Groupe de travail Réseau
Request for Comments : 4896
RFC mises à jour : 3320, 3321, 3485
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

A. Surtees, Siemens/Roke Manor Research
M. West, Siemens/Roke Manor Research
A.B. Roach, Estacado Systems
juin 2007

Corrections et précisions à la compression de signalisation (SigComp)

Résumé
Le présent document décrit des mauvaises interprétations courantes et certaines ambiguïtés dans le protocole de compression de signalisation (SigComp, *Signaling Compression Protocol*) et offre des lignes directrices aux développeurs pour résoudre les problèmes qui en résultent. SigComp définit un schéma pour la compression des messages générés par les protocoles d'application tels que le protocole d'initialisation de session (SIP, *Session Initiation Protocol*). Le présent document met à jour les RFC 3320, RFC 3321, et RFC 3485.

Table des matières

1. Introduction....2
1.1 Terminologie....2
2. Taille de mémoire de décompression....2
2.1 Codes d'octets dans la taille de mémoire de décompression....2
2.2 Taille de mémoire de décompression par défaut....2
3. Instructions UDVM....3
3.1 Instructions d'entrée des données....3
3.2 MULTILOAD....3
3.3 STATE-FREE....4
3.4 Utilisation de la pile....4
4. Règles de copie d'octet....4
4.1 Instructions qui utilisent les règles de copie d'octets....5
5. Priorité de rétention d'état....6
5.1 Valeurs de priorité....6
5.2 Priorités de rétention d'états multiples....6
5.3 Priorité de rétention 65535 (ou -1)....6
6. État dupliqué....8
7. Conflits d'identifiant d'état....8
8. Déclassement des messages....9
9. Rétroactions demandées....9
9.1 Rétroactions quand SMS est zéro....9
9.2 Mise à jour des demandes de rétroactions....9
10. Annonce des ressources....10
10.1 Bit I et éléments d'état local....10
10.2 Mise à jour dynamique de ressources....10
10.3 Annonce d'éléments d'état disponibles en local....10
11. Code d'octets non compressé....11
12. Dictionnaire statique SIP/SDP de la RFC 3485....12
13. Considérations sur la sécurité....13
14. Considérations relatives à l'IANA....13
15. Remerciements....13
16. Références....13

16.1 Références normatives....13
16.2 Références pour information....14
Appendice A. Protocole d'application factice....14
A.1 Introduction....14
A.2 Traitement d'un message DAP....14
A.3 Format de message DAP en ABNF....15
A.4 Exemple de message DAP....15
Adresse des auteurs....16
Déclaration complète de droits de reproduction....16

1. Introduction

SigComp [RFC3320] définit la machine virtuelle de décompresseur universel (UDVM, *Universal Decompressor Virtual Machine*) pour décompresser les messages envoyés par un compresseur conforme. SigComp décrit les mécanismes de traitement d'état, la structure de message, et d'autres détails. Alors que le comportement du décompresseur est spécifié en grand détail, le comportement du compresseur est laissé au choix de la mise en œuvre. Durant les essais de mise en œuvre et d'interopérabilité ont été identifiées des parties de SigComp qui appellent des précisions. Les sections qui suivent énumèrent les zones de problème identifiées dans la spécification, et tentent de fournir des éclaircissements.

Noter que, comme ce document se réfère aux paragraphes de plusieurs autres documents, on applique la notation suivante : "au paragraphe 3.4" se réfère au paragraphe 3.4 du présent document, "dans la RFC 3320-paragraphe 3.4" se réfère au paragraphe 3.4 de la [RFC3320].

1.1 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].

2. Taille de mémoire de décompression

2.1 Codes d'octets dans la taille de mémoire de décompression

SigComp [RFC3320] déclare que la taille de mémoire de décompression (DMS, *Decompression Memory Size*) par défaut est 2048 octets. La taille de mémoire UDVM est définie à la Section 7 de la RFC 3320 comme étant (DMS - n) où n est la taille du message SigComp, pour les messages transportés sur UDP et (DMS / 2) pour ceux transportés sur TCP. Cela signifie que quand le message contient le code d'octets (comme il va le faire pour au moins le premier message) il va y avoir en fait deux copies du code d'octets dans la mémoire du décompresseur (voir la Figure 1). La présence de la seconde copie du code d'octets dans la mémoire du décompresseur est correcte dans ce cas.

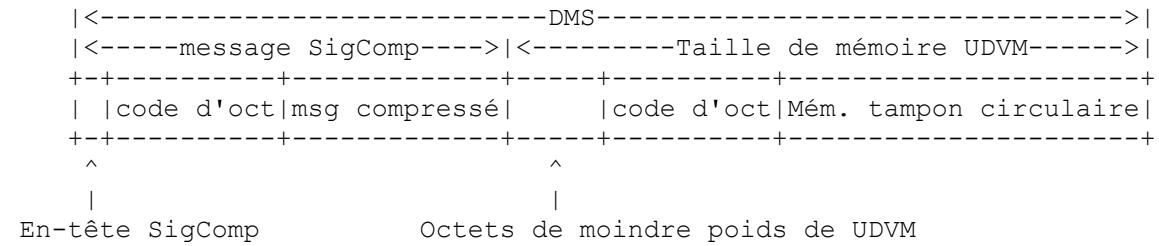


Figure 1 : Code d'octets et taille de mémoire d'UDVM dans la DMS

2.2 Taille de mémoire de décompression par défaut

Pour de nombreuses mises en œuvre, la longueur du code d'octets de décompression envoyé est dans la gamme de trois à quatre cent octets. Parce que SigComp spécifie une DMS par défaut de 2048 octets, le schéma décrit restreint sérieusement la taille de la mémoire tampon circulaire, et du message compressé lui-même. Dans certains cas, cet ensemble de circonstances a un effet dommageable sur le ratio de compression ; pour d'autres, cela rend complètement impossible d'envoyer certains messages compressés.

Pour traiter ce problème, ceux qui rendent obligatoire l'utilisation de SigComp doivent aussi fournir plus de spécification pour que leur application rende obligatoire l'utilisation d'une DMS d'une taille appropriée. Le dimentionnement d'une telle DMS devrait tenir compte (1) de la taille du code d'octets pour les algorithmes qui vont probablement être employés dans la compression des messages d'application, (2) de la taille de toutes les mémoires tampon ou structures nécessaires pour exécuter de tels algorithmes, (3) de la taille des messages d'application, et (4) de l'entropie moyenne présente dans un seul message d'application.

