Network Working Group
Request for Comments: 5524
Category: Standards Track

D. Cridland
Isode Limited

May 2009

Extended URLFETCH for Binary and Converted Parts

Status of This Memo

Copyright Notice

Abstract

The URLFETCH command defined as part of URLAUTH provides a mechanism for third parties to gain access to data held within messages in a user's private store; however, this data is sent verbatim, which is not suitable for a number of applications. This memo specifies a method for obtaining data in forms suitable for non-mail applications.

Table of Contents

1. Introduction

Although [URLAUTH] provides a URLFETCH command that can be used to dereference a URL and return the body-part data, it does so by returning the encoded form, without sufficient metadata to decode. This is suitable for use in mail applications such as [BURL], where the encoded form is suitable, but not where access to the actual content is required, such as in [STREAMING].

This memo specifies a mechanism that returns additional metadata about the part, such as its [MEDIATYPE] type, as well as removes any content transfer encoding that was used on the body part.

2. Conventions Used in This Document

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 [KEYWORDS].

Protocol examples are line-wrapped for clarity. Protocol strings are prefixed with C: and S: for client and server respectively, and elided data is represented by [...]. Implementors should note these notations are for editorial clarity only.

3. Extended URLFETCH

This extension is available in any IMAP server implementation that includes URLAUTH=BINARY within its capability string.

Such servers accept additional, per-URL parameters to the URLFETCH command and will provide, upon request, specific data for each URL dereferenced.

3.1. Command Parameters

The URLFETCH command is extended by the provision of optional parameters. The extended URLFETCH command is distinct by enclosing each URL and associated parameters in a parenthesized list. Cases where there is an absence of any parameters or where the URL is sent unenclosed cause the command to behave precisely as specified in [URLAUTH].

Similarly, if the URL is invalid, the command will behave precisely as specified in [URLAUTH] and return a simple NIL.

Available parameters are:

BODYPARTSTRUCTURE
: Provide a BODYPARTSTRUCTURE.

  BODYPARTSTRUCTURE is defined in [CONVERT] and provides metadata useful for processing applications, such as the type of data.

BINARY
: Provide the data without any Content-Transfer-Encoding.

  In particular, this means that the data MAY contain NUL octets and not be formed from textual lines. Data containing NUL octets MUST be transferred using the literal8 syntax defined in [BINARY].

BODY
: Provide the data as-is.

  This will provide the same data as the unextended [URLAUTH] as a metadata item.

Metadata items MUST NOT appear more than once per URL requested, and clients MUST NOT request both BINARY and BODY.

3.2. Response Metadata

In order to carry any requested metadata and provide additional information to the consumer, the URLFETCH response is similarly extended.

Following the URL itself, servers will include a series of parenthesized metadata elements. Defined metadata elements are as follows:

BODYPARTSTRUCTURE
The BODYPARTSTRUCTURE provides information about the data contained in the response, as it has been returned. It will reflect any conversions or decoding that have taken place. In particular, this will show an identity encoding if BINARY is also requested.

BINARY
The BINARY item provides the content, without any content transfer encoding applied. If this is not possible (for example, the content transfer encoding is unknown to the server), then this MAY contain NIL. Servers MUST understand all identity content transfer encodings defined in [MIME], as well as the transformation encodings "Base64" [BASE64] and "Quoted-Printable" [MIME].

BODY
The BODY item provides the content as found in the message, with any content transfer encoding still applied. Requesting only the BODY will provide equivalent functionality to the unextended [URLAUTH], however, using the extended syntax described herein.

Note that unlike [CONVERT], BODYPARTSTRUCTURE is not appended with the part specifier, as this is implicit in the URL.

4. Example Exchanges

A client requests the data with no content transfer encoding.

```
C: A001 URLFETCH  ("imap://joe@example.com/INBOX/;uid=20/;
   section=1.2;urlauth=anonymous:internal:
   91354a473744909de610943775f92038" BINARY)
S: * URLFETCH "imap://joe@example.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
```

Note that the data here does not contain a NUL octet; therefore, a literal -- not literal8 -- syntax has been used.

