Groupe de travail Réseau
Request for Comments: 5524
Catégorie : Sur la voie de la normalisation

D. Cridland, Isode Limited
mai 2009
Traduction Claude Brière de L'Isle

URLFETCH étendu pour parties binaires et converties

Résumé
La commande URLFETCH définie au titre de URLAUTH fournit un mécanisme pour que des tiers obtiennent l'accès aux données des messages dans une mémorisation privée d'utilisateur ; cependant, ces données sont envoyées telles quelles, ce qui ne convient pas pour un certain nombre d'applications. Le présent mémoire spécifie une méthode pour obtenir les données dans des formes convenables pour des applications qui ne sont pas de messagerie.

Table des matières

1. Introduction

Bien que la [RFC4467] fournisse une commande URLFETCH qui peut être utilisée pour déréférencer un URL et retourner les données de la partie de corps, elle le fait en retournant la forme codée, sans métadonnées suffisantes pour décoder. Cela convient pour l'utilisation dans les applications de messagerie comme la [RFC4468], où la forme codée convient, mais pas lorsque l'accès au contenu réel est exigé, comme dans la [RFC5616].

Le présent mémoire spécifie un mécanisme qui retourne des métadonnées supplémentaires sur la partie, comme son type [RFC2046], et supprime tout codage de transfert de contenu qui a été utilisé sur la partie de corps.

2. Conventions utilisées dans ce document

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],.

Les exemples du protocole vont à la ligne pour la facilité de lecture. Les chaînes de protocole sont précédées de C: et S: respectivement pour client et serveur, et les élisions de données sont représentées par [...]. On notera que ces notations sont seulement pour faciliter la lecture.

3. URLFETCH étendu

Cette extension est disponible dans toute mise en œuvre de serveur IMAP qui inclut URLAUTH=BINARY dans sa chaîne de capacités.

De tels serveurs acceptent des paramètres supplémentaires, par URL, à la commande URLFETCH et vont fournir, sur demande, des données spécifiques pour chaque URL déréférencé.

3.1 Paramètres de commandes

La commande URLFETCH est étendue par la fourniture de paramètres facultatifs. La commande URLFETCH étendue est distinguée en incluant chaque URL et paramètre associé dans une liste entre parenthèses. Les cas où il n'y a pas de paramètres ou lorsque l'URL est envoyé sans parenthèses cause le comportement de la commande précisément comme spécifié dans la [RFC4467].

De même, si l'URL est invalide, la commande va se comporter précisément comme spécifié dans la [RFC4467] et retourne un simple NIL.

Les paramètres disponibles sont :

BODYPARTSTRUCTURE : fournit une structure de parties de corps. BODYPARTSTRUCTURE est défini dans la [RFC5259] et donne des métadonnées utiles pour les applications traitantes, comme le type de données.

BINARY : fournit les données sans aucun codage de transfert de contenu. En particulier, cela signifie que les données PEUVENT contenir des octets NUL et ne pas être formées à partir de lignes textuelles. Les données qui contiennent des octets NUL DOIVENT être transférées en utilisant la syntaxe de literal8 définie dans la [RFC3516].

BODY : fournit les données telles qu'elles. Cela donne les mêmes données que la [RFC4467] non étendue comme élément de métadonnées.

Les éléments de métadonnées NE DOIVENT PAS apparaître plus d'une fois par URL demandé, et les clients NE DOIVENT PAS demander à la fois BINARY et BODY.

3.2 Métadonnées de réponse

Afin de porter les métadonnées demandées et fournir des informations supplémentaires au consommateur, la réponse URLFETCH est également étendue.

Suivant l'URL lui-même, les serveurs vont inclure une série d'éléments de métadonnées entre parenthèses. Les éléments de métadonnées définis sont les suivants :

BODYPARTSTRUCTURE : fournit des informations sur les données contenues dans la réponse, comme elles ont été retournées. Il va refléter toutes les conversions ou décodages qui ont eu lieu. En particulier, cela va montrer un codage d'identité si BINARY est aussi demandé.

BINARY : l'élément BINARY fournit le contenu, sans aucun codage de transfert de contenu appliqué. Si ce n'est pas possible (par exemple, le codage de transfert de contenu est inconnu du serveur) alors il PEUT contenir NIL. Les serveurs DOIVENT comprendre tous les codages de transfert de contenu d'identité définis dans la [RFC2045], ainsi que les codages de transformation "Base64" [RFC4648] et "Quoted-Printable" [RFC2045].