Par exemple, si on suppose un algorithme typique de compression exigeant approximativement 400 octets de code d'octets, plus environ 2432 octets de structures de données. La taille de mémoire UDVM requise est de 400 + 2432 = 2832. Pour un protocole fondé sur TCP, cela signifie que la DMS doit être d'au moins 5664 (2832 * 2) octets, qui est arrondi à 8 k. Pour un protocole fondé sur UDP, on doit prendre en compte la taille des messages SigComp eux-mêmes. En supposant un protocole fondé sur le texte avec une entropiée moyenne suffisante pour compresser un seul message de 50 % (sans aucun historique de message précédent) et des messages qui ne sont pas supposés excéder 8192 octets, le message de protocole lui-même va ajouter 4096 octets à la taille du message SigComp (par dessus les 400 octets de code d'octets plus un en-tête de 3 octets) ou 4096 + 400 + 3 = 4499. Pour calculer la DMS, on doit ajouter cela à la taille de mémoire d'UDVM requise : 2832 + 4499 = 6531, ce qui est là encore arrondi à 8 k de DMS.

3. Instructions UDVM

3.1 Instructions d'entrée des données

Quand on entre des données venant du message compressé, les instructions INPUT-BYTES *(entrer les octets)* (RFC 3320-paragraphe 9.4.2) et INPUT-BITS (RFC 3320-paragraphe 9.4.3) ont toutes deux le paragraphe :

> "Si l'instruction demande des données qui sont au delà de la fin du message SigComp, aucune donnée n'est retournée. L'UDVM déplace l'exécution de programme à l'adresse spécifiée par l'opérande adresse."

L'intention est que si n octets/bits sont demandés, mais que seulement m restent dans le message (où m < n) alors le répartiteur de décompression NE DOIT PAS retourner d'octets/bits à l'UDVM, et les m octets/bits qui sont là DOIVENT rester inchangés dansle message.

Par exemple, si les octets restants d'un message sont : 0x01 0x02 0x03 et si l'UDVM rencontre une instruction INPUT-BYTES (6, a, b) alors le répartiteur décompresseur ne retourne pas d'octets et saute à l'instruction spécifiée par b. Cela contient une instruction INPUT-BYTES (2, c, d) de sorte que le répartiteur décompresseur retourne avec succès les octets 0x01 et 0x02.

Dans le cas où une instruction INPUT-BYTES suit une instruction INPUT-BITS qui a laissé un octet partiel dans le message, l'octet partiel devrait quand même être éliminé même si il n'y a pas assez d'octets à entrer.

INPUT-BYTES (0, a, b) peut être utilisé pour purger un octet partiel.

3.2 MULTILOAD

Afin de rendre plus simple la mise en œuvre étape par étape, il est explicitement interdit à l'instruction MULTILOAD d'écrire dans une position de mémoire occupée par l'opcode MULTILOAD ou un de ses paramètres. De plus, si il y a des changements de paramètres, le changement DOIT être fait au moment de l'exécution.

Toute mise en œuvre technique autre qu'une mise en œuvre étape par étape (par exemple, décoder tous les opérandes puis exécuter, qui est le modèle de toutes les autres instructions) DOIT donner le même résultat qu'aurait une mise en œuvre étape par étape.

Par exemple:

à (64)

```
:location_a          pad (2)
:location_b          pad (2)
:location_c          pad (2)
```

```
pad (30)
:udvm_memory_size          pad (2)
:circular_buffer           pad (2)

align (64)

MULTILOAD (location_a, 3, circular_buffer, udvm_memory_size, $location_a)
```

La mise en œuvre étape par étape écrirait l'adresse de la mémoire tampon circulaire dans la localisation_a (adresse de mémoire 64) ; écrirait l'adresse de udvm_memory_size dans la localisation_a + 2 (adresse de mémoire 66) ; écrirait la valeur mémorisée dans la localisation_a (accédée en utilisant un accès indirect - qui est maintenant l'adresse de la mémoire tampon circulaire) dans la localisation_a + 4 (adresse de mémoire 68). Donc, à la fin de l'exécution par une mise en œuvre correcte, la localisation_c va contenir l'adresse de la mémoire tampon circulaire.

3.3 STATE-FREE

L'instruction STATE-FREE ne vérifie pas la longueur d'accès minimum *(minimum_access_length)*. Ceci est correct parce que l'état ne peut pas être libéré tant que l'application n'a pas authentifié le message. L'absence de vérification ne pose pas de risque pour la sécurité parce que si l'envoyeur a assez d'informations pour créer des messages authentifiés, alors l'envoi des messages qui sauvegardent l'état peut pousser l'état précédent hors de la mémorisation de toutes façons.

L'instruction STATE-FREE peut seulement libérer l'état dans le compartiment qui correspond au message à décompresser. Tenter de libérer l'état qui vient d'un autre compartiment, ou qui n'est associé à aucun compartiment, n'a aucun effet.

3.4 Utilisation de la pile

Les instructions PUSH, POP, CALL, et RETURN utilisent une pile qui est établie en utilisant la localisation de pile d'adresses mémoire bien connue pour définir où la pile est située dans la mémoire. L'utilisation de la pile est définie dans la RFC 3320-paragraphe 8.3, qui déclare : "Pousser" une valeur sur la pile est une abréviation pour copier la valeur à stack[stack_fill] et ensuite augmenter stack_fill de 1 ; et stack_fill est une abréviation pour le mot de deux octets à stack_location et stack_location + 1.

Dans le très rare cas où la valeur de stack_fill *(remplissage de pile)* est 0xFFFF quand une valeur est poussée sur la pile, la valeur originale de stack_fill DOIT être augmentée de 1 à 0x0000 et réécrite à stack_location *(localisation de pile)* et stack_location + 1 (ce qui va écraser la valeur qui avait été poussée sur la pile).

La nouvelle valeur poussée sur la pile a, en théorie, été écrite sur la pile [0xFFFF] = stack_location. Stack_fill va alors être augmenté de 1 ; cependant, la valeur à stack_location et stack_location + 1 a juste été mise à jour. Pour conserver l'intégrité de la pile à l'égard de sur-flux et sous-flux, stack_fill ne peut pas être relu à ce point, et la valeur poussée est écrasée.

4. Règles de copie d'octet

La RFC 3320-paragraphe 8.4 déclare que :"La chaîne d'octets est copiée en ordre ascendant d'adresse de mémoire, en respectant les limites établies par byte_copy_left *(copie d'octet à gauche)* et byte_copy_right" *(copie d'octet à droite)*. Ceci est trompeur parce qu'il est parfaitement légitime de copier des octets en dehors des limites établies par byte_copy_left et byte_copy_right. Les instruction byte_copy_left et byte_copy_right fournissent la capacité de maintenir une mémoire tampon circulaire comme suit :

Pour un mouvement à droite :
si current_byte == ((byte_copy_right - 1) mod 2 ^ 16) : next_byte = byte_copy_left
autrement : next_byte = (current_byte + 1) mod 2 ^ 16

ce qui est équivalent à l'algorithme donné dans la RFC 3320-paragraphe 8.4.

Pour un mouvement à gauche :
si current_byte == byte_copy_left : previous_byte = (byte_copy_right - 1) mod 2 ^ 16
autrement : previous_byte = (current_byte - 1) mod 2 ^ 16

Le mouvement à gauche est seulement utilisé pour COPY_OFFSET.

