Network Working Group
Request for Comments: 5255
Category: Standards Track

C. Newman
Sun Microsystems
A. Gulbrandsen
Oryx Mail Systems GmhH
A. Melnikov
Isode Limited

June 2008

Internet Message Access Protocol Internationalization

Status of This Memo

Abstract

Internet Message Access Protocol (IMAP) version 4rev1 has basic support for non-ASCII characters in mailbox names and search substrings. It also supports non-ASCII message headers and content encoded as specified by Multipurpose Internet Mail Extensions (MIME). This specification defines a collection of IMAP extensions that improve international support including language negotiation for international error text, translations for namespace prefixes, and comparator negotiation for search, sort, and thread.

Table of Contents

1. Introduction

This specification defines two IMAP4rev1 [RFC3501] extensions to enhance international support. These extensions can be advertised and implemented separately.

The LANGUAGE extension allows the client to request a suitable language for protocol error messages and in combination with the NAMESPACE extension [RFC2342] enables namespace translations.

The I18NLEVEL=2 extension allows the client to request a suitable collation that will modify the behavior of the base specification's SEARCH command as well as the SORT and THREAD extensions [SORT]. This leverages the collation registry [RFC4790]. The I18NLEVEL=1 extension updates SEARCH/SORT/THREAD to use i;unicode-casemap comparator, as defined in [UCM]. I18NLEVEL=1 is a simpler version of I18NLEVEL=2 with no ability to select a different collation.

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 [RFC2119].

The formal syntax uses the Augmented Backus-Naur Form (ABNF) [RFC5234] notation including the core rules defined in Appendix A.

The UTF-8-related productions are defined in [RFC3629].

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. If a single "C:" or "S:" label applies to multiple lines, then the line breaks between those lines are for editorial clarity only and are not part of the actual protocol exchange.

3. LANGUAGE Extension

IMAP allows server responses to include human-readable text that in many cases needs to be presented to the user. But that text is limited to US-ASCII by the IMAP specification [RFC3501] in order to preserve backwards compatibility with deployed IMAP implementations. This section specifies a way for an IMAP client to negotiate which language the server should use when sending human-readable text.

The LANGUAGE extension only provides a mechanism for altering fixed server strings such as response text and NAMESPACE folder names. Assigning localized language aliases to shared mailboxes would be done with a separate mechanism such as the proposed METADATA extension (see [METADATA]).

3.1. LANGUAGE Extension Requirements

IMAP servers that support this extension MUST list the keyword LANGUAGE in their CAPABILITY response as well as in the greeting CAPABILITY data.

A server that advertises this extension MUST use the language "i-default" as described in [RFC2277] as its default language until another supported language is negotiated by the client. A server MUST include "i-default" as one of its supported languages. IMAP servers SHOULD NOT advertise the LANGUAGE extension if they discover that they only support "i-default".

Clients and servers that support this extension MUST also support the NAMESPACE extension [RFC2342].

The LANGUAGE command is valid in all states. Clients SHOULD issue LANGUAGE before authentication, since some servers send valuable user information as part of authentication (e.g., "password is correct, but expired"). If a security layer (such as SASL or TLS) is subsequently negotiated by the client, it MUST re-issue the LANGUAGE command in order to make sure that no previous active attack (if any) on LANGUAGE negotiation has effect on subsequent error messages. (See Section 7 for a more detailed explanation of the attack.)

3.2. LANGUAGE Command

Arguments: Optional language range arguments.

Response: A possible LANGUAGE response (see Section 3.3).
A possible NAMESPACE response (see Section 3.4).

Result: OK - Command completed
NO - Could not complete command
BAD - Arguments invalid

The LANGUAGE command requests that human-readable text emitted by the server be localized to a language matching one of the language range argument as described by Section 2 of [RFC4647].

If the command succeeds, the server will return human-readable responses in the first supported language specified. These responses will be in UTF-8 [RFC3629]. The server MUST send a LANGUAGE response specifying the language used, and the change takes effect immediately after the LANGUAGE response.

If the command fails, the server continues to return human-readable responses in the language it was previously using.

The special "default" language range argument indicates a request to use a language designated as preferred by the server administrator. The preferred language MAY vary based on the currently active user.

If a language range does not match a known language tag exactly but does match a language by the rules of [RFC4647], the server MUST send an untagged LANGUAGE response indicating the language selected.

