Network Working Group
Request for Comments: 4194
Category: Standards Track

J. Strombergson
InformAsic AB
L. Walleij
Lunds Tekniska Hogskola
P. Faltstrom
Cisco Systems Inc

October 2005

The S Hexdump Format

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the “Internet Official Protocol Standards” (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document specifies the S Hexdump Format (SHF), a new, XML-based open format for describing binary data in hexadecimal notation. SHF provides the ability to describe both small and large, simple and complex hexadecimal data dumps in an open, modern, transport- and vendor-neutral format.

1. Introduction

In the computing, network, and embedded systems communities, several different types of data formats for hexadecimal data are being used. One of the more common formats is known as "S-records" (and several derivatives), which reportedly originated at the Motorola company. The S Hexdump Format is named in its honour.

Typical uses of these dump formats include executable object code for embedded systems (i.e., "firmware"), on-chip flash memories and filesystems, FPGA configuration bitstreams, graphics and other application resources, routing tables, etc. Unfortunately, none of the formats used are truly open, vendor-neutral, and/or well-defined.

Even more problematic is the fact that none of these formats are able to represent the large data sizes that are getting more and more common. Data dumps comprised of multiple sub-blocks with different Word sizes, and data sizes spanning anywhere from a few Bytes of data to much larger than 2^32 bits are not handled. Also, the checksums included in these formats are too simplistic and for larger data sizes, they provide insufficient ability to accurately detect errors. Alternatively, the overhead needed for proper error detection is very large.

Therefore, the S Hexdump format is an effort to provide a modern, XML-based format that is not too complex for simple tools and computing environments to implement, generate, parse, and use. Yet the format is able to handle large data sizes and complex data structures, and can provide high quality error detection by leveraging standardized cryptographic hash functions.

One of the simplifications introduced in the format is to disallow other number systems such as octal or decimal notation, and to allow for Word sizes of even bytes (8-bit groups) only. This is intentional and was done to simplify implementations aimed for practical present-day applications. Formats aimed for esoteric number systems or odd Word sizes may be implemented elsewhere.

At present, the usage of the SHF format may be mainly for Internet transport and file storage on development machinery. A parser for the XML format is presently not easily deployed in hardware devices, but the parsing and checksumming of the SHF data may be done by a workstation computer, which in turn converts the SHF tokens to an ordinary bitstream before the last step (e.g., of a firmware upgrade) commences.

SHF is a dump format only and shall not be confused with similar applications, such as binary configuration formats or patches, which are intended to, for example, alter contents of a core memory. Such applications require the possibility of modifying individual bits or groups of bits in the memory of a machine, and is not the intended usage of the mechanism described in the present document.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

The key word "Byte" is to be interpreted as a group of 8 bits. The key word "Octet" is another name for Byte.

The key word "Word" is to be interpreted as a group containing an integral number of Bytes.

The key word "Block" is to be interpreted as an ordered sequence of Words, beginning at a certain address, running from lower to higher addresses. A Block typically represents a sequence of Words at a certain address range in the memory of a computer.

The key word "Dump" is to be interpreted as a sequence of Blocks, which may or may not be in a particular order. A Dump typically represents some non-continuous, interesting parts of the memory of a computer, such that the Dump as a whole has a certain meaning, for example (but not limited to) a complete firmware for an embedded system.

The expression "2^n" is to be interpreted as the value two (2) raised to the n:th power. For example, 2^8 equals the value 256.

3. Features and Functionality

The SHF-format has the following features:

- Support for arbitrarily wide data Words
- Support for very large data Blocks
- Support for an arbitrary number of independent data Blocks
- Data integrity detection against errors provided by the RFC3174 specified (see [2]) SHA-1 cryptographic signature
- An XML-based format

In the embedded systems domain, 8- and 16-bit processors are still used in large numbers and will continue to be used for any foreseeable future. Simultaneously, more and more systems are using 64-bit and even larger Word sizes.

SHF supports all of these systems by allowing the Word size to be specified. The Word size MUST be an integer number of Bytes and at least one (1) Byte.

SHF is able to represent both large and small data Blocks. The data Block MUST contain at least one (1) Word. Additionally, the data Block MUST NOT be larger than (2^64)-1 bits.

