Groupe de travail Réseau
Request for Comments : 4194
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

J. Strombergson, InformAsic AB
L. Walleij, Lunds Tekniska Hogskola
P. Faltstrom, Cisco Systems Inc
octobre 2005

Format S Hexdump

Résumé
Le présent document spécifie le format S Hexdump (SHF, *S Hexdump Format*) un nouveau format ouvert, fondé sur XML, pour décrire des données binaires en notation hexadécimale. SHF apporte la capacité de décrire des dépôts de données hexadécimales à la fois petits et grands, simples et complexes, dans un format ouvert, moderne, neutre quant au transport et au fabricant.

Table of Contents

1. Introduction

Dans les communautés de l'informatique, des réseaux, et des systèmes incorporés, plusieurs différents types de formats de données sont utilisés pour les données hexadécimales. Un des formats les plus connus sous le nom de "S-records" (et plusieurs dérivés) à son origine chez la société Motorola. Le format S Hexdump est ainsi nommé en son honneur.

Les utilisations typiques de ces formats de dépôts incluent du code d'objet exécutable pour les systèmes incorporés (c'est-à-dire, du "microcode") sur des puces de mémoires flash et des systèmes de fichiers, des flux binaires de configuration FPGA, des ressources graphiques et d'autres applications, des tableaux d'acheminement, etc. Malheureusement, aucun des formats utilisés n'est vraiment ouvert, neutre quant au fabricant, et/ou bien défini.

Encore plus problématique est le fait qu'aucun de ces formats n'est capable de représenter les grandes tailles de données qui sont de plus en plus courantes. Les dépôts de données comportant de nombreux sous blocs avec des tailles de mot différentes, et des tailles de données s'étendant de quelques octets de données à plus de 2^32 bits ne sont pas traités. Aussi, les sommes de contrôle incluses dans ces formats sont trop simplistes et pour de plus grandes tailles de données, elles donnent une capacité insuffisante de détection précise des erreurs. Autrement, les frais généraux nécessaires pour une détection d'erreur appropriée sont très importants.

Donc, le format S Hexdump est un effort pour fournir un format moderne, fondé sur XML qui ne soit pas trop complexe pour des outils et des environnements informatiques simples à mettre en œuvre, générer, analyser et utiliser. Déjà le format est capable de traiter de grandes tailles de données et des structures de données complexes, et peut fournir une détection d'erreur de grande qualité en déployant des fonctions standard de hachage cryptographique.

Une des simplifications introduites dans le format est d'interdire d'autres systèmes numériques comme la notation octale ou décimale, et de ne permettre que des tailles de mots paires d'octets (groupes de 8 bits). Ceci est intentionnel et a été fait pour simplifier les mises en œuvre visant les applications pratiques d'aujourd'hui. Les formats visant des systèmes numériques ésotériques ou des tailles de mot impaires peuvent être mises en œuvre ailleurs.

À présent, l'usage du format SHF peut être principalement pour le transport Internet et la mémorisation de fichiers sur une machinerie de développement. Un analyseur pour le format XML n'est actuellement pas facilement déployé dans les matériels, mais l'analyse et la somme de contrôle des données SHF peuvent être faites par un ordinateur de bureau, qui à son tour convertira les jetons SHF en flux binaire ordinaire avant que la dernière étape (par exemple, une mise à niveau de microcode) ne commence.

SHF est seulement un format de dépôt et ne doit pas être confondu avec des applications similaires, comme des formats de configuration binaire ou des réparations, qui sont destinées, par exemple, à altérer le contenu d'un cœur de mémoire. De telles applications exigent la possibilité de modifier des bits individuels ou des groupes de bits dans la mémoire d'une machine, et ce n'est pas l'usage prévu pour le mécanisme décrit dans le présent document.

2. Terminologie

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Le mot clé "Byte" *(octet)* est à interpréter comme un groupe de 8 bits. Le mot clé "Octet" est un autre nom pour Byte.