If there aren't any arguments, the server SHOULD send an untagged LANGUAGE response listing the languages it supports. If the server is unable to enumerate the list of languages it supports it MAY return a tagged NO response to the enumeration request. If, after receiving a LANGUAGE request, the server discovers that it doesn't support any language other than i-default, it MUST return a tagged NO response to the enumeration request.

```
   < The server defaults to using English i-default responses until
     the user explicitly changes the language. >

   C: A001 LOGIN KAREN PASSWORD
   S: A001 OK LOGIN completed

   < Client requested MUL language, which no server supports. >

   C: A002 LANGUAGE MUL
   S: A002 NO Unsupported language MUL

   < A LANGUAGE command with no arguments is a request to enumerate
     the list of languages the server supports. >

   C: A003 LANGUAGE
   S: * LANGUAGE (EN DE IT i-default)
   S: A003 OK Supported languages have been enumerated

   C: B001 LANGUAGE
   S: B001 NO Server is unable to enumerate supported languages
```

```
      < Once the client changes the language, all responses will be in
        that language starting after the LANGUAGE response.  Note that
        this includes the NAMESPACE response.  Because RFCs are in US-
        ASCII, this document uses an ASCII transcription rather than
        UTF-8 text, e.g., "ue" in the word "ausgefuehrt" >

      C: C001 LANGUAGE DE
      S: * LANGUAGE (DE)
      S: * NAMESPACE (("" "/")) (("Other Users/" "/" "TRANSLATION"
            ("Andere Ben&APw-tzer/"))) (("Public Folders/" "/"
            "TRANSLATION" ("Gemeinsame Postf&AM8-cher/")))
      S: C001 OK Sprachwechsel durch LANGUAGE-Befehl ausgefuehrt

      < If a server does not support the requested primary language,
        responses will continue to be returned in the current language
        the server is using. >

      C: D001 LANGUAGE FR
      S: D001 NO Diese Sprache ist nicht unterstuetzt
      C: D002 LANGUAGE DE-IT
      S: * LANGUAGE (DE-IT)
      S: * NAMESPACE (("" "/"))(("Other Users/" "/" "TRANSLATION"
            ("Andere Ben&APw-tzer/"))) (("Public Folders/" "/"
            "TRANSLATION" ("Gemeinsame Postf&AM8-cher/")))
      S: D002 OK Sprachwechsel durch LANGUAGE-Befehl ausgefuehrt
      C: D003 LANGUAGE "default"
      S: * LANGUAGE (DE)
      S: D003 OK Sprachwechsel durch LANGUAGE-Befehl ausgefuehrt

      < Server does not speak French, but does speak English.  User
        speaks Canadian French and Canadian English. >

      C: E001 LANGUAGE FR-CA EN-CA
      S: * LANGUAGE (EN)
      S: E001 OK Now speaking English
```

3.3. LANGUAGE Response

Contents: A list of one or more language tags.

The LANGUAGE response occurs as a result of a LANGUAGE command. A LANGUAGE response with a list containing a single language tag indicates that the server is now using that language. A LANGUAGE response with a list containing multiple language tags indicates the server is communicating a list of available languages to the client, and no change in the active language has been made.

3.4. TRANSLATION Extension to the NAMESPACE Response

If localized representations of the namespace prefixes are available in the selected language, the server SHOULD include these in the TRANSLATION extension to the NAMESPACE response.

The TRANSLATION extension to the NAMESPACE response returns a single string, containing the modified UTF-7 [RFC3501] encoded translation of the namespace prefix. It is the responsibility of the client to convert between the namespace prefix and the translation of the namespace prefix when presenting mailbox names to the user.

In this example, a server supports the IMAP4 NAMESPACE command. It uses no prefix to the user's Personal Namespace, a prefix of "Other Users" to its Other Users' Namespace, and a prefix of "Public Folders" to its only Shared Namespace. Since a client will often display these prefixes to the user, the server includes a translation of them that can be presented to the user.

```
C: A001 LANGUAGE DE-IT
S: * NAMESPACE (("" "/")) (("Other Users/" "/" "TRANSLATION"
      ("Andere Ben&APw-tzer/"))) (("Public Folders/" "/"
      "TRANSLATION" ("Gemeinsame Postf&AM8-cher/")))
S: A001 OK LANGUAGE-Befehl ausgefuehrt
```

