                            IMAP URL Scheme

Status of This Memo

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

Abstract

   IMAP (RFC 3501) is a rich protocol for accessing remote message
   stores. It provides an ideal mechanism for accessing public mailing
   list archives as well as private and shared message stores. This
   document defines a URL scheme for referencing objects on an IMAP
   server.

   This document obsoletes RFC 2192. It also updates RFC 4467.

Table of Contents

1. Introduction

   The IMAP URL scheme is used to designate IMAP servers, mailboxes,
   messages, MIME bodies [MIME], and search programs on Internet hosts
   accessible using the IMAP protocol over TCP.

   The IMAP URL follows the common Internet scheme syntax as defined in
   [URI-GEN]. If :<port> is omitted, the port defaults to 143 (as
   defined in Section 2.1 of [IMAP4]).

An absolute IMAP URL takes one of the following forms:

    imap://<iserver>[/]

    imap://<iserver>/<enc-mailbox>[<uidvalidity>][?<enc-search>]

    imap://<iserver>/<enc-mailbox>[<uidvalidity>]<iuid>
     [<isection>][<ipartial>][<iurlauth>]

The first form is used to refer to an IMAP server (see Section 4),
the second form refers to the contents of a mailbox or a set of
messages resulting from a search (see Section 5), and the final form
refers to a specific message or message part, and possibly a byte
range in that part (see Section 6). If [URLAUTH] extension is
supported, then the final form can have the <iurlauth> component (see
Section 6.1 for more details).

The <iserver> component common to all types of absolute IMAP URLs has
the following syntax expressed in ABNF [ABNF]:

    [iuserinfo "@"] host [":" port]

The <iserver> component is the same as "authority" defined in
[URI-GEN]. The syntax and uses of the <iuserinfo> ("IMAP userinfo
component") are described in detail in Section 3. The syntax of
<host> and <port> is described in [URI-GEN].

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].

This document references many productions from [URI-GEN]. When the
document needs to emphasize IMAP URI-specific differences from [URI-
GEN] (i.e., for parts of IMAP URIs that have more restricted syntax
than generic URIs), it uses a non-terminal i<foo> to define an IMAP-
specific version of the non-terminal <foo> from [URI-GEN].

Note that the ABNF syntax shown in Section 11 is normative. Sections
2-6 may use a less formal syntax that does not necessarily match the
normative ABNF shown in Section 11. If there are any differences
between the syntax shown in Sections 2-6 and Section 11, then the
syntax shown in Section 11 must be treated as authoritative. Non-
syntax requirements included in Sections 2-6 are, of course,
normative.

3. IMAP userinfo Component (iuserinfo)

   The <iuserinfo> component conforms to the generic syntax of
   <userinfo> defined in [URI-GEN]. It has the following syntax
   expressed in ABNF [ABNF]:

       enc-user [iauth] / [enc-user] iauth

   The meaning of the different parts is described in subsections of
   this section.

3.1. IMAP Mailbox Naming Scope

   The "enc-user" part of the "iuserinfo" component, if present, denotes
   mailbox naming scope. If it is absent, the IMAP URL can only
   reference mailboxes with globally unique names, i.e., mailboxes with
   names that don't change depending on the user the client
   authenticated as to the IMAP server. Note that not all IMAP
   implementations support globally unique names.

   For example, a personal mailbox described by the following URL
   <imap://michael@example.org/INBOX> is most likely different from a
   personal mailbox described by <imap://bester@example.org/INBOX>, even
   though both URLs use the same mailbox name.

3.2. IMAP User Name and Authentication Mechanism

   The userinfo component (see [URI-GEN]) of an IMAP URI may contain an
   IMAP user name (a.k.a. authorization identity [SASL], "enc-user")
   and/or an authentication mechanism. (Note that the "enc-user" also
   defines a mailbox naming scope as described in Section 3.1). The
   IMAP user name and the authentication mechanism are used in the
   "LOGIN" or "AUTHENTICATE" commands after making the connection to the
   IMAP server.

   If no user name and no authentication mechanism are supplied, the
   client MUST authenticate as anonymous to the server. If the server
   advertises AUTH=ANONYMOUS IMAP capability, the client MUST use the
   AUTHENTICATE command with ANONYMOUS [ANONYMOUS] SASL mechanism. If
   SASL ANONYMOUS is not available, the (case-insensitive) user name
   "anonymous" is used with the "LOGIN" command and the Internet email
   address of the end user accessing the resource is supplied as the
   password. The latter option is given in order to provide for
   interoperability with deployed servers.

   Note that, as described in RFC 3501, the "LOGIN" command MUST NOT be
   used when the IMAP server advertises the LOGINDISABLED capability.

An authentication mechanism (as used by the IMAP AUTHENTICATE
command) can be expressed by adding ";AUTH=<enc-auth-type>" to the
end of the user name in an IMAP URL. When such an <enc-auth-type> is
indicated, the client SHOULD request appropriate credentials from
that mechanism and use the "AUTHENTICATE" command instead of the
"LOGIN" command. If no user name is specified, one MUST be obtained
from the mechanism or requested from the user/configuration as
appropriate.

The string ";AUTH=*" indicates that the client SHOULD select an
appropriate authentication mechanism. (Though the '*' character in
this usage is not strictly a delimiter, it is being treated like a
sub-delim [URI-GEN] in this instance. It MUST NOT be percent-encoded
in this usage, as ";AUTH=%2A" will not match this production.) It
MAY use any mechanism listed in the response to the CAPABILITY
command (or CAPABILITY response code) or use an out-of-band security
service resulting in a PREAUTH connection. If no user name is
specified and no appropriate authentication mechanisms are available,
the client SHOULD fall back to anonymous login as described above.
The behavior prescribed in this section allows a URL that grants
read-write access to authorized users and read-only anonymous access
to other users.

If a user name is included with no authentication mechanism, then
";AUTH=*" is assumed.

Clients must take care when resolving a URL that requires or requests
any sort of authentication, since URLs can easily come from untrusted
sources. Supplying authentication credentials to the wrong server
may compromise the security of the user's account; therefore, the
program resolving the URL should meet at least one of the following
criteria in this case:

1) The URL comes from a trusted source, such as a referral server
   that the client has validated and trusts according to site policy.
   Note that user entry of the URL may or may not count as a trusted
   source, depending on the experience level of the user and site
   policy.

2) Explicit local site policy permits the client to connect to the
   server in the URL. For example, a company example.com may have a
   site policy to trust all IMAP server names ending in example.com,
   whereas such a policy would be unwise for example.edu where random
   students can set up IMAP servers.