The SHF Dump MUST contain at least one (1) data Block. The maximum number of Blocks supported is 2^64. Each data Block in the Dump MAY have different Word sizes and start at different addresses.

The checksum (or message digest) used to verify the correctness or data integrity of each Block is 20 Bytes (160 bits) long. The digest MUST be calculated on the data actually represented by the SHF data Block, NOT the representation, i.e., NOT the ASCII-code. SHA-1 is only able to calculate a digest for a data Block no larger than (2^64)-1 bits and this limits the size of each data Block in SHF to (2^64)-1 bits.

4. SHF XML Specification

The SHF format consists of an XML data structure representing a Dump. The Dump consists of a Dump header section and one (1) or more Block sections containing data. Each Block of data is independent of any other Block.

A short, symbolic example of an SHF Dump is illustrated by the following structure:

```
<dump name="(Human readable string)" blocks="(64-bit value)">
  <block name="(Human readable string)" start_address="(64-bit
         value)" word_size="(64-bit value)" length="(64-bit value)"
         checksum="(20-Byte digest)">
     (Data)
  </block>
</dump>
```

4.1. Header Section

The header section comprises the Dump tag, which includes the following attributes:

- name: A compulsory string of arbitrary length used by any interested party to identify the specific SHF Dump.
- blocks: An optional 64-bit hexadecimal value representing the number of Blocks in the specific SHF Dump. Whenever available, this value should be supplied. However, there are potential scenarios where the number of Blocks cannot be given beforehand. If the value is present, it should be verified by implementers; if the value is untrue, the behaviour is implementation-defined.

After the opening Dump tag, one or more subsections of Blocks must follow. Finally, the complete SHF Dump ends with a closing Dump tag.

4.2. Block Subsection

The Block subsection contains a Block tag and a number of data words. The Block tag includes the following attributes:

- name: A compulsory string of arbitrary length used by any interested party to identify the specific Block.
- start_address: A compulsory, 64-bit hexadecimal value representing the start address in Bytes for the data in the Block.
- word_size: A compulsory 64-bit hexadecimal value representing the number of Bytes (the width) of one Word of the data.
- length: A compulsory hexadecimal representation of an unsigned 64-bit integer indicating the number of Words following inside the Block element. If this value turns out to be untrue, the Block MUST be discarded.
- checksum: A compulsory hexadecimal representation of the 20 Byte SHA-1 digest of the data in the Block.

The total size of the data in the Block (in bits) is given by the expression (8 * word_size * length). The expression MUST NOT be larger than (2^64)-1.

After the opening Block tag, a hexadecimal representation of the actual data in the Block follows. Finally, the Block section ends with a closing Block tag.

5. SHF Rules and Limits

There are several rules and limits in SHF:

- All attribute values representing an actual value and the data MUST be in hexadecimal notation. The only attribute excluded from this rule is the name attribute in the Dump and Block tags. This restriction has been imposed for ease of reading the dump: a reader shall not be uncertain about whether a figure is in hex notation or not, and can always assume it is hexadecimal.
- All attribute values, with the exception of the checksum, MAY omit leading zeros. Conversely, the checksum MUST NOT omit leading zeros.
- The data represented in a Block MUST NOT be larger than (2^64)-1 bits.
- The size of a Word MUST NOT be larger than (2^64)-1 bits. This implies that a Block with a Word defined to the maximum width cannot contain more than one Word. An SHF consumer shall assure that it can handle a certain Word length before beginning to parse blocks of an SHF Dump. Failure to do so may cause buffer overflows and endanger the stability and security of the system running the consuming application.
- The attribute values representing an actual value MUST be in big-endian format. This means that the most significant hexadecimal digits are to be put to the left in a hexadecimal Word, address, or similar field. For example, the address value 1234 represents the address 1234 and not 3412. While some computing architectures may be using little-endian Words as their native format, it is the responsibility of any SHF producer running on such an architecture to swap the attribute values to a big-endian format. The reverse holds for a consumer receiving the big-endian SHF attributes: if the consumer is little-endian, the values have to be swapped around.
- Likewise, the words inside a Dump MUST be stored in a big-endian format if the word size is larger than one Byte. Here, the same need for swapping Bytes around may arise, as mentioned in the previous paragraph.