Le mot clé "Word" *(mot)* est à interpréter comme un groupe contenant un nombre entier d'octets.

Le mot clé "Block" *(bloc)* est à interpréter comme une séquence ordonnée de mots, commençant à une certaine adresse, allant des adresses inférieures aux adresses supérieures. Un bloc représente normalement une séquence de mots à une certaine gamme d'adresses dans la mémoire d'un ordinateur.

Le mot clé "Dump" *(dépôt)* est à interpréter comme une séquence de blocs, qui peuvent être ou non dans un ordre particulier. Un dépôt représente normalement des parties intéressantes non continues de la mémoire d'un ordinateur, telles que le dépôt a comme un tout une certaine signification, par exemple (mais sans s'y limiter) un microcode complet pour un système incorporé.

L'expression "2^n" est à interpréter comme la valeur deux (2) à la puissance nième. Par exemple, 2^8 vaut 256.

3. Caractéristiques et fonctionnalités

Le format SHF a les caractéristiques suivantes :
- prise en charge de mots de données de longueur arbitraire,
- prise en charge de blocs de données très grands,
- prise en charge d'un nombre arbitraire de blocs de données indépendants,
- détection de l'intégrité des données à l'égard des erreurs, fournie par la signature cryptographique SHA-1 spécifiée par la [RFC3174],
- un format fondé sur XML.

Dans le domaine des systèmes incorporés, les processeurs de 8 et 16 bits sont encore utilisés en grands nombres et continueront de l'être à l'avenir pour autant qu'on puisse le prévoir. Simultanément, de plus en plus de systèmes utilisent des tailles de mots de 64 bits et même plus.

SHF prend en charge tous ces systèmes en permettant de spécifier la taille de mot. La taille de mot DOIT être un nombre entier d'octets et d'au moins un octet.

SHF est capable de représenter des blocs de données aussi bien grands que petits. Le bloc de données DOIT contenir au moins un (1) mot. De plus, le bloc de données NE DOIT PAS faire plus de (2^64)-1 bits.

Le dépôt SHF DOIT contenir au moins un (1) bloc de données. Le nombre maximum de blocs supporté est 2^64. Chaque bloc de données dans le dépôt PEUT avoir des tailles de mot différentes et commencer à des adresses différentes.

La somme de contrôle (ou résumé de message) utilisée pour vérifier la correction ou l'intégrité des données de chaque bloc est de 20 octets (160 bits). Le résumé DOIT être calculé sur les données réellement représentées par le bloc de données SHF, et NON sur la représentation, c'est-à-dire, PAS sur le code ASCII. SHA-1 est seulement capable de calculer un résumé pour un bloc de données inférieur ou égal à (2^64)-1 bits et cela limite la taille de chaque bloc de données en SHF à (2^64)-1 bits.

4. Spécification du SHF XML

Le format SHF consiste en une structure de données XML représentant un dépôt. Le dépôt consiste en une section d'en-tête de dépôt et une (1) ou plusieur sections de bloc contenant les données. Chaque bloc de données est indépendant de tout autre bloc.

Un bref exemple symbolique de dépôt SHF est illustré par la structure suivante :

```
<nom du dépôt="()" blocs="(valeur de 64 bits)">
  <nom de bloc="(chaîne lisible par l'homme)"
    adresse_de_début="(valeur de 64 bits)"
    taille_de_mot="(valeur de 64 bits)"
    longueur="(valeur de 64 bits)"
     somme de contrôle="(résumé de 20 octets)">
     (Données)
  </bloc>
</dépôt>
```

4.1 Section d'en-tête

La section d'en-tête comporte l'étiquette Dépôt *(Dump)* qui inclut les attributs suivants :

- nom : chaîne obligatoire de longueur arbitraire utilisée par toute partie intéressée pour identifier le dépôt SHF en question.
- blocs : valeur facultative de 64 bits en hexadécimal qui représente le nombre de blocs dans le dépôt SHF en question. Chaque fois qu'elle est disponible, cette valeur devrait être fournie. Cependant, il peut exister des scénarios où le nombre de blocs ne peut être donné à l'avance. Si la valeur est présente, elle devrait être vérifiée par la mise en œuvre ; si la valeur n'est pas vraie, le comportement est défini par la mise en œuvre.