3.5. Formal Syntax

The following syntax specification inherits ABNF [RFC5234] rules from IMAP4rev1 [RFC3501], IMAP4 Namespace [RFC2342], Tags for the Identifying Languages [RFC4646], UTF-8 [RFC3629], and Collected Extensions to IMAP4 ABNF [RFC4466].

```
command-any       =/ language-cmd
    ; LANGUAGE command is valid in all states

language-cmd      = "LANGUAGE" *(SP lang-range-quoted)

response-payload  =/ language-data

language-data     = "LANGUAGE" SP "(" lang-tag-quoted *(SP
                    lang-tag-quoted) ")"

namespace-trans   = SP DQUOTE "TRANSLATION" DQUOTE SP "(" string ")"
    ; the string is encoded in Modified UTF-7.
    ; this is a subset of the syntax permitted by
    ; the Namespace-Response-Extension rule in [RFC4466]

lang-range-quoted = astring
    ; Once any literal wrapper or quoting is removed, this
    ; follows the language-range rule in [RFC4647]

lang-tag-quoted   = astring
    ; Once any literal wrapper or quoting is removed, this follows
    ; the Language-Tag rule in [RFC4646]

resp-text         = ["[" resp-text-code "]" SP ] UTF8-TEXT-CHAR
                      *(UTF8-TEXT-CHAR / "[")
    ; After the server is changed to a language other than
    ; i-default, this resp-text rule replaces the resp-text
    ; rule from [RFC3501].

UTF8-TEXT-CHAR    = %x20-5A / %x5C-7E / UTF8-2 / UTF8-3 / UTF8-4
    ; UTF-8 excluding 7-bit control characters and "["
```

4. I18NLEVEL=1 and I18NLEVEL=2 Extensions

4.1. Introduction and Overview

IMAP4rev1 [RFC3501] includes the SEARCH command that can be used to locate messages matching criteria including human-readable text. The SORT extension [SORT] to IMAP allows the client to ask the server to determine the order of messages based on criteria including human-readable text. These mechanisms require the ability to support non-English search and sort functions.

Section 4 defines two IMAP extensions for internationalizing IMAP SEARCH, SORT, and THREAD [SORT] using the comparator framework [RFC4790].

The I18NLEVEL=1 extension updates SEARCH/SORT/THREAD to use i;unicode-casemap comparator, as defined in [UCM]. See Sections 4.2 and 4.3 for more details.

The I18NLEVEL=2 extension is a superset of the I18NLEVEL=1 extension. It adds to I18NLEVEL=1 extension the ability to determine the active comparator (see definition below) and to negotiate use of comparators using the COMPARATOR command. It also adds the COMPARATOR response that indicates the active comparator and possibly other available comparators. See Sections 4.2 and 4.4 for more details.

4.2. Requirements Common to Both I18NLEVEL=1 and I18NLEVEL=2

The term "default comparator" refers to the comparator that is used by SEARCH and SORT absent any negotiation using the COMPARATOR command (see Section 4.7). The term "active comparator" refers to the comparator which will be used within a session, e.g., by SEARCH and SORT. The COMPARATOR command is used to change the active comparator.

The active comparator applies to the following SEARCH keys: "BCC", "BODY", "CC", "FROM", "SUBJECT", "TEXT", "TO", and "HEADER". If the server also advertises the "SORT" extension, then the active comparator applies to the following SORT keys: "CC", "FROM", "SUBJECT", and "TO". If the server advertises THREAD=ORDEREDSUBJECT, then the active comparator applies to the ORDEREDSUBJECT threading algorithm. If the server advertises THREAD=REFERENCES, then the active comparator applies to the subject field comparisons done by REFERENCES threading algorithm. Future extensions may choose to apply the active comparator to their SEARCH keys.

For SORT and THREAD, the pre-processing necessary to extract the base subject text from a Subject header occurs prior to the application of a comparator.

A server that advertises I18NLEVEL=1 or I18NLEVEL=2 extension MUST implement the i;unicode-casemap comparator, as defined in [UCM].

A server that advertises I18NLEVEL=1 or I18NLEVEL=2 extension MUST support UTF-8 as a SEARCH charset.

4.3. I18NLEVEL=1 Extension Requirements

An IMAP server that satisfies all requirements specified in Sections 4.2 and 4.6 (and that doesn't support/advertise any other I18NLEVEL=<n> extension, where n > 1) MUST list the keyword I18NLEVEL=1 in its CAPABILITY data once IMAP enters the authenticated state, and MAY list that keyword in other states.

4.4. I18NLEVEL=2 Extension Requirements

An IMAP server that satisfies all requirements specified in Sections 4.2, 4.4, and 4.6-4.10 (and that doesn't support/advertise any other I18NLEVEL=<n> extension, where n > 2) MUST list the keyword I18NLEVEL=2 in its CAPABILITY data once IMAP enters the authenticated state, and MAY list that keyword in other states.

A server that advertises this extension MUST implement the i;unicode-casemap comparator, as defined in [UCM]. It MAY implement other comparators from the IANA registry established by [RFC4790]. See also Section 4.5 of this document.