Par conséquent, la copie pourrait commencer à la gauche de byte_copy_left et continuer à travers lui (et sauter en arrière jusqu'à lui selon l'algorithme concerné si nécessaire) et pourrait commencer à byte_copy_right ou à sa droite (bien qu'il faille faire attention à empêcher un échec de décompression dû à l'écriture/lecture au delà de la mémoire d'UDVM).

Pour être plus clair : considérons la mémoire de l'UDVM disposée comme suit, avec byte_copy_left et byte_copy_right dans les localisations indiquée respectivement par "BCL" et "BCR" :

```
+---------------------------------------+
|                                       |
+----------^------------^-----------------+
           BCL          BCR
```

Si un opcode lit ou écrit des octets en commençant à la gauche de byte_copy_left, il va le faire dans l'ordre suivant :

```
+---------------------------------------+
|        abcdefghijkl                   |
+----------^------------^-----------------+
           BCL          BCR
```

Si le opcode continue de lire ou écrire jusqu'à attteindre byte_copy_right, il va alors revenir à byte_copy_left et continuer (les lettres après le retour sont en majuscules pour que ce soit clair) :

```
+---------------------------------------+
|        abcQRSTUVjklmnop               |
+----------^------------^-----------------+
           BCL          BCR
```

De même, écrire à la droite de byte_copy_right est une opération parfaitement valide pour les opcodes qui respectent les règles de copie d'octets :

```
+---------------------------------------+
|                           abcdefg     |
+----------^------------^-----------------+
           BCL          BCR
```

Une règle finale, relique un peu étrange de celles en cours se produit quand byte_copy_right est en fait moins que byte_copy_left. Dans ce cas, les lectures et écritures vont sauter la mémoire entre les pointeurs :

```
+---------------------------------------+
|     abcde              fghijkl        |
+----------^------------^-----------------+
           BCR          BCL
```

4.1 Instructions qui utilisent les règles de copie d'octets

Le présent document amende la liste des instructions qui obéissent aux règles de copie des octets dans la RFC 3320-paragraphe 8.4 pour inclure STATE-CREATE et CRC.

La RFC 3320-paragraphe 8.4 spécifie les règles de copie d'octets et inclut une liste d'instructions qui leur obéissent. STATE-CREATE n'est pas dans cette liste mais END-MESSAGE y est. Cela a causé de la confusion à cause du fait qu'aucune instruction ne fait en réalité de copie d'octets ; les deux instructions donnent plutôt des informations au traitement d'état pour créer l'état. Logiquement, les deux instructions devraient avoir les mêmes informations sur la copie d'octets.

Quand l'état est créé par le traitement d'état (que ce soit à partir d'une instruction END-MESSAGE ou STATE-CREATE) les règles de copie d'octet de la RFC 3320-paragraphe 8.4 s'appliquent.

Noter que, si le contenu de l'UDVM change entre l'occurrence de l'instruction STATE-CREATE et de l'état créé, les octets

qui sont mémorisés sont ceux de la mémoire tampon au moment de la création (c'est-à-dire, quand le message a été décompressé et authentifié).

Le CRC n'est pas mentionné dans la RFC 3320-paragraphe 8.4 dans la liste des instructions qui obéissent aux règles de copie d'octets, mais sa description au paragraphe 9.3.5 de la RFC 3320 déclare que ces règles doivent être respectées. Quand on lit les données sur lesquelles effectuer la vérification de CRC, les règles de copie d'octets s'appliquent comme spécifié dans la RFC 3320-paragraphe 8.4.

Quand l'identifiant partiel pour une instruction STATE-FREE est lu, (durant l'exécution de END-MESSAGE) les règles de copie d'octets de la RFC 3320-paragraphe 8.4 s'appliquent.

Étant donné que la lecture de la mémoire tampon pour créer et libérer l'état au sein de l'instruction END-MESSAGE obéit aux règles de copie d'octets, il peut y avoir une certaine confusion quant à savoir si la lecture des éléments de rétroaction devrait aussi obéir aux règles de copie d'octets. Les règles de copie d'octets ne s'appliquent pas à la lecture des éléments de rétroaction.

5. Priorité de rétention d'état

5.1 Valeurs de priorité

Pour la priorité de rétention d'état, 65535 < 0 < 1 < ... < 65534. Ceci est un peu contre intuitif mais est correct.

5.2 Priorités de rétention d'états multiples

Il peut y avoir une certaine confusion quand le même état est crée à deux priorités de rétention différentes. On précise cela ci-après :

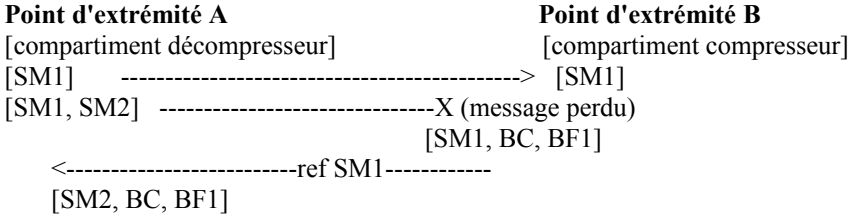
La priorité de rétention DOIT être associée au compartiment et non à l'élément d'état. Par exemple, si le point d'extrémité A crée un élément d'état avec la priorité de rétention 1 et si le point d'extrémité B crée exactement le même état avec la priorité de rétention 2, il devrait y avoir une copie (en supposant le modèle de gestion d'état suggéré dans SigComp [RFC3320]) de l'état actuel, mais chaque compartiment devrait garder un enregistrement de cet élément d'état ave sa propre priorité. (Si cela ne se produit pas, alors l'état ne pourra pas être conservé plus longtemps que ce que A a prévu ou moins que B n'a anticipé, selon quelle priorité est utilisée. Cela pourrait causer un échec de décompression.)

Si le même élément d'état est créé au sein d'un compartiment avec une priorité différente, alors une copie de l'état devrait être mémorisée avec la nouvelle priorité et il DOIT compter seulement pour une fois par rapport à la taille de mémoire d'état (SMS, S*tate Memory Size*). C'est-à-dire, la création d'état met à jour la priorité plutôt que de créer un nouvel élement d'état.

5.3 Priorité de rétention 65535 (ou -1)

Il y a un problème potentiel avec la mémorisation de plusieurs éléments d'état sous la priorité de rétention minimum (65535) telle que définie dans SigComp [RFC3320]. Cela peut être montré en considérant les exemples suivants qui sont de mode partagé, documenté dans SigComp étendu [RFC3321]. L'élément clé au sujet de l'état pour la priorité de rétention 65535 est qu'elle peut être créée par un point d'extrémité dans le compartiment décompresseur à l'insu du compresseur distant (qui contrôle la création d'état dans le compartiment décompresseur).