3) The user confirms that connecting to that domain name with the
   specified credentials and/or mechanism is permitted. For example,
   when using "LOGIN" or SASL PLAIN with Transport Layer Security

(TLS), the IMAP URL client presents a dialog box "Is it OK to send
your password to server "example.com"? Please be aware the owners
of example.com will be able to reuse your password to connect to
other servers on your behalf".

4) A mechanism is used that validates the server before passing
   potentially compromising client credentials. For example, a site
   has a designated TLS certificate used to certify site-trusted IMAP
   server certificates, and this has been configured explicitly into
   the IMAP URL client. Another example is use of a Simple
   Authentication and Security Layer (SASL) mechanism such as
   DIGEST-MD5 [DIGEST-MD5], which supports mutual authentication.

5) An authentication mechanism is used that will not reveal any
   information to the server that could be used to compromise future
   connections. Examples are SASL ANONYMOUS [ANONYMOUS] or GSSAPI
   [GSSAPI].

URLs that do not include a user name but include an authentication
mechanism (";AUTH=<mech>") must be treated with extra care, since for
some <mech>s they are more likely to compromise the user's primary
account. A URL containing ";AUTH=*" must also be treated with extra
care since it might fall back on a weaker security mechanism.
Finally, clients are discouraged from using a plaintext password as a
fallback with ";AUTH=*" unless the connection has strong encryption.

A program interpreting IMAP URLs MAY cache open connections to an
IMAP server for later reuse. If a URL contains a user name, only
connections authenticated as that user may be reused. If a URL does
not contain a user name or authentication mechanism, then only an
anonymous connection may be reused.

Note that if unsafe or reserved characters such as " " (space) or ";"
are present in the user name or authentication mechanism, they MUST
be percent-encoded as described in [URI-GEN].

3.3. Limitations of enc-user

As per Sections 3.1 and 3.2 of this document, the IMAP URI enc-user
has two purposes:

1) It provides context for user-specific mailbox paths such as
   "INBOX" (Section 3.1).

2) It specifies that resolution of the URL requires logging in as
   that user and limits use of that URL to only that user (Section
   3.2).

An obvious limitation of using the same field for both purposes is
that the URL can be resolved only by the mailbox owner. In order to
avoid this restriction, implementations should use globally unique
mailbox names (see Section 3.1) whenever possible.

   Note: There is currently no general way in IMAP of learning a
   globally unique name for a mailbox. However, by looking at the
   NAMESPACE [NAMESPACE] command result, it is possible to determine
   whether or not a mailbox name is globally unique.

The URLAUTH component overrides the second purpose of the enc-user in
the IMAP URI and by default permits the URI to be resolved by any
user permitted by the <access> identifier. URLAUTH and <access>
identifier are described in Section 6.1.

4. IMAP Server

An IMAP URL referring to an IMAP server has the following form:

   imap://<iserver>[/]

This URL type is frequently used to describe a location of an IMAP
server, both in referrals and in configuration. It may optionally
contain the <iuserinfo> component (see Sections 3 and 11). A program
interpreting this URL would issue the standard set of commands it
uses to present a view of the content of the IMAP server, as visible
to the user described by the "enc-user" part of the <iuserinfo>
component, if the "enc-user" part is specified.

5. Lists of Messages

An IMAP URL referring to a list of messages has the following form:

   imap://<iserver>/<enc-mailbox>[<uidvalidity>][?<enc-search>]

The <enc-mailbox> field is used as the argument to the IMAP4 "SELECT"
or "EXAMINE" command. Note that if unsafe or reserved characters
such as " " (space), ";", or "?" are present in <enc-mailbox>, they
MUST be percent-encoded as described in [URI-GEN].

The <uidvalidity> field is optional. If it is present, it MUST be
the same as the value of IMAP4 UIDVALIDITY response code at the time
the URL was created. This MUST be used by the program interpreting
the IMAP URL to determine if the URL is stale. If the IMAP URL is
stale, then the program should behave as if the corresponding mailbox
doesn't exist.

Note that the <uidvalidity> field is a modifier to the <enc-mailbox>,
i.e., it is considered a part of the last "component" (as used in
[URI-GEN]) of the <enc-mailbox>. This is significant during relative
URI resolution.

The "?<enc-search>" field is optional. If it is not present, the
program interpreting the URL will present the entire content of the
mailbox.

If the "?<enc-search>" field is present, the program interpreting the
URL should use the contents of this field as arguments following an
IMAP4 SEARCH command. These arguments are likely to contain unsafe
characters such as " " (space) (which are likely to be present in the
<enc-search>). If unsafe characters are present, they MUST be
percent-encoded as described in [URI-GEN].

Note that quoted strings and non-synchronizing literals [LITERAL+]
are allowed in the <enc-search> content; however, synchronizing
literals are not allowed, as their presence would effectively mean
that the agent interpreting IMAP URLs needs to parse an <enc-search>
content, find all synchronizing literals, and perform proper command
continuation request handling (see Sections 4.3 and 7 of [IMAP4]).

6. A Specific Message or Message Part

An IMAP URL referring to a specific message or message part has the
following form:

    imap://<iserver>/<enc-mailbox>[<uidvalidity>]<iuid>
    [<isection>][<ipartial>][<iurlauth>]

The <enc-mailbox> and [uidvalidity] are as defined in Section 5
above.

If <uidvalidity> is present in this form, it SHOULD be used by the
program interpreting the URL to determine if the URL is stale.

The <iuid> refers to an IMAP4 message Unique Identifier (UID), and it
SHOULD be used as the <set> argument to the IMAP4 "UID FETCH"
command.

The <isection> field is optional. If not present, the URL refers to
the entire Internet message as returned by the IMAP command "UID
FETCH <uid> BODY.PEEK[]". If present, the URL refers to the object
returned by a "UID FETCH <uid> BODY.PEEK[<section>]" command. The
type of the object may be determined by using a "UID FETCH <uid>
BODYSTRUCTURE" command and locating the appropriate part in the

resulting BODYSTRUCTURE. Note that unsafe characters in [isection]
MUST be percent-encoded as described in [URI-GEN].

The <ipartial> field is optional. If present, it effectively appends
"<<partial-range>>" to the end of the UID FETCH BODY.PEEK[<section>]
command constructed as described in the previous paragraph. In other
words, it allows the client to request a byte range of the
message/message part.