Après l'étiquette de dépôt d'ouverture, une ou plusieurs sous sections de blocs doivent suivre. Finalement, le dépôt SHF complet se termine par une étiquette de clôture de dépôt.

4.2 Sous-section de bloc

La sous section de bloc contient une étiquette de bloc et un nombre de mots de données. L'étiquette de bloc comporte les attributs suivants :

- nom : chaîne obligatoire de longueur arbitraire utilisée par toute partie intéressée pour identifier le bloc spécifique.
- adresse_de_début : valeur obligatoire de 64 bits en hexadécimal représentant l'adresse de début en octets pour des données du bloc.
- taille_de_mot : valeur obligatoire de 64 bits en hexadécimal représentant le nombre d'octets (la largeur) d'un mot des données.
- longueur : représentation hexadécimale obligatoire d'un entier non signé de 64 bits indiquant le nombre de mots qui suit à l'intérieur de l'élément de bloc. Si cette valeur se révèle inexacte, le bloc DOIT être éliminé.
- somme de contrôle : représentation hexadécimale obligatoire du résumé SHA-1 de 20 octets des données du bloc.

La taille totale des données dans le bloc (en bits) est donnée par l'expression (8 * taille_de_mot * longueur). L'expression NE DOIT PAS faire plus de (2^64)-1.

Après l'étiquette d'ouverture de bloc suit une représentation hexadécimale des données réelles dans le bloc. Finalement, la

section de bloc se termine par une étiquette de clôture de bloc.

5. Règles et limites de SHF

Il y a plusieurs règles et limites dans SHF :

- Toutes les valeurs d'attribut qui représentent une valeur réelle et les données DOIVENT être en notation hexadécimale. Le seul attribut exclu de cette règle est l'attribut de nom dans les étiquettes Dépôt et Bloc. Cette restriction a été imposée pour faciliter la lecture d'un dépôt : le lecteur ne devra pas avoir d'incertitude sur le fait que la notation est ou non en hexadécimal, et peut toujours supposer qu'elle est hexadécimale.

- Toutes les valeurs d'attribut, à l'exception de la somme de contrôle, PEUVENT omettre les zéros en tête. À l'inverse, la somme de contrôle NE DOIT PAS omettre les zéros en tête.

- Les données représentées dans un bloc NE DOIVENT PAS faire plus de (2^64)-1 bits.

- La taille d'un mot NE DOIT PAS être supérieure à (2^64)-1 bits. Cela implique qu'un bloc avec un mot défini de la largeur maximum ne peut contenir plus d'un mot. Un utilisateur SHF devra s'assurer qu'il peut traiter une certaine longueur de mot avant de commencer l'analyse des blocs d'un dépôt SHF. Manquer à le faire peut causer des débordements de mémoire tampon et mettre en danger la stabilité et la sécurité du système sur lequel fonctionne l'application.

- Les valeurs d'attribut qui représentent une valeur réelle DOIVENT être en format gros boutien. Cela signifie que les chiffres hexadécimaux de plus fort poids doivent être mis à la gauche du mot, adresse, ou champ similaire, hexadécimal. Par exemple, la valeur d'adresse 1234 représente l'adresse 1234 et non 3412. Bien que certaines architectures informatiques puissent utiliser des mots petit boutiens comme format natif, il est de la responsabilité de tout producteur SHF fonctionnant sous une telle architecture de passer les valeurs d'attribut au format gros boutien. L'inverse tient pour un consommateur qui reçoit les attributs gros boutiens de SHF : si le consommateur est petit boutien, les valeurs doivent être renversées.

- De même, les mots dans un dépôt DOIVENT être mémorisés en format gros boutien si la taille de mot est supérieure à un octet. Ici, le même besoin de renverser les octets peut survenir, comme mentionné au point précédent.