Exemple 1 :
[L'état SMn est l'état en mode partagé (priorité 65535),
BC est l'état de code d'octets (priorité 1),
BFn est l'état de mémoire tampon (priorité 0)]



```
                                                  Le point d'extrémité B croit encore que SM1 est au point d'extrémité A
                                                  [BC, BF1, BF2]
    <--------------------------ref SM1-----------------------
Échec de décompression en A parce que SM1 a déjà été supprimé

Exemple 2 :
Point d'extrémité A                          Point d'extrémité B
[compartiment décompresseur]                 [compartiment compresseur]
    [SM1]
    ---------------------------------------------------------->
                    [SM1]

                    [SM1, BC, BF1]
    (message perdu)X------ref SM1-----

    [SM1, SM2]
    ---------------------------------------------------------->
                                  Le point d'extrémité B ne crée pas SM2 parce qu'il n'y a pas d'espace
                                 [SM1, BC, BF1]

                    [SM1, BC, BF1, BF2]
    <---------------------------------ref SM1----------------------
    [SM2, BC, BF2]
                                       Le point d'extrémité B croit encore que SM1 est au point d'extrémité A
                                         [BC, BF1, BF2, BF3]
    <--------------------------ref SM1--------------------------------

Échec de décompression en A parce que SM1 a déjà été supprimé
```

Figure 2 : Exemples de priorité de rétention 65535

Une fois qu'il y a plus d'un élément d'état de priorité minimum créé dans un compartiment de décompresseur, le compresseur correspondant ne peut pas être certain de quels éléments d'état sont présents dans ce compartiment (décompresseur). Si il y a seulement un élément d'état, une telle ambiguïté n'existe pas.

Le problème est la conséquence des différentes règles pour la création de l'état de priorité minimum. En particulier, la création du second élément d'état à l'insu du compresseur pourrait signifier que le premier élément est poussé dehors plus tôt que ne l'attend le compresseur (en dépit du fait que les règles de traitement d'état de SigComp [RFC3320] ne sont pas mises en œuvre correctement).

SigComp [RFC3320] déclare aussi qu'un compresseur DOIT être certain que toutes les données nécessaires pour décompresser un message SigComp sont disponibles au point d'extrémité receveur. Donc, il NE DEVRAIT PAS faire référence à un état tant qu'il ne peut être sûr que l'état existe. Le fait que le compresseur en B n'ait pas de moyen de savoir combien d'état a été créé en A peut conduire à une perte de synchronisation entre les points d'extrémité, ce qui n'est pas acceptable.

On peut observer qu'il est toujours sûr de faire référence à un élément d'état de priorité minimum à la suite de la réception de l'annonce de l'état.

Si il est connu que les deux points d'extrtémité utilisent SigComp version 2, comme défini dans le NACK [RFC4077], alors un point d'extrémité PEUT supposer que la probabilité d'une perte de synchronisation est très faible, et s'appuyer sur le mécanisme de NACK pour la récupération.

Cependant, pour qu'un compresseur essaye et évite de causer la génération de NACK, il doit être capable de faire des hypothèses sur le comportement du compresseur homologue. Aussi, si un des points d'extrémité ne prend pas en charge le NACK, une autre solution est nécessaire.

Par conséquent, lorsque le NACK n'est pas pris en charge, ou pour les compresseurs qui éprouvent une certaine aversion pour le NACK, la recommandation est qu'un seul élément d'état de priorité minimum DEVRAIT être présent dans un compartiment à tout moment. Si les deux points d'extrémité prennent en charge le NACK [RFC4077], alors cette recommandation PEUT être assouplie, mais les mises en œuvre doivent soupeser avec soin les conséquences de la création

de plusieurs éléments d'état de priorité minimum. Dans l'un et l'autre cas, si le comportement de l'application restreint le flux de messages, ce fait pourrait être exploité pour permettre la création sûre de plusieurs états de priorité minimum ; cependant, il faut quand même faire attention.

Noter que si un compresseur souhaite que le point d'extrémité distant soit capable de créer un nouvel élément d'état de priorité minimum, il peut utiliser l'instruction STATE-FREE pour supprimer l'élément d'état existant.

6. État dupliqué

Si un élément d'état est créé dans un compartiment dans lequel il existe déjà, l'heure de sa création DEVRAIT être mise à jour comme si il venait d'être créé, sans considération de si il y a ou non une nouvelle priorité de rétention d'état.

7. Conflits d'identifiant d'état

La RFC 3320-paragraphe 6.2 déclare que quand un élément d'état est créé, le hachage complet de 20 octets devrait être vérifié pour voir si il existe ou non un autre élément d'état avec cet identifiant. Si il en existe un, et si l'élément d'état n'est pas identique, alors la nouvelle création DOIT échouer. Il est déclaré que la probabilité que cela se produise est extrêment faible (et il en est ainsi, voir ci-dessous).

Cependant, quand on accède à l'état, seuls les n premiers octets de l'identifiant d'état sont utilisés, où n pourrait être de pas moins de 6. À ce point, si il y a deux éléments d'état avec les mêmes n premiers octets d'identifiant d'état, l'instruction STATE-ACCESS va causer l'échec de la décompression. Le compresseur qui fait référence à l'état ne va pas s'atttendre à ce mode d'échec parce que la création d'état a réussi sans encombre. À un point d'extrémité de serveur où il pourrait y avoir des milliers ou des millions d'éléments d'état, comment cela a t-il des chances de se produire ?

Considérons le paradoxe de l'anniversaire (où il faut seulement 23 personnes dans une pièce pour avoir plus de 50 % de chances que deux d'entre elles aient la même date d'anniversaire [Birth])).

La calcul simple utilisant les factoriels donne :

$$Pd(N,s) = 1 - \frac{N!}{(N - s)!\ N^s}$$

où N est le nombre de valeurs possibles et s est la taille de l'échantillon.

Cependant, comme on traite des grands nombres, une approximation est nécessaire :

$$Pd(N,s) = 1 - e^{(LnFact(N) - LnFact(N-s) - s\ Ln(N))}$$

où LnFact (x) est le logarithme de x!, qui peut être approximé par :

$$LnFact(x) \sim (x + 1/2)\ Ln(x) - x + Ln(2*Pi)/2 + \frac{1}{12x} - \frac{1}{360\ x^3} + \frac{1}{1260\ x^5} - \frac{1}{1680\ x^7}$$

qui, en utilisant N = 2^48 [identifiant d'état partiel de 6 octets] donne :
s = 1 000 000 : Pd (N,s) = 0,018 %
s = 10 000 000 : Pd (N,s) = 16,28 %
s = 100 000 000 : Pd (N,s) = 100,00 %

de sorte que lors de la mise en œuvre, il faut se demander si 6 octets d'identifiant d'octet est assez pour assurer que l'accès à l'état va réussir (en particulier à un serveur).