The <iurlauth> field is described in detail in Section 6.1.

6.1. URLAUTH Authorized URL

   URLAUTH authorized URLs are only supported by an IMAP server
   advertising the URLAUTH IMAP capability [URLAUTH].

6.1.1. Concepts

6.1.1.1. URLAUTH

   URLAUTH is a component, appended at the end of a URL, that conveys
   authorization to access the data addressed by that URL. It contains
   an authorized access identifier, an authorization mechanism name, and
   an authorization token. The authorization token is generated from
   the URL, the authorized access identifier, authorization mechanism
   name, and a mailbox access key.

      Note: This specification only allows for the URLAUTH component in
      IMAP URLs describing a message or its part.

6.1.1.2. Mailbox Access Key

   The mailbox access key is an unpredictable, random string. To ensure
   unpredictability, the random string with at least 128 bits of entropy
   is generated by software or hardware (not by the human user).

   Each user has a table of mailboxes and an associated mailbox access
   key for each mailbox. Consequently, the mailbox access key is per-
   user and per-mailbox. In other words, two users sharing the same
   mailbox each have a different mailbox access key for that mailbox,
   and each mailbox accessed by a single user also has a different
   mailbox access key.

6.1.1.3. Authorized Access Identifier

   The authorized <access> identifier restricts use of the URLAUTH
   authorized URL to certain users authorized on the server, as
   described in Section 6.1.2.

6.1.1.4. Authorization Mechanism

   The authorization mechanism is the algorithm by which the URLAUTH is
   generated and subsequently verified, using the mailbox access key.

6.1.1.5. Authorization Token

   The authorization token is a deterministic string of at least 128
   bits that an entity with knowledge of the secret mailbox access key
   and URL authorization mechanism can use to verify the URL.

6.1.2. URLAUTH Extensions to IMAP URL

   A specific message or message part IMAP URL can optionally contain
   ";EXPIRE=<datetime>" and/or ";URLAUTH=<access>:<mech>:<token>".

   When ";EXPIRE=<datetime>" is used, this indicates the latest date and
   time that the URL is valid. After that date and time, the URL has
   expired and server implementations MUST reject the URL. If
   ";EXPIRE=<datetime>" is not used, the URL has no expiration, but can
   still be revoked using the RESETKEY command [URLAUTH].

   The URLAUTH takes the form ";URLAUTH=<access>:<mech>:<token>", and it
   MUST be at the end of the URL. It is composed of three parts. The
   <access> portion provides the authorized access identifiers that may
   constrain the operations and users that are permitted to use this
   URL. The <mech> portion provides the authorization mechanism used by
   the IMAP server to generate the authorization token that follows.
   The <token> portion provides the authorization token, which can be
   generated using the GENURLAUTH command [URLAUTH].

   The "submit+" <access> identifier prefix, followed by a userid,
   indicates that only a userid authorized as a message submission
   entity on behalf of the specified userid is permitted to use this
   URL. The IMAP server does not validate the specified userid but does
   validate that the IMAP session has an authorization identity that is
   authorized as a message submission entity. The authorized message
   submission entity MUST validate the userid prior to contacting the
   IMAP server.

   The "user+" <access> identifier prefix, followed by a userid,
   indicates that use of this URL is limited to IMAP sessions that are
   logged in as the specified userid (that is, have authorization
   identity as that userid).

      Note: If a SASL mechanism that provides both authorization and
      authentication identifiers is used to authenticate to the IMAP
      server, the "user+" <access> identifier MUST match the

authorization identifier. If the SASL mechanism can't transport
the authorization identifier, the "user+" <access> identifier MUST
match the authorization identifier derived from the authentication
identifier (see [SASL]).

The "authuser" <access> identifier indicates that use of this URL is
limited to authenticated IMAP sessions that are logged in as any
non-anonymous user (that is, have authorization identity as a non-
anonymous user) of that IMAP server. To restate this: use of this
type of URL is prohibited to anonymous IMAP sessions, i.e., any
URLFETCH command containing this type of URL issued in an anonymous
session MUST return NIL in the URLFETCH response.

The "anonymous" <access> identifier indicates that use of this URL is
not restricted by session authorization identity; that is, any IMAP
session in authenticated or selected state (as defined in [IMAP4]),
including anonymous sessions, may issue a URLFETCH [URLAUTH] using
this URL.

The authorization token is represented as an ASCII-encoded
hexadecimal string, which is used to authorize the URL. The length
and the calculation of the authorization token depend upon the
mechanism used, but in all cases, the authorization token is at least
128 bits (and therefore at least 32 hexadecimal digits).

Example:

   <imap://joe@example.com/INBOX/;uid=20/;section=1.2;urlauth=
   submit+fred:internal:91354a473744909de610943775f92038>

7. Relative IMAP URLs

   Relative IMAP URLs are permitted and are resolved according to the
   rules defined in [URI-GEN]. In particular, in IMAP URLs parameters
   (such as ";uid=" or ";section=") are treated as part of the normal
   path with respect to relative URL resolution.

   [URI-GEN] defines four forms of relative URLs: <inetwork-path>,
   <iabsolute-path>, <irelative-path>, and <ipath-empty>. Their syntax
   is defined in Section 11.

   A relative reference that begins with two slash characters is termed
   a network-path reference (<inetwork-path>); such references are
   rarely used, because in most cases they can be replaced with an
   equivalent absolute URL. A relative reference that begins with a
   single slash character is termed an absolute-path reference
   (<iabsolute-path>; see also Section 7.1). A relative reference that
   does not begin with a slash character is termed a relative-path

reference (<irelative-path>; see also Section 7.2). The final form
is <ipath-empty>, which is "same-document reference" (see Section 4.4
of [URI-GEN]).

The following observations about relative URLs are important:

The <iauth> grammar element (which is a part of <iuserinfo>, which
is, in turn, a part of <iserver>; see Section 3) is considered part
of the user name for purposes of resolving relative IMAP URLs. This
means that unless a new user name/server specification is included in
the relative URL, the authentication mechanism is inherited from the
base IMAP URL.

URLs always use "/" as the hierarchy delimiter for the purpose of
resolving paths in relative URLs. IMAP4 permits the use of any
hierarchy delimiter in mailbox names. For this reason, relative
mailbox paths will only work if the mailbox uses "/" as the hierarchy
delimiter. Relative URLs may be used on mailboxes that use other
delimiters, but in that case, the entire mailbox name MUST be
specified in the relative URL or inherited as a whole from the base
URL.