6. DTD SHF

Le contenu de l'élément nommé "bloc" et les attributs "blocs", "adresse", "taille_de_mot" et "somme_de_contrôle" ne devraient contenir que les caractères qui sont des séquences d'octets hexadécimaux valides. Ce sont :

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

Un analyseur qui lit un fichier SHF devrait ignorer en silence tout autre caractère qui apparaîtrait (par erreur) dans un de ces éléments ou attributs. Ces caractères étrangers devraient être traités comme si ils n'existaient pas. Noter aussi que "whitespace" n'a pas de signification sémantique ; ce n'est valide que pour améliorer la lecture du dépôt par l'homme. Les espaces peuvent tout aussi bien être supprimées et les séquences d'octets hexadécimaux enchaînées si on le désire. On remarquera que le fait que la taille de mot soit donnée en nombre d'octets implique que le nombre de chiffres héxadecimaux dans un bloc doit être pair. Les blocs mal formés devraient être ignorés par les mises en œuvre.

```
<!-- DTD pour le format S Hexdump, au 10 octobre 2003 par Linus Walleij, Joachim Strombergson, Patrik Faltstrom 2003

On se réfère à ce DTD comme :

<!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. Exemples de SHF

Cette section contient trois exemples de SHF différents, illustrant l'usage de SHF et les attributs dans SHF.

Le premier exemple est un simple dépôt SHF avec un seul bloc de données :

```
<?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>
```

Le second exemple est un programme en code machine 6502 résidant à l'adresse mémoire 0x1000, qui calcule les 13 premiers nombres de Fibonacci et les mémorise à 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>
```