A client again requests data with no content transfer encoding, but this time requests the body structure.

```
      C: A001 URLFETCH  ("imap://joe@example.com/INBOX/;uid=20/;
         section=1.3;urlauth=anonymous:internal:
         ae354a473744909de610943775f92038" BINARY BODYPARTSTRUCTURE)
      S: * URLFETCH "imap://joe@example.com/INBOX/;uid=20/;
         section=1.3;urlauth=anonymous:internal:
         ae354a473744909de610943775f92038" (BODYPARTSTRUCTURE
         ("IMAGE" "PNG" () NIL NIL "BINARY" 123)) (BINARY ~{123}
      S: [123 octets of data, some of which is NUL])
      S: A001 OK URLFETCH completed
```

A client requests only the body structure.

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

A client requests the body structure and the original content.

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

Some parts cannot be decoded, so the server will provide the BODYPARTSTRUCTURE of the part as is and provide NIL for the binary content:

```
      C: A001 URLFETCH ("imap://joe@example.com/INBOX/;uid=20/;
         section=1.4;urlauth=anonymous:internal:
         87ecbd02095b815e699503fc20d869c8" BODYPARTSTRUCTURE BINARY)
      S: * URLFETCH "imap://joe@example.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
```

If a part simply doesn't exist, however, or the URI is invalid for some other reason, then NIL is returned instead of metadata:

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

5. Formal Syntax

This formal syntax uses ABNF as specified in [ABNF], and includes productions defined in [URLAUTH], [BINARY], and [IMAP].

```
capability       =/ "URLAUTH=BINARY"

   ; Command parameters; see Section 3.1

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

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

url-fetch-simple =  url-full
   ; Unextended URLFETCH.

url-fetch-ext    =  "(" url-full *(SP url-fetch-param) ")"
   ; If no url-fetch-param present, as unextended.

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

   ; Response; see Section 3.2

urlfetch-data    =  "*" SP "URLFETCH"
                    1*(SP (urldata-simple / urldata-ext /
                           urldata-error))

urldata-error    =  SP url-full SP nil

urldata-simple   =  SP url-full SP nstring
   ; If client issues url-fetch-simple, server MUST respond with
   ; 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 )
      ; If content contains a NUL octet, literal8 MUST be used.
      ; Otherwise, content SHOULD use nstring.
      ; On decoding error, NIL should be used.

   url-meta-body         =  "BODY" SP nstring
```

6. IANA Considerations

IMAP4 capabilities are registered by publishing a Standards Track or IESG-approved Experimental RFC.

This document defines the URLFETCH=BINARY IMAP capability. IANA has added it to the registry accordingly.

7. Security Considerations

Implementors are directed to the security considerations within [IMAP], [URLAUTH], and [BINARY].

The ability of the holder of a URL to be able to fetch metadata about the content pointed to by the URL as well as the content itself allows a potential attacker to discover more about the content than was previously possible, including its original filename and user-supplied description.

The additional value of this information to an attacker is marginal, and applies only to those URLs for which the attacker does not have direct access, such as those produced by [URLAUTH]. Implementors are therefore directed to the security considerations present in [URLAUTH].

8. Acknowledgements

Comments were received on this idea and/or document from Neil Cook, Philip Guenther, Alexey Melnikov, Ken Murchison, and others. Whether in agreement or dissent, the comments have refined and otherwise influenced this document.

9. References

9.1. Normative References

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[BINARY] Nerenberg, L., "IMAP4 Binary Content Extension", RFC 3516, April 2003.

[CONVERT] Melnikov, A. and P. Coates, "Internet Message Access Protocol - CONVERT Extension", RFC 5259, July 2008.

[IMAP] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

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

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[URLAUTH] Crispin, M., "Internet Message Access Protocol (IMAP) - URLAUTH Extension", RFC 4467, May 2006.

9.2. Informative References

[BURL] Newman, C., "Message Submission BURL Extension", RFC 4468, May 2006.

[MEDIATYPE] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[STREAMING] Cook, N., "Streaming Internet Messaging Attachments", Work in Progress, March 2009.

Author's Address

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

EMail: dave.cridland@isode.com