If an IMAP server allows for mailbox names starting with "./" or
"../", ending with "/." or "/..", or containing sequences "/../" or
"/./", then such mailbox names MUST be percent-encoded as described
in [URI-GEN]. Otherwise, they would be misinterpreted as dot-
segments (see Section 3.3 of [URI-GEN]), which are processed
specially during the relative path resolution process.

7.1. absolute-path References

A relative reference that begins with a single slash character is
termed an absolute-path reference (see Section 4.2 of [URI-GEN]). If
an IMAP server permits mailbox names with a leading "/", then the
leading "/" MUST be percent-encoded as described in [URI-GEN].
Otherwise, the produced absolute-path reference URI will be
misinterpreted as a network-path reference [URI-GEN] described by the
<inetwork-path> non-terminal.

7.2. relative-path References

A relative reference that does not begin with a slash character is
termed a relative-path reference [URI-GEN]. Implementations MUST NOT
generate or accept relative-path IMAP references.

See also Section 4.2 of [URI-GEN] for restrictions on relative-path
references.

8. Internationalization Considerations

   IMAP4, Section 5.1.3 [IMAP4] includes a convention for encoding non-
   US-ASCII characters in IMAP mailbox names. Because this convention
   is private to IMAP, it is necessary to convert IMAP's encoding to one
   that can be more easily interpreted by a URL display program. For
   this reason, IMAP's modified UTF-7 encoding for mailboxes MUST be
   converted to UTF-8 [UTF-8]. Since 8-bit octets are not permitted in
   URLs, the UTF-8 octets are percent-encoded as required by the URL
   specification [URI-GEN], Section 2.1. Sample code is included in
   Appendix A to demonstrate this conversion.

   IMAP user names are UTF-8 strings and MUST be percent-encoded as
   required by the URL specification [URI-GEN], Section 2.1.

   Also note that IMAP SEARCH criteria can contain non-US-ASCII
   characters. 8-bit octets in those strings MUST be percent-encoded as
   required by the URL specification [URI-GEN], Section 2.1.

9. Examples

   The following examples demonstrate how an IMAP4 client program might
   translate various IMAP4 URLs into a series of IMAP4 commands.
   Commands sent from the client to the server are prefixed with "C:",
   and responses sent from the server to the client are prefixed with
   "S:".

   The URL:

      <imap://minbari.example.org/gray-council;UIDVALIDITY=385759045/;
      UID=20/;PARTIAL=0.1024>

   may result in the following client commands and server responses:

      <connect to minbari.example.org, port 143>
      S: * OK [CAPABILITY IMAP4rev1 STARTTLS AUTH=ANONYMOUS] Welcome
      C: A001 AUTHENTICATE ANONYMOUS
      S: +
      C: c2hlcmlkYW5AYmFieWxvbjUuZXhhbXBsZS5vcmc=
      S: A001 OK Welcome sheridan@babylon5.example.org
      C: A002 SELECT gray-council
      <client verifies the UIDVALIDITY matches>
      C: A003 UID FETCH 20 BODY.PEEK[]<0.1024>

   The URL:

      <imap://psicorp.example.org/~peter/%E6%97%A5%E6%9C%AC%E8%AA%9E/
      %E5%8F%B0%E5%8C%97>

   may result in the following client commands:

      <connect to psicorp.example.org, port 143>
      S: * OK [CAPABILITY IMAP4rev1 STARTTLS AUTH=CRAM-MD5] Welcome
      C: A001 LOGIN ANONYMOUS bester@psycop.psicorp.example.org
      C: A002 SELECT ~peter/&ZeVnLIqe-/&U,BTFw-
      <commands the client uses for viewing the contents of
       the mailbox>

   The URL:

      <imap://;AUTH=GSSAPI@minbari.example.org/gray-council/;uid=20/
      ;section=1.2>

   may result in the following client commands:

      <connect to minbari.example.org, port 143>
      S: * OK Greetings
      C: A000 CAPABILITY
      S: * CAPABILITY IMAP4rev1 STARTTLS AUTH=GSSAPI
      S: A000 OK
      C: A001 AUTHENTICATE GSSAPI
      <authentication exchange>
      C: A002 SELECT gray-council
      C: A003 UID FETCH 20 BODY.PEEK[1.2]

   If the following relative URL is located in that body part:

      <;section=1.4>

   this could result in the following client commands:

      C: A004 UID FETCH 20 (BODY.PEEK[1.2.MIME]
            BODY.PEEK[1.MIME]
            BODY.PEEK[HEADER.FIELDS (Content-Location)])
      <Client looks for Content-Location headers in
       result. If no such headers, then it does the following>
      C: A005 UID FETCH 20 BODY.PEEK[1.4]

   The URL:

      <imap://;AUTH=*@minbari.example.org/gray%20council?
      SUBJECT%20shadows>

could result in the following:

```
<connect to minbari.example.org, port 143>
S: * OK Welcome
C: A001 CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=DIGEST-MD5
S: A001 OK
C: A002 AUTHENTICATE DIGEST-MD5
<authentication exchange>
S: A002 OK user lennier authenticated
C: A003 SELECT "gray council"
...
C: A004 SEARCH SUBJECT shadows
S: * SEARCH 8 10 13 14 15 16
S: A004 OK SEARCH completed
C: A005 FETCH 8,10,13:16 ALL
...
```

In the example above, the client has implementation-dependent
choices. The authentication mechanism could be anything, including
PREAUTH. The final FETCH command could fetch more or less
information about the messages, depending on what it wishes to
display to the user.

The URL:

```
<imap://john;AUTH=*@minbari.example.org/babylon5/personel?
charset%20UTF-8%20SUBJECT%20%7B14+%7D%0D%0A%D0%98%D0%B2%
D0%B0%D0%BD%D0%BE%D0%B2%D0%B0>
```

shows that 8-bit data can be sent using non-synchronizing literals
[LITERAL+]. This could result in the following:

```
<connect to minbari.example.org, port 143>
S: * OK Hi there
C: A001 CAPABILITY
S: * CAPABILITY IMAP4rev1 LITERAL+ AUTH=DIGEST-MD5
S: A001 OK
C: A002 AUTHENTICATE DIGEST-MD5
<authentication exchange>
S: A002 OK user john authenticated
C: A003 SELECT babylon5/personel
...
C: A004 SEARCH CHARSET UTF-8 SUBJECT {14+}
C: XXXXXXXXXXXXXX
S: * SEARCH 7 10 12
S: A004 OK SEARCH completed
C: A005 FETCH 7,10,12 ALL
```

        ...

    where XXXXXXXXXXXXXX is 14 bytes of UTF-8 encoded data as specified
    in the URL above.