L'exemple final contient un bloc de données de 40 bits :

```
<?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. Considérations de sécurité pour SHF

Le format SHF est un format pour représenter des données hexadécimales qu'on veut transférer, gérer, ou transformer. Le format lui-même ne garantit pas que les données représentées ne sont pas faussement représentées, malveillantes, ou par ailleurs dangereuses.

L'intégrité des données du fichier SHF comme un tout est à assurer, si nécessaire, par des moyens extérieurs au fichier SHF, comme le mécanisme de signature générique décrit par la [RFC3275].

9. Considérations relatives à l'IANA

Cette section contient les informations d'enregistrement pour le type MIME en SHF. Le type de support a été choisi pour être conforme aux lignes directrices de la [RFC3023].

Enregistrement : application/shf+xml
Nom de type de support MIME : application
Nom de sous type MIME : shf+xml
Paramètres exigés : jeu de caractère. Ce paramètre doit exister et doit être réglé à "UTF-8". Aucun autre jeu de caractère n'est permis pour transporter les données SHF. La désignation de jeu de caractères DOIT être en majuscules.

Considérations de codage : ce type de support peut avoir un contenu binaire ; en conséquence, lorsque utilisé sur un transport qui ne permet pas de transfert binaire, un codage approprié doit être appliqué.

Considérations sur la sécurité : un dépôt hexadécimal ne pose pas par lui-même d'autre problème de sécurité que ceux qui s'appliquent pour tout autre fichier XML. Cependant, les données binaires incluses peuvent en forme décodée contenir du code exécutable pour une plateforme cible. Si une sécurité supplémentaire est désirée, des solutions de sécurité de transport supplémentaires peuvent être appliquées. Pour le code cible contenu dans un dépôt hexadécimal, les développeurs peuvent vouloir inclure des certificats, des sommes de contrôle, et autres en forme de dépôt hexadécimal pour la plateforme cible. Une telle utilisation sort du domaine d'application du présent document et est une affaire de mise en œuvre.

Considérations d'interopérabilité : non applicable

Spécification publiée : ce type de support est un sous ensemble de la spécification XML 1.0 [XML]. Une restriction est faite : aucune référence d'entité autre que les cinq références d'entité générales prédéfinies ("&", "<", ">", "'", et """) et les références d'entités numériques ne peut être présente. Ni la déclaration "XML" (par exemple, <?xml version="1.0" ?>) ni la déclaration "DOCTYPE" (par exemple, <!DOCTYPE ...>) n'a besoin d'être présente. (Les fragments XML sont permis.) Toutes les autres instructions XML 1.0 (par exemple, blocs CDATA, instructions de traitement, et ainsi de suite) sont permises.

Applications qui utilisent ce type de support : tout programme ou individu qui souhaite utiliser ce sous ensemble XML 1.0 pour des échanges hexdump.

Information supplémentaires :
- o Numéro magique : il n'y a pas de séquence initiale d'octets seule qui soit toujours présente pour les fichiers SHF.
- o Extension de fichier : shf
- o Code de type de fichier Macintosh : aucun

Utilisation prévue : COMMUNE.
Auteur/contrôleur des changements : ce type de transport MIME est contrôlé par l'IETF.

10. Extensions

Les attributs des éléments en format SHF XML peuvent être étendus lorsque le besoin s'en fait sentir. Par exemple, certaines applications vont vouloir représenter du code exécutable comme un dépôt SHF, et auront ensuite besoin d'une adresse de début d'exécution associée à certains blocs Dump, afin que l'adresse puisse être configurée comme point de départ pour la partie CPU de tout code de traitement présent dans le bloc, par opposition aux données brutes, qui ont déjà reçu une adresse de début par l'attribut "address". Cela peut être fait en étendant l'étiquette Block avec un attribut

"start_address".

Un autre scénario possible est lorsque un dépôt est appliqué à un système informatique avec plusieurs espaces d'adresses indépendants, comme un système avec deux CPU, ayant chacun des mémoires indépendantes. Dans ce cas, un utilisateur peut vouloir ajouter un attribut "address_space".

Tant que de tels nouveaux attributs sont ajoutés sans que des attributs soient supprimés ou redéfinis, le dépôt résultant devra être considéré comme un dépôt SHF valide et être transporté en utilisant le type de transport application/xml+shf. Les analyseurs qui ignorent l'espace de noms modifié devront ignorer en silence tout ces attributs étendus, ou simplement les dupliquer de l'entrée à la sortie lors du traitement d'un fichier SHF comme un filtre. La gestion de tels attributs étendus est affaire de convention entre les différentes classes d'utilisateurs et non l'affaire de l'IETF.

11. Informations supplémentaires

Contact pour des informations supplémentaires : voir la section "Adresse des auteurs" du présent mémoire.

Remerciements : Le dépôt de mémoire SMIL a été fourni par Sten Henriksson de Lund University. La relecture et les retours sur le document SHF ont été généreusement fournis par Peter Lindgren, Tony Hansen, Larry Masinter, et Clive D.W. Feather. Nous tenons aussi à remercier le groupe de travail de la zone Applications de son aide durant le développement.

12. Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC3023] M. Murata, S. St.Laurent et D. Kohn, "Types de supports XML", janvier 2001. (*Obsolète, voir* RFC7303)

[RFC3174] D. Eastlake 3rd et P. Jones, "Algorithme US de hachage sécurisé n° 1 (SHA1)", septembre 2001. *(Info, MàJ par* RFC4634 *et* 6234*)*

[RFC3275] D. Eastlake 3rd , J. Reagle, D. Solo, "Syntaxe et traitement de signature en langage de balisage extensible (XML)", mars 2002. *(D.S.)*

[XML] Tim Bray, Jean Paoli, C. Sperberg-McQueen, Eve Maler et François Yergeau, "Extensible Markup Language (XML) 1.0", (troisième édition) à http://www.w3.org/TR/REC-xml .

Adresse des auteurs

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

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

Patrik Faltstrom
Cisco Systems Inc
Ledasa
273 71 Lovestad
SE

mél : paf@cisco.com
URI : http://www.cisco.com

Déclaration de droits de reproduction

Propriété intellectuelle

Remerciement

Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.