A server that advertises this extension SHOULD use i;unicode-casemap as the default comparator. (Note that i;unicode-casemap is the default comparator for I18NLEVEL=1, but not necessarily the default for I18NLEVEL=2.) The selection of the default comparator MAY be adjustable by the server administrator, and MAY be sensitive to the current user. Once the IMAP connection enters authenticated state, the default comparator MUST remain static for the remainder of that connection.

Note that since SEARCH uses the substring operation, IMAP servers can only implement collations that offer the substring operation (see [RFC4790], Section 4.2.2). Since SORT uses the ordering operation (which in turn uses the equality operation), IMAP servers that advertise the SORT extension can only implement collations that offer all three operations (see [RFC4790], Sections 4.2.2-4.2.4).

If the active collation does not provide the operations needed by an IMAP command, the server MUST respond with a tagged BAD.

4.5. Compatibility Notes

Several server implementations deployed prior to the publication of this specification comply with I18NLEVEL=1 (see Section 4.3), but do not advertise that. Other legacy servers use the i;ascii-casemap comparator (see [RFC4790]).

There is no good way for a client to know which comparator a legacy server uses. If the client has to assume the worst, it may end up doing expensive local operations to obtain i;unicode-casemap comparisons even though the server implements it.

Legacy server implementations which comply with I18NLEVEL=1 should be updated to advertise I18NLEVEL=1. All server implementations should eventually be updated to comply with the I18NLEVEL=2 extension.

4.6. Comparators and Character Encodings

RFC 3501, Section 6.4.4, says:

> In all search keys that use strings, a message matches the key if the string is a substring of the field. The matching is case-insensitive.

When performing the SEARCH operation, the active comparator is applied instead of the case-insensitive matching specified above.

An IMAP server which performs collation operations (e.g., as part of commands such as SEARCH, SORT, and THREAD) does so according to the following procedure:

(a) MIME encoding (for example, see [RFC2047] for headers and [RFC2045] for body parts) MUST be removed in the texts being collated.

    If MIME encoding removal fails for a message (e.g., a body part of the message has an unsupported Content-Transfer-Encoding, uses characters not allowed by the Content-Transfer-Encoding, etc.), the collation of this message is undefined by this specification, and is handled in an implementation-dependent manner.

(b) The decoded text from (a) MUST be converted to the charset expected by the active comparator.

(c) For the substring operation:

If step (b) failed (e.g., the text is in an unknown charset, contains a sequence that is not valid according in that charset, etc.), the original decoded text from (a) (i.e., before the charset conversion attempt) is collated using the i;octet comparator (see [RFC4790]).

If step (b) was successful, the converted text from (b) is collated according to the active comparator.

For the ordering operation:

All strings that were successfully converted by step (b) are separated from all strings that failed step (b). Strings in each group are collated independently. All strings successfully converted by step (b) are then validated by the active comparator. Strings that pass validation are collated using the active comparator. All strings that either fail step (b) or fail the active collation's validity operation are collated (after applying step (a)) using the i;octet comparator (see [RFC4790]). The resulting sorted list is produced by appending all collated "failed" strings after all strings collated using the active comparator.

Example: The following example demonstrates ordering of 4 different strings using the i;unicode-casemap [UCM] comparator. Strings are represented using hexadecimal notation used by ABNF [RFC5234].

(1) %xD0 %xC0 %xD0 %xBD %xD0 %xB4 %xD1 %x80 %xD0 %xB5 %xD0 %xB9 (labeled with charset=UTF-8)
(2) %xD1 %x81 %xD0 %x95 %xD0 %xA0 %xD0 %x93 %xD0 %x95 %xD0 %x99 (labeled with charset=UTF-8)
(3) %xD0 %x92 %xD0 %xB0 %xD1 %x81 %xD0 %xB8 %xD0 %xBB %xD0 %xB8 %xFF %xB9 (labeled with charset=UTF-8)
(4) %xE1 %xCC %xC5 %xCB %xD3 %xC5 %xCA (labeled with charset=KOI8-R)

Step (b) will convert string (4) to the following sequence of octets (in UTF-8):

%xD0 %x90 %xD0 %xBB %xD0 %xB5 %xD0 %xBA %xD1 %x81 %xD0 %xB5 %xD0 %xB9

and will reject strings (1) and (3), as they contain octets not allowed in charset=UTF-8.

After that, using the i;unicode-casemap collation, string (4) will collate before string (2). Using the i;octet collation on the original strings, string (3) will collate before string (1). So the final ordering is as follows: (4) (2) (3) (1).