9.1. Examples of Relative URLs

    The following absolute-path reference

        </foo/;UID=20/..>

    is the same as

        </foo>

    That is, both of them reference the mailbox "foo" located on the IMAP
    server described by the corresponding Base URI.

    The following relative-path reference

        <;UID=20>

    references a message with UID in the mailbox specified by the Base
    URI.

    The following edge case example demonstrates that the ;UIDVALIDITY=
    modifier is a part of the mailbox name as far as relative URI
    resolution is concerned:

        <..;UIDVALIDITY=385759045/;UID=20>

    In this example, ".." is not a dot-segment [URI-GEN].

10. Security Considerations

    Security considerations discussed in the IMAP specification [IMAP4]
    and the URI specification [URI-GEN] are relevant. Security
    considerations related to authenticated URLs are discussed in Section
    3.2 of this document.

    Many email clients store the plaintext password for later use after
    logging into an IMAP server. Such clients MUST NOT use a stored
    password in response to an IMAP URL without explicit permission from
    the user to supply that password to the specified host name.

    Clients resolving IMAP URLs that wish to achieve data confidentiality
    and/or integrity SHOULD use the STARTTLS command (if supported by the

server) before starting authentication, or use a SASL mechanism, such
as GSSAPI, that provides a confidentiality security layer.

10.1. Security Consideration Specific to URLAUTH Authorized URL

The "user+<userid>" <access> identifier limits resolution of that URL
to a particular userid, whereas the "submit+<userid>" <access>
identifier is more general and simply requires that the session be
authorized by a user that has been granted a "submit" role within the
authentication system. Use of either of these mechanisms limits the
scope of the URL. An attacker who cannot authenticate using the
appropriate credentials cannot make use of the URL.

The "authuser" and "anonymous" <access> identifiers do not have this
level of protection. These access identifiers are primarily useful
for public export of data from an IMAP server, without requiring that
it be copied to a web or anonymous FTP server.

The decision to use the "authuser" <access> identifier should be made
with caution. An "authuser" <access> identifier can be used by any
authorized user of the IMAP server; therefore, use of this access
identifier should be limited to content that may be disclosed to any
authorized user of the IMAP server.

The decision to use the "anonymous" <access> identifier should be
made with extreme caution. An "anonymous" <access> identifier can be
used by anyone; therefore, use of this access identifier should be
limited to content that may be disclosed to anyone.

11. ABNF for IMAP URL Scheme

Formal syntax is defined using ABNF [ABNF], extending the ABNF rules
in Section 9 of [IMAP4]. Elements not defined here can be found in
[ABNF], [IMAP4], [IMAPABNF], or [URI-GEN]. Strings are not case
sensitive, and free insertion of linear white space is not permitted.

```
sub-delims-sh = "!" / "$" / "'" / "(" / ")" /
                "*" / "+" / ","
                 ;; Same as [URI-GEN] sub-delims,
                 ;; but without ";", "&" and "=".

uchar         = unreserved / sub-delims-sh / pct-encoded

achar         = uchar / "&" / "="
                 ;; Same as [URI-GEN] 'unreserved / sub-delims /
                 ;; pct-encoded', but ";" is disallowed.

bchar         = achar / ":" / "@" / "/"
```

```
   enc-auth-type    = 1*achar
                   ; %-encoded version of [IMAP4] "auth-type"


   enc-mailbox      = 1*bchar
                   ; %-encoded version of [IMAP4] "mailbox"


   enc-search       = 1*bchar
                             ; %-encoded version of [IMAPABNF]
                             ; "search-program".  Note that IMAP4
                             ; literals may not be used in
                             ; a "search-program", i.e., only
                             ; quoted or non-synchronizing
                             ; literals (if the server supports
                             ; LITERAL+ [LITERAL+]) are allowed.


   enc-section      = 1*bchar
                   ; %-encoded version of [IMAP4] "section-spec"


   enc-user         = 1*achar
                   ; %-encoded version of [IMAP4] authorization
                   ; identity or "userid".


   imapurl          = "imap://" iserver ipath-query
               ; Defines an absolute IMAP URL


   ipath-query      = ["/" [ icommand ]]
                   ; Corresponds to "path-abempty [ "?" query ]"
                   ; in [URI-GEN]
```

   Generic syntax for relative URLs is defined in Section 4.2 of
   [URI-GEN]. For ease of implementation, the relative IMAP URL syntax
   is defined below:

```
   imapurl-rel      = inetwork-path

                       / iabsolute-path
                       / irelative-path
                       / ipath-empty


   inetwork-path    = "//" iserver ipath-query
               ; Corresponds to '"//" authority path-abempty
               ; [ "?" query ]' in [URI-GEN]


   iabsolute-path = "/" [ icommand ]
               ; icommand, if present, MUST NOT start with '/'.
               ;
               ; Corresponds to 'path-absolute [ "?" query ]'
               ; in [URI-GEN]
```

```
   irelative-path  = imessagelist /
                     imsg-or-part
               ; Corresponds to 'path-noscheme [ "?" query ]'
               ; in [URI-GEN]

   imsg-or-part    = ( imailbox-ref "/" iuid-only ["/" isection-only]
                       ["/" ipartial-only] ) /
                     ( iuid-only ["/" isection-only]
                       ["/" ipartial-only] ) /
                     ( isection-only ["/" ipartial-only] ) /
                     ipartial-only

   ipath-empty     = 0<pchar>
                      ; Zero characters.
                      ; The same-document reference.

   The following three rules are only used in the presence of the IMAP
   [URLAUTH] extension:

   authimapurl     = "imap://" iserver "/" imessagepart
                      ; Same as "imapurl" when "[icommand]" is
                      ; "imessagepart"

   authimapurlfull = authimapurl iurlauth
                      ; Same as "imapurl" when "[icommand]" is
                      ; "imessagepart iurlauth"

   authimapurlrump = authimapurl iurlauth-rump


   enc-urlauth     = 32*HEXDIG

   iurlauth        = iurlauth-rump iua-verifier

   iua-verifier    = ":" uauth-mechanism ":" enc-urlauth

   iurlauth-rump   = [expire] ";URLAUTH=" access

   access          = ("submit+" enc-user) / ("user+" enc-user) /
                       "authuser" / "anonymous"

   expire          = ";EXPIRE=" date-time
                        ; date-time is defined in [DATETIME]

   uauth-mechanism = "INTERNAL" / 1*(ALPHA / DIGIT / "-" / ".")
                        ; Case-insensitive.
                        ; New mechanisms MUST be registered with IANA.
```

```
   iauth             = ";AUTH=" ( "*" / enc-auth-type )

   icommand          = imessagelist /
                         imessagepart [iurlauth]

   imailbox-ref      = enc-mailbox [uidvalidity]

   imessagelist      = imailbox-ref [ "?" enc-search ]
                    ; "enc-search" is [URI-GEN] "query".

   imessagepart      = imailbox-ref iuid [isection] [ipartial]

   ipartial          = "/" ipartial-only

   ipartial-only     = ";PARTIAL=" partial-range

   isection          = "/" isection-only

   isection-only     = ";SECTION=" enc-section

   iserver           = [iuserinfo "@"] host [ ":" port ]
                           ; This is the same as "authority" defined
                           ; in [URI-GEN].  See [URI-GEN] for "host"
                           ; and "port" definitions.

   iuid              = "/" iuid-only

   iuid-only         = ";UID=" nz-number
                    ; See [IMAP4] for "nz-number" definition

   iuserinfo         = enc-user [iauth] / [enc-user] iauth
                           ; conforms to the generic syntax of
                           ; "userinfo" as defined in [URI-GEN].

   partial-range     = number ["." nz-number]
                    ; partial FETCH.  The first number is
                           ; the offset of the first byte,
                           ; the second number is the length of
                           ; the fragment.

   uidvalidity       = ";UIDVALIDITY=" nz-number
                           ; See [IMAP4] for "nz-number" definition
```