6. SHF DTD

The contents of the element named "block" and the attributes "blocks", "address", "word_size" and "checksum" should only contain the characters that are valid hexbyte sequences. These are:

```
whitespace ::= (#x20 | #x9 | #xC | #xD | #xA)
hexdigit   ::= [0-9A-Fa-f]
hexbytes   ::= whitespace* hexdigit (hexdigit|whitespace)*
```

A parser reading in an SHF file should silently ignore any other characters that (by mistake) appear in any of these elements or attributes. These alien characters should be treated as if they did not exist. Also note that "whitespace" has no semantic meaning; it is only valid for the reason of improving the human readability of the Dump. Whitespace may be altogether removed and the hexbyte sequences concatenated if desired. Notice that the fact that word size is to be given in a number of bytes implies that the number of hexadecimal digits inside a block need to be even. Malformed blocks should be ignored by implementations.

```
<!--
  DTD for the S Hexdump Format, as of 2003-10-10
  Linus Walleij, Joachim Strombergson, Patrik Faltstrom 2003

  Refer to this DTD as:

  <!ENTITY % SHF PUBLIC "-//IETF//DTD SHF//EN"
             "http://ietf.org/dtd/shf.dtd">
       %SHF;
-->
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT dump (block)+>
<!ATTLIST dump
       name          CDATA    #REQUIRED
       blocks        CDATA    #IMPLIED>

<!ELEMENT block (#PCDATA)>
<!ATTLIST block
       name          CDATA    #REQUIRED
       address       CDATA    #REQUIRED
       word_size     CDATA    #REQUIRED
       length        CDATA    #REQUIRED
       checksum      CDATA    #REQUIRED>
```

7. SHF Examples

This section contains three different SHF examples, illustrating the usage of SHF and the attributes in SHF.

The first example is a simple SHF Dump with a single Block of data:

```
<?xml version="1.0" encoding="UTF-8"?>
<dump name="Simple SHF example" blocks="01">
  <block name="Important message in hex format" address="0400"
    word_size="01" length="1f"
    checksum="5601b6acad7da5c7b92036786250b053f05852c3">
      41 6c 6c 20 79 6f 75 72 20 62 61 73 65 20 61 72
      65 20 62 65 6c 6f 6e 67 20 74 6f 20 75 73 0a
  </block>
</dump>
```

The second example is a program in 6502 machine code residing at memory address 0x1000, which calculates the 13 first Fibonacci numbers and stores them at 0x1101-0x110d:

```
<?xml version="1.0" encoding="UTF-8"?>
<dump name="6502 Fibonacci" blocks="02">
  <block name="Code" address="1000" word_size="01" length="2a"
    checksum="5cab5bf8ee299af1ad17e8093d941914eb5930c7">
      a9 01 85 20 85 21 20 1e 10 20 1e 10 18 a5 21 aa
      65 20 86 20 85 21 20 1e 10 c9 c8 90 ef 60 ae 00
      11 a5 21 9d 00 11 ee 00 11 60
  </block>
  <block name="Mem" address="1100" word_size="01" length="e"
    checksum="c8c2001c42b0226a5d9f7c2f24bd47393166487a">
      01 00 00 00 00 00 00 00 00 00 00 00 00 00
  </block>
</dump>
```

The final example contains a Block of 40-bit wide data:

```
<?xml version="1.0" encoding="UTF-8"?>
<dump name="Example of an SHF dump with wide data words" blocks="00001">
  <block name="SMIL memory dump" address="000" word_size="5"
        length="1A" checksum="ff2033489aff0e4e4f0cd7901afc985f7a213c97">
      00100 00200 00000 00090 00000 00036 00300 00400
      00852 00250 00230 00858 00500 00600 014DC 00058
      002A8 000B8 00700 00800 000B0 00192 00100 00000
      00900 00A00 00000 0000A 40000 00000 00B00 00C00
      00000 00000 00000 00001 00D00 00E00 00000 00100
      0CCCC CCCCD 00F00 01000 00000 00010 80000 00000
      00100 00790 00000 00234
  </block>
</dump>
```

8. SHF Security Considerations

The SHF format is a format for representing hexadecimal data that one wants to transfer, manage, or transform. The format itself does not guarantee that the represented data is not falsely represented, malicious, or otherwise dangerous.