La probabilité d'un conflit quand on utilise les 20 octets d'identifiant d'état, n'a donc pas une probabilité extrêment faible : utiliser N = 2^160 [l'identifiant d'état complet de 20 octets] donne :
s = 1 000 000 : Pd (N,s) = 3,42E-35 %
s = 10 000 000 : Pd (N,s) = 3,42E-33 %

s = 100 000 000 : Pd (N,s) = 3,42E-31 %

Par conséquent, il faut faire attention quand on décide de combien d'octets d'identifiant d'état utiliser pour accéder à l'état au serveur.

8. Déclassement des messages

SigComp [RFC3320] fait seulement une référence à la possibilité de messages dans le désordre. Cependant, la déclaration que "le compresseur DOIT s'assurer que le message peut être décompressé en utilisant les ressources disponibles au point d'extrémité distant" met à la charge du compresseur la prise en compte de la possibilité de production d'un désordre.

Il dépend de la définition du compartiment et du protocole de transport utilisé qu'un désordre puisse se produire et qu'il ait un impact. Donc, il appartient à la mise en œuvre de compresseur de prendre ces facteurs en compte.

9. Rétroactions demandées

9.1 Rétroactions quand la SMS est zéro

Si un point d'extrémité reçoit une demande de rétroaction, il DEVRAIT alors retourner la rétroaction même si sa SMS est zéro. Les frais généraux de mémorisation de la rétroaction demandée NE font PAS partie de la SMS.

9.2 Mise à jour des demandes de rétroactions

Quand un point d'extrémité reçoit un message valide, il met à jour les données de rétroaction demandées pour ce compartiment. La RFC 3320-Section 5 déclare qu'il n'est pas besoin de transmettre plus d'une fois un élément de rétroaction demandé. Cependant, il y a des cas où il va être avantageux que la rétroaction soit envoyée plus d'une fois (par exemple, un message SIP 200 OK retransmis [RFC3261] à un message SIP INVITE implique que le 200 OK original, et la rétroaction qu'il porte, pourraient n'avoir pas atteint le point d'extrémité distant). Donc, un point d'extrémité DEVRAIT transmettre de façon répétée les rétroactions jusqu'à ce qu'il reçoive un autre message valide qui met à jour la rétroaction.

La RFC 3320-paragraphe 9.4.9 déclare que quand requested_feedback_location *(localisation de rétroaction demandée)* égale zéro, aucune demande de rétroaction n'est faite. Cependant, il n'est pas indiqué si cela signifie que les données de rétroaction existantes restent intouchées ou si cela signifie que les données de rétroaction existantes DEVRAIENT être écrasées pour qu'il n'y ait plus de données de rétroaction. Si requested_feedback_location égale zéro, les données de rétroaction existantes DEVRAIENT être laissées intouchées et retournées dans tous les messages suivants comme auparavant.

La RFC 3320-paragraphe 9.4.9 ne fait aussi aucune déclaration sur ce qui arrive aux données de rétroaction existantes quand requested_feedback_location n'est pas égal à zéro mais que le fanion Q qui indique la présence/absence d'un requested_feedback_item *(élément_de_rétroaction_demandé)* est à zéro. Dans ce cas, les données de rétroaction existantes DEVRAIENT être écrasées pour être "pas de données de rétroaction".

10. Annonce des ressources

10.1 Bit I et éléments d'état local

Le bit I dans la rétroaction demandée est un mécanisme par lequel un compresseur peut dire à un point d'extrémité distant qu'il ne va accéder à aucun élément d'état local. Ce faisant, il donne au point d'extrémité distant l'option de ne pas les annoncer dans les messages suivants. Établir le bit I n'oblige pas le point d'extrémité distant à cesser d'envoyer des annonces.

Le point d'extrémité distant DEVRAIT quand même annoncer ses paramètres DMS et SMS. (Ceci est particulièrement important ; si l'envoyeur du premier message établit le bit I, il va quand même vouloir l'annonce des paramètres par le receveur. Si il ne les reçoit pas, il doit supposer les paramètres par défaut, ce qui va affecter l'efficacité de compression.)

Le point d'extrémité qui reçoit un bit I de 1 peut réclamer la mémoire utilisée pour mémoriser les éléments d'état disponibles localement. Cependant, cela N'A PAS d'impact sur un état créé par l'envoyeur en utilisant les instructions

END-MESSAGE ou STATE-CREATE.

10.2 Mise à jour dynamique de ressources

Les ressources de décompresseur comme la SMS et la DMS peuvent être mises à jour dynamiquement au compresseur par l'utilisation des bits SMS et DMS dans les paramètres de rétroaction retournés (voir le paragraphe 9.4.9 de la RFC 3320). Changer les ressources dynamiquement (à part les annonces initiales pour chaque compartiment) n'est pas supposé arriver très souvent.

Si des ressources supplémentaires sont annoncées à un compresseur, il appartient alors à la mise en œuvre de compresseur de décider d'utiliser ou non ces ressources. Par exemple, si le décompresseur annonce une SMS de 8 k mais que le compresseur a seulement 4 k de SMS, alors le compresseur PEUT choisir de ne pas utiliser les 4 k supplémentaires (par exemple, afin de surveiller l'état économisé au décompresseur). Dans ce cas, il n'y a pas de problème de synchronisation. Le compresseur NE DOIT PAS utiliser plus que les dernières ressources annoncées. Noter que la SMS du compresseur n'est pas officielle (elle permet au compresseur de surveiller l'état du décompresseur) et elle est séparée de la SMS annoncée par le décompresseur.

Reduire les ressources pose des problèmes potentiels de synchronisation et donc NE DEVRAIT PAS être fait sans nécessité absolue. Si c'est le cas, alors la mémoire NE DOIT PAS être reprise avant que le point d'extrémité distant ait accusé réception du message envoyé avec l'annonce. Si l'état doit être supprimé pour s'accommoder d'une réduction de la SMS, alors les deux points d'extrémité DOIVENT le supprimer en accord avec la priorité de rétention d'état (voir la RFC 3320-paragraphe 6.2). Le compresseur NE DOIT alors PAS utiliser plus que la quantité de ressources annoncée en dernier.

10.3 Annonce d'éléments d'état disponibles en local

La RFC 3320-paragraphe 3.3.3 définit les éléments d'état disponibles en local comme étant des éléments d'état qui sont disponibles à un point d'extrémité mais qui n'ont pas été chargés par le message SigComp. Les exemples donnés sont les dictionnaires et les éléments de code d'octets bien connus ; et le mécanisme d'annonce discuté au paragraphe 9.4.9 de la RFC 3320 donne un moyen pour que le point d'extrémité annonce les éléments d'état localement disponible qu'il a.