12. IANA Considerations

   IANA has updated the "imap" definition in the "Uniform Resource
   Identifier scheme registry" to point to this document.

   The registration template (as per [URI-REG]) is specified in Section
   12.1 of this document.

12.1. IANA Registration of imap: URI Scheme

   This section provides the information required to register the imap:
   URI scheme.

   URI scheme name: imap

   Status: permanent

   URI scheme syntax:

      See Section 11 of [RFC5092].

   URI scheme semantics:

      The imap: URI scheme is used to designate IMAP servers, mailboxes,
      messages, MIME bodies [MIME] and their parts, and search programs
      on Internet hosts accessible using the IMAP protocol.

      There is no MIME type associated with this URI.

   Encoding considerations:

      See Section 8 of [RFC5092].

   Applications/protocols that use this URI scheme name:

      The imap: URI is intended to be used by applications that might
      need access to an IMAP mailstore. Such applications may include
      (but are not limited to) IMAP-capable web browsers; IMAP clients
      that wish to access a mailbox, message, or edit a message on the
      server using [CATENATE]; [SUBMIT] clients and servers that are
      requested to assemble a complete message on submission using
      [BURL].

   Interoperability considerations:

      A widely deployed IMAP client Netscape Mail (and possibly
      Mozilla/Thunderbird/Seamonkey) uses a different imap: scheme
      internally.

Security considerations:

    See Security Considerations (Section 10) of [RFC5092].

Contact:

    Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller:

    IESG

References:

    [RFC5092] and [IMAP4].

13. References

13.1. Normative References

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

    [IMAP4]   Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION
                  4rev1", RFC 3501, March 2003.

    [IMAPABNF]   Melnikov, A. and C. Daboo, "Collected Extensions to
                  IMAP4 ABNF", RFC 4466, April 2006.

    [ABNF]   Crocker, D., Ed., and P. Overell, "Augmented BNF for
                  Syntax Specifications: ABNF", RFC 4234, October 2005.

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

    [URI-GEN]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
                  Resource Identifier (URI): Generic Syntax", STD 66, RFC
                  3986, January 2005.

    [UTF-8]   Yergeau, F., "UTF-8, a transformation format of ISO
                  10646", STD 63, RFC 3629, November 2003.

    [NAMESPACE]   Gahrns, M. and C. Newman, "IMAP4 Namespace", RFC 2342,
                  May 1998.

    [LITERAL+]   Myers, J., "IMAP4 non-synchronizing literals", RFC 2088,
                  January 1997.

   [ANONYMOUS]   Zeilenga, K., "Anonymous Simple Authentication and
                 Security Layer (SASL) Mechanism", RFC 4505, June 2006.

   [DATETIME]   Klyne, G. and C. Newman, "Date and Time on the Internet:
                 Timestamps", RFC 3339, July 2002.

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

13.2.   Informative References

   [SUBMIT]   Gellens, R. and J. Klensin, "Message Submission for
                 Mail", RFC 4409, April 2006.

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

   [CATENATE]   Resnick, P., "Internet Message Access Protocol (IMAP)
                 CATENATE Extension", RFC 4469, April 2006.

   [SASL]   Melnikov, A., Ed., and K. Zeilenga, Ed., "Simple
                 Authentication and Security Layer (SASL)", RFC 4422,
                 June 2006.

   [GSSAPI]   Melnikov, A., Ed., "The Kerberos V5 ("GSSAPI") Simple
                 Authentication and Security Layer (SASL) Mechanism", RFC
                 4752, November 2006.

   [DIGEST-MD5] Leach, P. and C. Newman, "Using Digest Authentication as
                 a SASL Mechanism", RFC 2831, May 2000.

   [URI-REG]   Hansen, T., Hardie, T., and L. Masinter, "Guidelines and
                 Registration Procedures for New URI Schemes", BCP 115,
                 RFC 4395, February 2006.

