 4 bytes long.

Index