BODY : l'élément BODY fournit le contenu tel que trouvé dans le message, avec tout codage de transfert de contenu

encore appliqué. Demander seulement le BODY va fournir une fonctionnalité équivalente à celle de la [RFC4467] non étendue, cependant, en utilisant la syntaxe étendue décrite ici.

Noter qu'à la différence de la [RFC5259], BODYPARTSTRUCTURE n'est pas augmenté du spécificateur de partie, car c'est implicite dans l'URL.

4. Exemple d'échanges

Un client demande les données sans codage de transfert de contenu.

```
C: A001 URLFETCH  ("imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.2;urlauth=anonymous:internal:
       91354a473744909de610943775f92038" BINARY)
S: * URLFETCH "imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.2;urlauth=anonymous:internal:
       91354a473744909de610943775f92038" (BINARY {28}
S: Si vis pacem, para bellum.
S:
S: )
S: A001 OK URLFETCH completed
```

Noter qu'ici les données ne contiennent aucun octet NUL ; donc une syntaxe de literal -- pas literal8 -- a été utilisée.

Un client demande encore des données sans codage de transfert de contenu, mais demande cette fois la structure de corps.

```
C: A001 URLFETCH  ("imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.3;urlauth=anonymous:internal:
       ae354a473744909de610943775f92038" BINARY BODYPARTSTRUCTURE)
S: * URLFETCH "imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.3;urlauth=anonymous:internal:
       ae354a473744909de610943775f92038" (BODYPARTSTRUCTURE
       ("IMAGE" "PNG" () NIL NIL "BINARY" 123)) (BINARY ~{123}
S: [123 octets de données, dont certaines sont NUL])
S: A001 OK URLFETCH completed
```

Un client demande seulement la structure de corps.

```
C: A001 URLFETCH  ("imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.3;urlauth=anonymous:internal:
       ae354a473744909de610943775f92038" BODYPARTSTRUCTURE)
S: * URLFETCH "imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.3;urlauth=anonymous:internal:
       ae354a473744909de610943775f92038" (BODYPARTSTRUCTURE
       ("IMAGE" "PNG" () NIL NIL "BASE64" 164))
S: A001 OK URLFETCH completed
```

Un client demande la structure de corps et le contenu original.

```
C: A001 URLFETCH  ("imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.3;urlauth=anonymous:internal:
       ae354a473744909de610943775f92038" BODYPARTSTRUCTURE BODY)
S: * URLFETCH "imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.3;urlauth=anonymous:internal:
       ae354a473744909de610943775f92038" (BODYPARTSTRUCTURE
       ("IMAGE" "PNG" () NIL NIL "BASE64" 164)) (BODY {164}
S: [164 octets de données codées en base64])
S: A001 OK URLFETCH completed
```

Certaines parties ne peuvent pas être décodées, donc le serveur va fournir la BODYPARTSTRUCTURE de la partie telle

qu'elle est et fournir NIL pour le contenu binaire :

```
C: A001 URLFETCH ("imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.4;urlauth=anonymous:internal:
       87ecbd02095b815e699503fc20d869c8" BODYPARTSTRUCTURE BINARY)
S: * URLFETCH "imap://joe@exemple.com/INBOX/;uid=20/;
       section=1.4;urlauth=anonymous:internal:
       87ecbd02095b815e699503fc20d869c8" (BODYPARTSTRUCTURE
       ("IMAGE" "PNG" () NIL NIL "X-BLURDYBLOOP" 123))
       (BINARY NIL)
S: A001 OK URLFETCH completed
```

Cependant, si une partie n'existe simplement pas, ou si l'URI est invalide pour quelque autre raison, alors NIL est retourné à la place des métadonnées :

```
C: A001 URLFETCH ("imap://joe@exemple.com/INBOX/;uid=20/;
       section=200;urlauth=anonymous:internal:
       88066d37e2e5410e1a6486350a8836ee" BODYPARTSTRUCTURE BODY)
S: * URLFETCH "imap://joe@exemple.com/INBOX/;uid=20/;
       section=200;urlauth=anonymous:internal:
       88066d37e2e5410e1a6486350a8836ee" NIL
S: A001 OK URLFETCH completed
```