Appendix A.   Sample Code

    Here is sample C source code to convert between URL paths and IMAP
    mailbox names, taking into account mapping between IMAP's modified
    UTF-7 [IMAP4] and hex-encoded UTF-8, which is more appropriate for
    URLs. This code has not been rigorously tested nor does it
    necessarily behave reasonably with invalid input, but it should serve
    as a useful example. This code just converts the mailbox portion of
    the URL and does not deal with parameters, query, or server
    components of the URL.

```
/* Copyright (C) The IETF Trust (2007).  This version of
   sample C code is part of RFC XXXX; see the RFC itself
   for full legal notices.

   Regarding this sample C code (or any portion of it), the authors
   make no guarantees and are not responsible for any damage
   resulting from its use.  The authors grant irrevocable permission
   to anyone to use, modify, and distribute it in any way that does
   not diminish the rights of anyone else to use, modify, and
   distribute it, provided that redistributed derivative works do
   not contain misleading author or version information.

   Derivative works need not be licensed under similar terms.
 */

#include <stdio.h>
#include <string.h>

/* hexadecimal lookup table */
static const char hex[] = "0123456789ABCDEF";

#define XX 127
/*
 * Table for decoding hexadecimal in %encoding
 */
static const char index_hex[256] = {
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
     0, 1, 2, 3,  4, 5, 6, 7,  8, 9,XX,XX, XX,XX,XX,XX,
    XX,10,11,12, 13,14,15,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,10,11,12, 13,14,15,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
```

```
       XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,
       XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,
       XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,
       XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,
       XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,  XX,XX,XX,XX,
};
#define HEXCHAR(c)  (index_hex[(unsigned char)(c)])


/* "gen-delims" excluding "/" but including "%" */
#define GENERAL_DELIMS_NO_SLASH     ":?#[]@" "%"

/* "gen-delims" (excluding "/", but including "%")
   plus subset of "sub-delims" */
#define GENERAL_UNSAFE_NO_SLASH     GENERAL_DELIMS_NO_SLASH ";&=+"
#define OTHER_UNSAFE                " \"<>\\^`{|}"

/* URL unsafe printable characters */
static const char mailbox_url_unsafe[] = GENERAL_UNSAFE_NO_SLASH
                                         OTHER_UNSAFE;

/* UTF7 modified base64 alphabet */
static const char base64chars[] =
  "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,";


#define UNDEFINED 64

/* UTF16 definitions */
#define UTF16MASK    0x03FFUL
#define UTF16SHIFT   10
#define UTF16BASE    0x10000UL
#define UTF16HIGHSTART   0xD800UL
#define UTF16HIGHEND     0xDBFFUL
#define UTF16LOSTART     0xDC00UL
#define UTF16LOEND   0xDFFFUL


/* Convert an IMAP mailbox to a URL path
 *  dst needs to have roughly 4 times the storage space of src
 *     Hex encoding can triple the size of the input
 *     UTF-7 can be slightly denser than UTF-8
 *     (worst case: 8 octets UTF-7 becomes 9 octets UTF-8)
 */
void MailboxToURL(char *dst, char *src)
{
    unsigned char c, i, bitcount;
    unsigned long ucs4, utf16, bitbuf;
    unsigned char base64[256], utf8[6];

    /* initialize modified base64 decoding table */
```

```
    memset(base64, UNDEFINED, sizeof (base64));
    for (i = 0; i < sizeof (base64chars); ++i) {
     base64[(int) base64chars[i]] = i;
    }

    /* loop until end of string */
    while (*src != '\0') {
     c = *src++;
     /* deal with literal characters and &- */
     if (c != '&' || *src == '-') {
         /* NB: There are no "URL safe" characters after the '~' */
         if (c < ' ' || c > '~' ||
             strchr(mailbox_url_unsafe, c) != NULL) {
          /* hex encode if necessary */
          dst[0] = '%';
          dst[1] = hex[c >> 4];
          dst[2] = hex[c & 0x0f];
          dst += 3;
         } else {
          /* encode literally */
          *dst++ = c;
         }
         /* skip over the '-' if this is an &- sequence */
         if (c == '&') ++src;

     } else {
         /* convert modified UTF-7 -> UTF-16 -> UCS-4 -> UTF-8 -> HEX */
         bitbuf = 0;
         bitcount = 0;
         ucs4 = 0;
         while ((c = base64[(unsigned char) *src]) != UNDEFINED) {
          ++src;
          bitbuf = (bitbuf << 6) | c;
          bitcount += 6;
          /* enough bits for a UTF-16 character? */
          if (bitcount >= 16) {
              bitcount -= 16;
              utf16 = (bitcount ? bitbuf >> bitcount
                              : bitbuf) & 0xffff;
              /* convert UTF16 to UCS4 */
              if
                    (utf16 >= UTF16HIGHSTART && utf16 <= UTF16HIGHEND) {
               ucs4 = (utf16 - UTF16HIGHSTART) << UTF16SHIFT;
               continue;
              } else if
                    (utf16 >= UTF16LOSTART && utf16 <= UTF16LOEND) {
               ucs4 += utf16 - UTF16LOSTART + UTF16BASE;
              } else {
```

```
                 ucs4 = utf16;
               }
               /* convert UTF-16 range of UCS4 to UTF-8 */
               if (ucs4 <= 0x7fUL) {
                utf8[0] = (unsigned char) ucs4;
                i = 1;
               } else if (ucs4 <= 0x7ffUL) {
                utf8[0] = 0xc0 | (unsigned char) (ucs4 >> 6);
                utf8[1] = 0x80 | (unsigned char) (ucs4 & 0x3f);
                i = 2;
               } else if (ucs4 <= 0xffffUL) {
                utf8[0] = 0xe0 | (unsigned char) (ucs4 >> 12);
                utf8[1] = 0x80 | (unsigned char) ((ucs4 >> 6) & 0x3f);
                utf8[2] = 0x80 | (unsigned char) (ucs4 & 0x3f);
                i = 3;
               } else {
                utf8[0] = 0xf0 | (unsigned char) (ucs4 >> 18);
                utf8[1] = 0x80 | (unsigned char) ((ucs4 >> 12) & 0x3f);
                utf8[2] = 0x80 | (unsigned char) ((ucs4 >> 6) & 0x3f);
                utf8[3] = 0x80 | (unsigned char) (ucs4 & 0x3f);
                i = 4;
               }
               /* convert utf8 to hex */
               for (c = 0; c < i; ++c) {
                dst[0] = '%';
                dst[1] = hex[utf8[c] >> 4];
                dst[2] = hex[utf8[c] & 0x0f];
                dst += 3;
               }
           }
          }
          /* skip over trailing '-' in modified UTF-7 encoding */
          if (*src == '-') ++src;
       }
      }
     /* terminate destination string */
     *dst = '\0';
}

/* Convert hex coded UTF-8 URL path to modified UTF-7 IMAP mailbox
 *   dst should be about twice the length of src to deal with non-hex
 *   coded URLs
 */
int URLtoMailbox(char *dst, char *src)
{
    unsigned int utf8pos = 0;
    unsigned int utf8total, i, c, utf7mode, bitstogo, utf16flag;
    unsigned long ucs4 = 0, bitbuf = 0;
```

```
     utf7mode = 0; /* is the output UTF7 currently in base64 mode? */
     utf8total = 0; /* how many octets is the current input UTF-8 char;
                       0 == between characters */
     bitstogo = 0; /* bits that need to be encoded into base64; if
                      bitstogo != 0 then utf7mode == 1 */
     while ((c = (unsigned char)*src) != '\0') {
      ++src;
      /* undo hex-encoding */
      if (c == '%' && src[0] != '\0' && src[1] != '\0') {
          c = HEXCHAR(src[0]);
          i = HEXCHAR(src[1]);
          if (c == XX || i == XX) {
              return 0;
          } else {
              c = (char)((c << 4) | i);
          }
          src += 2;
      }
      /* normal character? */
      if (c >= ' ' && c <= '~') {
          /* switch out of UTF-7 mode */
          if (utf7mode) {
           if (bitstogo) {
           *dst++ = base64chars[(bitbuf << (6 - bitstogo)) & 0x3F];
           }
           *dst++ = '-';
           utf7mode = 0;
           bitstogo = bitbuf = 0;
          }
          *dst++ = c;
          /* encode '&' as '&-' */
          if (c == '&') {
           *dst++ = '-';
          }
          continue;
      }
      /* switch to UTF-7 mode */
      if (!utf7mode) {
          *dst++ = '&';
          utf7mode = 1;
      }
      /* Encode US-ASCII characters as themselves */
      if (c < 0x80) {
          ucs4 = c;
          utf8total = 1;
      } else if (utf8total) {
          /* this is a subsequent octet of a multi-octet character */
          /* save UTF8 bits into UCS4 */
```

```
            ucs4 = (ucs4 << 6) | (c & 0x3FUL);
            if (++utf8pos < utf8total) {
             continue;
            }
        } else {
            /* this is the first octet of a multi-octet character */
            utf8pos = 1;
            if (c < 0xE0) {
             utf8total = 2;
             ucs4 = c & 0x1F;
            } else if (c < 0xF0) {
             utf8total = 3;
             ucs4 = c & 0x0F;
            } else {
             /* NOTE: can't convert UTF8 sequences longer than 4 */
             utf8total = 4;
             ucs4 = c & 0x03;
            }
            continue;
        }
        /* Finished with UTF-8 character.  Make sure it isn't an
           overlong sequence.  If it is, return failure. */
        if ((ucs4 < 0x80 && utf8total > 1) ||
            (ucs4 < 0x0800 && utf8total > 2) ||
            (ucs4 < 0x00010000 && utf8total > 3) ||
            (ucs4 < 0x00200000 && utf8total > 4) ||
            (ucs4 < 0x04000000 && utf8total > 5) ||
            (ucs4 < 0x80000000 && utf8total > 6)) {
            return 0;
        }
        /* loop to split ucs4 into two utf16 chars if necessary */
        utf8total = 0;
        do {
            if (ucs4 >= UTF16BASE) {
                    ucs4 -= UTF16BASE;
             bitbuf = (bitbuf << 16) | ((ucs4 >> UTF16SHIFT)
                                + UTF16HIGHSTART);
             ucs4 = (ucs4 & UTF16MASK) + UTF16LOSTART;
             utf16flag = 1;
            } else {
             bitbuf = (bitbuf << 16) | ucs4;
             utf16flag = 0;
            }
            bitstogo += 16;
            /* spew out base64 */
            while (bitstogo >= 6) {
             bitstogo -= 6;
             *dst++ = base64chars[(bitstogo ? (bitbuf >> bitstogo)
```

```
                             : bitbuf)
                        & 0x3F];
         }
     } while (utf16flag);
    }
    /* if in UTF-7 mode, finish in ASCII */
    if (utf7mode) {
     if (bitstogo) {
         *dst++ = base64chars[(bitbuf << (6 - bitstogo)) & 0x3F];
     }
     *dst++ = '-';
    }
    /* tie off string */
    *dst = '\0';
    return 1;
}
```