Cependant, SigComp [RFC3320] ne définit pas pleinement (et n'a jamais été destiné à le faire) l'utilisation des éléments d'état localement disponibles, en particulier, la durée pendant laquelle ils vont être disponibles. La définition de l'utilisation d'éléments d'état localement disponibles est laissée à d'autres documents. Cependant, ce fait, couplé au fait que SigComp ne contient pas de d'accroches pour les utilisations des éléments d'état localement disponibles et le fait que certaines des définitions de telles utilisations (dans SigComp étendu [RFC3321]) sont incomplètes a causé une certaine confusion. Donc, cette section précise la situation.

Noter qu'aucune définition des utilisations des éléments d'état localement disponibles NE DOIT être en conflit avec d'autres utilisations.

10.3.1 SigComp de base

SigComp fournit un mécanisme pour qu'un point d'extrémité annonce l'état localement disponible (RFC 3320-paragraphe 9.4.9). Si le point d'extrémité qui reçoit l'annonce ne la "reconnaît pas" et donc connaît les propriétés de l'état, par exemple, sa longueur et sa durée de vie, le compresseur doit examiner très attentivement si il accède ou non à l'état ; en particulier si le NACK [RFC4077] n'est pas disponible.

SigComp fournit les accroches suivantes à utiliser en conjonction avec les éléments d'état localement disponibles. Sans autre définition, l'état localement disponible NE DEVRAIT PAS être utilisé.

La RFC 3320-paragraphe 6.2 permet la possibilité de transposer les éléments d'état localement disponibles en un compartiment et déclare que, si cela est fait, les éléments d'état DOIVENT avoir la priorité de rétention d'état de 65535 afin de ne pas interférer avec l'état créé à la demande du compresseur distant. Noter que le paragraphe 5.3 recommande aussi qu'un seul élément d'état de cette sorte DEVRAIT être créé par compartiment.

Le bit I dans la localisation de rétroaction demandée (voir la RFC 3320-paragraphe 9.4.9) permet à un compresseur d'indiquer au point d'extrémité distant qu'il ne va faire référence à aucun des états précédemment anoncés en local. Selon le modèle de mise en œuvre pour le traitemenent d'état au point d'extrémité distant, cela pourrait permettre au point d'extrémité distant de réclamer la mémoire utilisée par ces éléments d'état.

10.3.2 Dictionnaires

L'utilisation la plus basique des annonces d'état local est l'annonce d'un dictionnaire (par exemple, le dictionnaire spécifié par le dictionnaire statique SIP/SDP [RFC3485]) ou un élément de code d'octets. En géneral, ces éléments d'état :

- o ne sont pas transposés en compartiments,
- o sont locaux pour le point d'extrémité,
- o sont disponibles pour au moins la durée du compartiment,
- o n'ont pas d'impact sur la SMS du compartiment.

Cependant, pour un élément d'état donné, la durée de vie exacte doit être définie, par exemple, dans des spécifications publiques telles que SigComp pour SIP [RFC5049] ou la spécification 3GPP IMS [Multi]. Une telle spécification devrait aussi indiquer si l'annonce de l'état est ou non nécessaire.

10.3.3 Mécanismes d'extension de SigComp

SigComp étendu [RFC3321] définit des utilisations d'annonces d'état local pour lesquelles des éclaircissements supplémentaires sont donnés ici.

Le mode partagé (RFC 3321-paragraphe 5.2) est bien défini (quand il est combiné avec la précision du paragraphe 5.3). En particulier, les états qui sont créés et annoncés sont transposés en le compartiment, ont la priorité de rétention minimum et ne persistent que jusqu'à ce qu'ils soient supprimés par la création d'un nouvel état (non de priorité de rétention minimum) ou l'utilisation d'une instruction STATE-FREE.

La définition des accusés de réception initiés par le point d'extrémité (RFC 3321-paragraphe 5.1.2) exige des précisions afin de s'assurer que la définition n'empêche pas d'utiliser les annonces pour indiquer que l'état va être conservé au delà de la durée de vie du compartiment (comme discuté dans SigComp pour SIP [RFC5049]). Donc, la précision est que :

Lorsque le point d'extrémité A demande la création d'état au point d'extrémité B, le point d'extrémité B PEUT ensuite annoncer le hachage de l'élément d'état créé au point d'extrémité A. Cela porte au point d'extrémité A (i) que l'état a bien été créé dansle compartiment; et (ii) que l'état va être disponible pour au moins la durée de vie de l'état, comme défini par les règles de suppression d'état en accord avec l'âge et la priorité de rétention de SigComp [RFC3320]. Si l'état est disponible au point d'extrémité B après qu'il aurait dû être supprimé du compartiment selon la [RFC3320], alors l'état ne compte plus dans la SMS du compartiment. Comme il n'est pas garanti qu'un tel état soit disponible au delà de sa durée de vie normalement définie, les points d'extrémité DEVRAIENT seulement tenter d'accéder à l'état après le moment où il est connu que le NACK [RFC4077] est disponible.

11. Code d'octets non compressé

Il est possible d'écrire un code d'octets qui donne simplement pour instruction au décompresseur de sortir le message entier (l'envoyant effectivement non ncompressé, mais dans un message SigComp). Ceci est particulièrement utile si le code d'octets est bien connu (de sorte que les décompresseurs peuvent reconnaître et sortir les octets sans faire fonctionner une machine virtuelle si ils le souhaitent) ; donc, il est donné ici.

Le code mnémonique est :

```
at (0)
:udvm_memory_size        pad (2)
:cycles_per_bit          pad (2)
:sigcomp_version         pad (2)
:partial_state_id_length  pad (2)
:state_length            pad (2)
:reserved                pad (2)
at (64)
:byte_copy_left          pad (2)
:byte_copy_right         pad (2)
:input_bit_order         pad (2)
:stack_location          pad (2)
```

```
; Simple boucle
;     Lire un octet
;     Sortir un octet
; Jusqu'à ce qu'il n'y ait plus d'octet !

at (128)
:start
INPUT-BYTES (1, byte_copy_left, end)
OUTPUT (byte_copy_left, 1)
JUMP (start)

:end
END-MESSAGE (0,0,0,0,0,0,0)
```

qui se traduit en le message SigComp suivant :

0xf8, 0x00, 0xa1, 0x1c, 0x01, 0x86, 0x09, 0x22, 0x86, 0x01, 0x16, 0xf9, 0x23

12. Dictionnaire statique SIP/SDP de la RFC 3485

Le dictionnaire statique SIP/SDP [RFC3485] fournit un dictionnaire de chaînes fréquemment utilisées dans les messages SIP et SDP. Le format du dictionnaire est la liste des chaînes suivie par un tableau des références de décalage aux chaînes afin qu'un compresseur puisse choisir de faire référence à l'adresse de la chaîne ou à l'entrée dans le tableau. Les deux parties du dictionnaire sont divisées en 5 sections munies de priorités pour permettre aux compresseurs de choisir quelle quantité il vont en utiliser (ce qui est particulièrement utile dans le cas où il faut le télécharger). Si seulement une partie du dictionnaire est utilisée, alors les sections correspondantes des deux parties (chaînes et tableau des décalages) sont utilisées.