5. Syntaxe formelle

Cette syntaxe formelle utilise l'ABNF comme spécifié dans la [RFC5234], et inclut des productions définies dans les [RFC4467], [RFC3516], et [RFC3501].

```
capability =/ "URLAUTH=BINARY"

; pour les paramètres de commande, voir le paragraphe 3.1

urlfetch =  "URLFETCH" 1*(SP url-fetch-arg)

url-fetch-arg =  url-fetch-simple / url-fetch-ext

url-fetch-simple =  url-full                                    ; URLFETCH non étendu.

url-fetch-ext =  "(" url-full *(SP url-fetch-param) ")"
; Si aucun paramètre url-fetch n'est présent, comme non étendu.

url-fetch-param  =  "BODY" / "BINARY" / "BODYPARTSTRUCTURE" / atom

; Réponse, voir le paragraphe 3.2

urlfetch-données =  "*" SP "URLFETCH" 1*(SP (urldata-simple / urldata-ext / urldata-erreur))

urldata-erreur  =  SP url-full SP nil

urldata-simple =  SP url-full SP nstring
; Si le client produit url-fetch-simple, le serveur DOIT répondre avec urldata-simple.

urldata-ext =  SP url-full url-metadata

url-metadata =  1*(SP "(" url-metadata-el ")")

url-metadata-el =  url-meta-bodystruct / url-meta-body / url-meta-binary

url-meta-bodystruct =  "BODYPARTSTRUCTURE" SP body
```

url-meta-binary = "BINARY" SP (nstring / literal8)
; Si le contenu contient un octet NUL, literal8 DOIT être utilisé.
; Autrement, le contenu DEVRAIT utiliser nstring.
; Au décodage d'une erreur, NIL devrait être utilisé.

url-meta-body = "BODY" SP nstring

6. Considérations relatives à l'IANA

Les capacités IMAP4 sont enregistrées en publiant une RFC sur la voie de la normalisation ou une RFC expérimentale approuvée par l'IESG.

Le présent document définit la capacité IMAP URLFETCH=BINARY. L'IANA l'a ajoutée en conséquence au registre.

7. Considérations sur la sécurité

Les mises en œuvres sont invitées à se reporter aux considérations sur la sécurité des [RFC3501], [RFC4467], et [RFC3516].

La capacité du détenteur d'un URL à aller chercher des métadonnées sur le contenu pointé par l'URL ainsi que le contenu lui-même permet à un attaquant potentiel d'en découvrir plus sur le contenu que ce qui était précédemment possible, incluant son nom de fichier original et la description fournie par l'utilisateur.

La valeur ajoutée de ces informations pour un attaquant est marginale, et s'applique seulement aux URL pour lesquels l'attaquant n'a pas d'accès direct, comme celles produites par la [RFC4467]. Les mises en œuvres sont donc invitées à se reporter aux considérations sur la sécurité présentes dans la [RFC4467].

8. Remerciements

Des commentaires ont été reçus sur cette idée et/ou document de Neil Cook, Philip Guenther, Alexey Melnikov, Ken Murchison, et d'autres. En accord ou en désaccord, ces commentaires ont précisé et influencé le présent document.

9. Références

9.1 Références normatives

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", novembre 1996. *(D. S., MàJ par* 2184, 2231, 5335.)

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

[RFC3501] M. Crispin, "Protocole d'accès au message Internet - version 4rev1", mars 2003. *(P.S. ; MàJ par* RFC4466, 4469, 4551, 5032, 5182, 7817, 8314, 8437, 8474 ; *remplacée par la* RFC9051

[RFC3516] L. Nerenberg, "Extension Contenu binaire à IMAP4", avril 2003. (*MàJ par* RFC4466) *(P.S.)*

[RFC4467] M. Crispin, "Protocole d'accès au message Internet (IMAP) - Extension URLAUTH", mai 2006. (*P.S. ; MàJ par* RFC5092)

[RFC4648] S. Josefsson, "Codages de données Base16, Base32 et Base64", octobre 2006. (*Remplace* RFC3548) *(P.S.)*

[RFC5234] D. Crocker, P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", janvier 2008. (STD0068)

[RFC5259] A. Melnikov et P. Coates, "Protocole d'accès au message Internet - extension CONVERT", juillet 2008. *(P.S.)*

9.2 Références pour information

[RFC2046] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 2 : Types de support", novembre 1996. *(D. S., MàJ par* 2646, 3798, 5147, 6657, 8098)

[RFC4468] C. Newman, "Extension BURL de soumission de message", mai 2006. (*MàJ* RFC3463) (*MàJ par* RFC5248) *(P.S.)*

[RFC5616] N. Cook, "Transformation en flux directs des pièces jointes de la messagerie Internet", août 2009. *(Information)*

Adresse de l'auteur

Dave Cridland
Isode Limited
5 Castle Business Village
36, Station Road
Hampton, Middlesex TW12 2BX
GB

mél : dave.cridland@isode.com