The data integrity of the SHF file as a whole is to be provided, if needed, by means external to the SHF file, such as the generic signing mechanism described by RFC 3275 [3].

9. IANA Considerations

This section contains the registration information for the MIME type to SHF. The media type has been chosen to comply with the guidelines in [4].

- Registration: application/shf+xml
- MIME media type name: application
- MIME subtype name: shf+xml
- Required parameters: charset

Required parameters: charset

This parameter must exist and must be set to "UTF-8". No other character sets are allowed for transporting SHF data. The character set designator MUST be uppercase.

Encoding considerations:

This media type may contain binary content; accordingly, when used over a transport that does not permit binary transfer, an appropriate encoding must be applied.

Security considerations:

A hex Dump in itself has no other security considerations than what applies for any other XML file. However, the included binary data may in decoded form contain any executable code for a target platform. If additional security is desired, additional transport security solutions may be applied. For target code contained in a hex Dump, developers may want to include certificates, checksums, and the like in hexdump form for the target platform. Such uses are outside the scope of this document and a matter of implementation.

Interoperability considerations:

n/a

Published specification:

This media type is a proper subset of the XML 1.0 specification [5]. One restriction is made: no entity references other than the five predefined general entities references ("&", "<", ">", "'", and """) and numeric entity references may be present. Neither the "XML" declaration (e.g., <?xml version="1.0" ?>) nor the "DOCTYPE" declaration (e.g., <!DOCTYPE ...>) need be present. (XML fragments are allowed.) All other XML 1.0 instructions (e.g., CDATA blocks, processing instructions, and so on) are allowed. Applications that use this media type: any program or individual wishing to make use of this XML 1.0 subset for hexdump exchange.

Additional information:

- Magic number: There is no single initial Byte sequence that is always present for SHF files
- File extension: shf
- Macintosh File Type code: none

Intended usage: COMMON.

Author/Change controller: this MIME transport type is controlled by the IETF.

10. Extensions

The attributes of elements in the SHF XML format may be extended when need arises. For example, certain applications will want to represent executable code as an SHF Dump, and may then need an execution start address to be associated with certain Dump Blocks, so that the address can be configured as a starting point for the CPU part of any processor code present in the Block, as opposed to the raw data, which is already given a start address by way of the "address" attribute. This can be done by extending the Block tag with a "start_address" attribute.

Another possible scenario is when a dump is applied to a computer system with several independent address spaces, such as a system with two CPUs, each with independent memories. In this case, a user may want to add an "address_space" attribute.

As long as such new attributes are added, with no attributes being removed or redefined, the resulting Dump shall be considered a valid SHF Dump and transported using the application/xml+shf transport type. Parsers unaware of the modified namespace shall silently ignore any such extended attributes, or simply duplicate them from input to output when processing an SHF file as a filter. The management of such extended attributes is a matter of convention between different classes of users and not a matter of the IETF.

11. Additional Information

Contact for further information: c.f., the "Authors' Addresses" section of this memo.

Acknowledgements: The SMIL memory Dump was kindly provided by Sten Henriksson at Lund University. Proofreading and good feedback on the SHF document was generously provided by Peter Lindgren, Tony Hansen, Larry Masinter, and Clive D.W. Feather. We also want to thank the Applications area workgroup for their help during development.

12. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[3] Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.

[4] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[5] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", <http://www.w3.org/TR/REC-xml>.

Authors' Addresses

Joachim Strombergson
InformAsic AB
Hugo Grauers gata 5a
Gothenburg, 411 33
SE
Phone: +46 31 68 54 90
EMail: Joachim.Strombergson@InformAsic.com
URI: http://www.InformAsic.com/

Linus Walleij
Lunds Tekniska Hogskola
Master Olofs Vag 24
Lund, 224 66
SE
Phone: +46 703 193678
EMail: triad@df.lth.se

Patrik Faltstrom
Cisco Systems Inc
Ledasa
Lovestad, 273 71
Sweden
EMail: paf@cisco.com
URI: http://www.cisco.com

Full Copyright Statement

Intellectual Property

[1] http://www.ietf.org/ipr

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org[2].

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

[2] mailto:ietf-ipr@ietf.org