Cependant, il y a quelques erreurs mineures dans le dictionnaire. À un certain nombre d'endroits, l'entrée dans le tableau des décalages se réfère à une adresse qui n'est pas dans la section correspondante de priorité dans la liste des chaînes. Par conséquent, si le code d'octets utilise le tableau des décalages et limite l'utilisation du dictionnaire aux priorités inférieures à 4, il faut faire attention à ne pas utiliser les chaînes suivantes dans le dictionnaire :

'application' à 0x0334 n'est pas à la priorité 2 (c'est la priorité 4)
'sdp' à 0x064b n'est pas à la priorité 2 (c'est la priorité 4)
'send' à 0x089d n'est pas à la priorité 2 (c'est la priorité 3)
'recv' à 0x0553 n'est pas à la priorité 2 (c'est la priorité 4)
'phone' à 0x00f2 n'est pas à la priorité 3 (c'est la priorité 4)

Le présent document ne corrige pas le dictionnaire, car tout changement au dictionnaire lui-même ne serait pas rétro compatible, et exigerait que toutes les mises en œuvre tiennent deux copies différentes du dictionnaire. Un tel coût est beaucoup trop élevé pour une erreur qu'il est trivial de contourner et a un effet négligeable sur les ratios de compression. L'erreur est plutôt mentionnée pour permettre aux mises en œuvre d'éviter les problémes qui en résulteraient. Précisément, si le code d'octets envoyé à un point d'extrémité distant contient des instructions qui chargent seulement une sous portion du dictionnaire SIP/SDP, alors le flux d'entrées fourni à ce code d'octets ne peut pas faire référence à un de ces cinq décalages dans le tableau des décalages, sauf si la portion de chaîne correspondante du dictionnaire a aussi été chargée. Par exemple, si le code d'octets charge seulement les trois premières priorités du dictionnaire (de chaîne et de décalage), l'utilisation du décalage pour "send" (à 0x089d) va être valide; cependant, l'utilisation du décalage pour "phone" (à 0x00f2) ne le sera pas.

13. Considérations sur la sécurité

Le présent document met à jour SigComp [RFC3320], SigComp étendu [RFC3321], et le dictionnaire statique SigComp [RFC3485]. Les considérations sur la sécurité pour les [RFC3321] et [RFC3485] sont les mêmes que pour la [RFC3320] ; donc, cette section discute seulement de comment les considérations sur la sécurité pour la [RFC3320] sont affectées par les mises à jour.

Plusieurs risques pour la sécurité sont discutés dans la [RFC3320]. Il sont brièvement évoqués ici ; cependant, cette mise à jour ne change pas les considérations sur la sécurité de SigComp :

Espionner l'état d'autres utilisateurs - ceci est atténué en utilisant au moins 48 bits de hachage. Cette mise à jour ne réduit pas ce minimum et recommande d'utiliser plus de bits dans certaines circonstances.

Falsification d'état, ou changement, non autorisé - ceci est atténué par le fait que la couche application doit autoriser les manipulations d'état. Cette mise à jour ne change pas ce mécanisme.

Utilisation de SigComp comme outil d'attaque de déni de Ssrvice (DoS) - ceci est atténué par le fait que SigComp génère seulement un message décompressé par message compressé entrant. Ceci n'est pas changé par cette mise à jour.

Attaque de SigComp comme cible de DoS en le remplissant d'états - ceci est atténué par le fait que la couche d'application doit autoriser les manipulations d'état. Cette mise à jour ne change pas ce mécanisme.

Attaque de l'UDVM par l'envoi d'un code en boucle - ceci est atténué par la limite supérieure de "cycles UDVM", qui n'est pas changée par cette mise à jour.

14. Considérations relatives à l'IANA

Le présent document met à jour SigComp [RFC3320], mais n'en change pas la version. Par conséquent, les considérations relatives à l'IANA sont les mêmes que pour la [RFC3320].

Le présent document met à jour SigComp étendu [RFC3321], mais n'en change pas la version. Par conséquent, les considérations relatives à l'IANA sont les mêmes que pour la [RFC3321].

Le présent document met à jour le dictionnaire statique [RFC3485], mais n'en change pas la version. Par conséquent, les considérations relatives à l'IANA sont les mêmes que pour la [RFC3485].

15. Remerciements

Nons tenons à remercier les personnes suivantes qui, en plus d'être des auteurs ou développeurs de SigComp, nous ont fourni des suggestions et commentaires : Richard Price, Lajos Zaccomer, Timo Forsman, Tor-Erik Malen, Jan Christoffersson, Kwang Mien Chan, William Kembery, et Pekka Pessi.

16. Références

16.1 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)

[RFC3320] R. Price, et autres, "Compression de signalisation (SigComp)", janvier 2003. (*MàJ par* RFC4896) *(P.S.)*

[RFC3321] H. Hannu et autres, "Compression de signalisation (SigComp) - Opérations d'extension", janvier 2003. (*MàJ par* RFC4896) *(P.S.)*

[RFC3485] M. Garcia-Martin et autres,"Dictionnaire statique du protocole d'initialisation de session (SIP) et du protocole de description de session (SDP) pour la compression de signaux (SigComp)", février 2003. (*MàJ par* RFC4896) *(P.S.)*

[RFC4077] A.B. Roach, "Mécanisme d'accusé de réception négatif pour la compression de signalisation", mai 2005. *(P.S.)*

16.2 Références pour information

[Birth] Ritter, T., "Estimating Population from Repetitions in Accumulated Random Samples", 1994, <http://www.ciphersbyritter.com/ARTS/BIRTHDAY.HTM>.

[Multi] "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP)", octobre 2006.

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

[RFC3261] J. Rosenberg et autres, "SIP : Protocole d'initialisation de session", juin 2002. (*Mise à jour par* 3265, 3853, 4320, 4916, 5393, 6665, 8217, 8760)

[RFC5049] C. Bormann et autres, "Application de la compression de signalisation (SigComp) au protocole d'initialisation de session (SIP)", décembre 2007. (*MàJ* RFC3486) *(P.S.)*

Appendice A. Protocole d'application factice

A.1 Introduction

Cet appendice définit un simple protocole d'application factice (DAP, *dummy application protocol*) qui peut être utilisé pour des essais d'interopérabilité SigComp. C'est utile pour les mises en œuvre de SigComp qui ne sont pas intégrées dans une pile SIP. Il fournit aussi des caractéristiques qui facilitent les essais des opérations internes de SigComp.

Le format de message est assez simple. Chaque message consiste en un en-tête de message de 8 lignes, une ligne vide, et un corps de message FACULTATIF. Le style ressemble à celui de SIP et HTTP.