Appendix B. List of Changes since RFC 2192

   Updated boilerplate, list of editor's, etc.
   Updated references.
   Updated ABNF not to use _, to use SP instead of SPACE, etc.
   Updated example domains to use example.org.
   Fixed ABNF error in "imessagelist" non-terminal.
   Updated ABNF, due to changes in RFC 3501, RFC 4466, and RFC 3986.
   Renamed "iuserauth" non-terminal to <iuserinfo>.
   Clarified that the userinfo component describes both authorization
   identity and mailbox naming scope.
   Allow for non-synchronizing literals in "enc-search".
   Added "ipartial" specifier that denotes a partial FETCH.
   Moved URLAUTH text from RFC 4467 to this document.
   Updated ABNF for the whole server to allow missing trailing "/"
   (e.g., "imap://imap.example.com" is now valid and is the same as
   "imap://imap.example.com/").
   Clarified how relative-path references are constructed.
   Added more examples demonstrating relative-path references.
   Added rules for relative URLs and restructured ABNF as the result.
   Removed text on use of relative URLs in MHTML.
   Added examples demonstrating security considerations when resolving
   URLs.
   Recommend usage of STARTTLS/SASL security layer to protect
   confidential data.
   Removed some advices about connection reuse that were incorrect.
   Removed URLs referencing a list of mailboxes, as this feature
   hasn't seen any deployments.
   Clarified that user name "anonymous" is case-insensitive.

Appendix C. List of Changes since RFC 4467

   Renamed <mechanism> to <uauth-mechanism>. Restructured ABNF.

Appendix D. Acknowledgments

   Text describing URLAUTH was lifted from [URLAUTH] by Mark Crispin.

   Stephane H. Maes contributed some ideas to this document; he also
   co-edited early versions of this document.

   The editors would like to thank Mark Crispin, Ken Murchison, Ted
   Hardie, Zoltan Ordogh, Dave Cridland, Kjetil Torgrim Homme, Lisa
   Dusseault, Spencer Dawkins, Filip Navara, Shawn M. Emery, Sam
   Hartman, Russ Housley, and Lars Eggert for the time they devoted to
   reviewing this document and/or for the comments received.

Authors' Addresses

   Chris Newman (Author/Editor)
   Sun Microsystems
   3401 Centrelake Dr., Suite 410
   Ontario, CA 91761
   EMail: chris.newman@sun.com

   Alexey Melnikov (Editor)
   Isode Limited
   5 Castle Business Village
   36 Station Road
   Hampton, Middlesex
   TW12 2BX, UK
   EMail: Alexey.Melnikov@isode.com
   URI: http://www.melnikov.ca/

Full Copyright Statement

Intellectual Property