Le format exact du message est donné plus loin en format Backus-Naur augmenté (ABNF) [RFC2234]. Voici quelques notes :
Chaque ligne d'en-tête de message DOIT être terminée par un CRLF.
La ligne vide DOIT être présente même si il n'y a pas de corps de message.
La longueur de corps est la longueur du corps du message, à l'exclusion du CRLF qui sépare le corps de message de l'en-tête de message.
Toutes les chaînes dans l'en-tête de message sont insensibles à la casse.
Pour la mise en œuvre selon cet appendice, la version de DAP DOIT être réglée à 1.

A.2 Traitement d'un message DAP

Un message avec un format invalide va être éliminé au DAP receveur. Pour les besoins d'essais, un message avec un format valide va être retourné à l'envoyeur d'origine (adresse IP, numéro d'accès) en clair, c'est-à-dire, sans compression. C'est le cas même si l'envoyeur demande à ce receveur de rejeter le message. Noter que le message DAP entier (en-tête de message + CRLF + corps de message) est retourné. Cela permet à l'envoyeur de comparer ce qu'il envoie avec ce que le receveur a décompressé.

L'identifiant de point d'extrémité est l'identifiant mondial du point d'extrémité envoyeur. Il peut être utilisé pour vérifier le cas où plusieurs points d'extrémité SigComp communiquent avec le même point d'extrémité SigComp distant. Pour simplifier, l'adresse IPv4 est utilisée à cette fin.

Identifiant de compartiment est l'identifiant du compartiment *compresseur* que le point d'extrémité *envoyeur* a utilisé pour compresser ce message. Il est alloué par l'envoyeur et donc n'est unique que par point d'extrémité envoyeur ; c'est-à-dire que les messages DAP envoyés par des points d'extrémité différents PEUVENT porter le même identifiant de compartiment. Donc, le receveur DEVRAIT utiliser la paire (identifiant de point d'extrémité, identifiant de compartiment) portée dans un message pour déterminer l'identifiant de compartiment décompresseur pour ce message. La représentation locale exacte de l'identifiant de compartiment déduit est un choix de mise en œuvre.

Pour vérifier les rétroactions SigComp [RFC3320], les compartiments homologues entre deux points d'extrémité sont définis dans DAP comme ceux qui ont le même identifiant de compartiment. Par exemple, (point d'extrémité-A, 1) et (point d'extrémité-B, 1) sont des compartiments homologues. Cela signifie que la rétroaction SigComp pour un message DAP envoyé du compartiment 1 du point d'extrémité-A au point d'extrémité-B va être portée sur un message DAP envoyé du compartiment 1 du point d'extrémité-B au point d'extrémité-A.

Un receveur DAP va suivre l'instruction portée dans l'en-tête de message ligne-5 pour accepter ou rejeter un message DAP. Note : ligne-6 et ligne-7 vont être ignorées si le message est rejeté.

Un receveur DAP va suivre l'instruction de ligne-6 pour créer ou clore le compartiment décompresseur qui est associé au message DAP reçu (voir ci-dessus).

Si la ligne-7 d'un en-tête de message DAP reçu porte "VRAI", le receveur va renvoyer un message de réponse à l'envoyeur. Cela permet la vérification de la rétroaction SigComp. Comme mentionné précédemment, le message de réponse DOIT être compressé par, et envoyé du compartiment compresseur local qui est un homologue du compartiment compresseur distant. En dehors de cette contrainte, le message de réponse est juste un message DAP régulier qui peut porter un en-tête de message et corps de message arbitraires. Par exemple, le champ "réponse exigée" de la réponse peut aussi être réglé à VRAI, ce qui va déclencher une réponse à la réponse, et ainsi de suite. Noter que comme chaque point d'extrémité a le contrôle sur le champ "réponse exigée" de ses propres messages, cela ne conduit pas à une boucle sans fin. Une mise en œuvre responsable d'envoyeur DAP NE DEVRAIT PAS régler aveuglément ce champ à VRAI sauf si une réponse est désirée. Pour la vérification, le corps de message d'une réponse PEUT contenir l'en-tête de message du message d'origine qui a déclenché la réponse.

Un numéro de séquence de message peut être utilisé par l'envoyeur DAP pour suivre la trace de chaque message qu'il envoie, par exemple, en cas de pertes. Des pertes de messages peuvent arriver sur le chemin ou au point d'extrémité receveur (c'est-à-dire, à cause d'un échec de décompression). L'allocation d'un numéro de séquence de message relève de l'envoyeur. Par exemple, il pourrait être alloué par compartiment ou par point d'extrémité. Cela n'a pas d'impact sur le côté receveur.

A.3 Format de message DAP en ABNF

(Note : voir les règles de base de l'ABNF dans la [RFC2234].)

DAP-message = en-tête de message CRLF [corps de message]

corps de message = *OCTET

en-tête de message = ligne-1 ligne-2 ligne-3 ligne-4 ligne-5 ligne-6 ligne-7 ligne-8

ligne-1 = "Version-DAP" ":" 1*CHIFFRE CRLF
ligne-2 = "Identifiant-de-point-d'extrémité" ":" adresseIPv4 CRLF
ligne-3 = "Identifiant-de-compartiment" ":" 1*CHIFFRE CRLF
ligne-4 = "Numéro-de-séquence-de-message" ":" 1*CHIFFRE CRLF
ligne-5 = "Authentifiant-de-message" ":" ("ACCEPTE" / "REJET") CRLF
ligne-6 = "Opération-de-compartiment" ":" ("CREER" / "CLORE" / "AUCUNE") CRLF
ligne-7 = "Réponse exigée" ":" ("VRAI" / "FAUX")
ligne-8 = "Longueur-de-corps" ":" 1*CHIFFRE CRLF

adresseIPv4 = 1*3CHIFFRE "." 1*3CHIFFRE "." 1*3CHIFFRE "." 1*3CHIFFRE

A.4 Exemple de message DAP

Version-DAP : 1
Identifiant-de-point-d'extrémité : 123.45.67.89
Identifiant-de-compartiment : 2
Numéro-de-séquence-de-message : 0
Authentifiant-de-message : ACCEPTE
Opération-de-compartiment : CREER
Réponse exigée : VRAI
Longueur-de-corps : 228

C'est un message DAP envoyé du point d'extrémité SigComp à l'adresse IP 123.45.67.89. C'est le premier message envoyé du compartiment 2. Prière d'accepter le message, créer le compartiment associé, et renvoyer un message de réponse.

Adresse des auteurs

Abigail Surtees
Siemens/Roke Manor Research
Roke Manor Research Ltd.
Romsey, Hants SO51 0ZN
UK
mél : abigail.surtees@roke.co.uk
URI: http://www.roke.co.uk

Mark A. West
Siemens/Roke Manor Research
Roke Manor Research Ltd.
Romsey, Hants SO51 0ZN
UK
mél : mark.a.west@roke.co.uk
URI: http://www.roke.co.uk

Adam Roach
Estacado Systems
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US
mél : adam@estacado.net

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est fourni par l'activité de soutien administratif de l'IETF (IASA).