If the substring operation (e.g., IMAP SEARCH) of the active comparator returns the "undefined" result (see Section 4.2.3 of [RFC4790]) for either the text specified in the SEARCH command or the message text, then the operation is repeated on the result of step (a) using the i;octet comparator.

The ordering operation (e.g., IMAP SORT and THREAD) SHOULD collate the following together: strings encoded using unknown or invalid character encodings, strings in unrecognized charsets, and invalid input (as defined by the active collation).

4.7. COMPARATOR Command

Arguments: Optional comparator order arguments.

Response: A possible COMPARATOR response (see Section 4.8).

Result: OK - Command completed
NO - No matching comparator found
BAD - Arguments invalid

The COMPARATOR command is valid in authenticated and selected states.

The COMPARATOR command is used to determine or change the active comparator. When issued with no arguments, it results in a COMPARATOR response indicating the currently active comparator.

When issued with one or more comparator arguments, it changes the active comparator as directed. (If more than one installed comparator is matched by an argument, the first argument wins.) The COMPARATOR response lists all matching comparators if more than one matches the specified patterns.

The argument "default" refers to the server's default comparator. Otherwise, each argument is a collation specification as defined in the Internet Application Protocol Comparator Registry [RFC4790].

< The client requests activating a Czech comparator if possible, or else a generic international comparator which it considers suitable for Czech. The server picks the first supported comparator. >

```
C: A001 COMPARATOR "cz;*" i;basic
S: * COMPARATOR i;basic
S: A001 OK Will use i;basic for collation
```

4.8. COMPARATOR Response

Contents: The active comparator. An optional list of available matching comparators

The COMPARATOR response occurs as a result of a COMPARATOR command. The first argument in the comparator response is the name of the active comparator. The second argument is a list of comparators which matched any of the arguments to the COMPARATOR command and is present only if more than one match is found.

4.9. BADCOMPARATOR Response Code

This response code SHOULD be returned as a result of server failing an IMAP command (returning NO), when the server knows that none of the specified comparators match the requested comparator(s).

4.10. Formal Syntax

The following syntax specification inherits ABNF [RFC5234] rules from IMAP4rev1 [RFC3501] and the Internet Application Protocol Comparator Registry [RFC4790].

```
command-auth      =/ comparator-cmd

resp-text-code    =/ "BADCOMPARATOR"

comparator-cmd    = "COMPARATOR" *(SP comp-order-quoted)

response-payload  =/ comparator-data

comparator-data   = "COMPARATOR" SP comp-sel-quoted [SP "("
                    comp-id-quoted *(SP comp-id-quoted) ")"]

comp-id-quoted    = astring
    ; Once any literal wrapper or quoting is removed, this
    ; follows the collation-id rule from [RFC4790]

comp-order-quoted = astring
    ; Once any literal wrapper or quoting is removed, this
    ; follows the collation-order rule from [RFC4790]
```

```
comp-sel-quoted   = astring
    ; Once any literal wrapper or quoting is removed, this
    ; follows the collation-selected rule from [RFC4790]
```

5. Other IMAP Internationalization Issues

The following sections provide an overview of various other IMAP internationalization issues. These issues are not resolved by this specification, but could be resolved by other standards work, such as that being done by the EAI working group (see [IMAP-EAI]).

5.1. Unicode Userids and Passwords

IMAP4rev1 currently restricts the userid and password fields of the LOGIN command to US-ASCII. The "userid" and "password" fields of the IMAP LOGIN command are restricted to US-ASCII only until a future standards track RFC states otherwise. Servers are encouraged to validate both fields to make sure they conform to the formal syntax of UTF-8 and to reject the LOGIN command if that syntax is violated. Servers MAY reject the LOGIN command if either the "userid" or "password" field contains an octet with the highest bit set.

When AUTHENTICATE is used, some servers may support userids and passwords in Unicode [RFC3490] since SASL (see [RFC4422]) allows that. However, such userids cannot be used as part of email addresses.

5.2. UTF-8 Mailbox Names

The modified UTF-7 mailbox naming convention described in Section 5.1.3 of RFC 3501 is best viewed as an transition from the status quo in 1996 when modified UTF-7 was first specified. At that time, there was widespread unofficial use of local character sets such as ISO-8859-1 and Shift-JIS for non-ASCII mailbox names, with resultant non-interoperability.

The requirements in Section 5.1 of RFC 3501 are very important if we’re ever going to be able to deploy UTF-8 mailbox names. Servers are encouraged to enforce them.

5.3. UTF-8 Domains, Addresses, and Mail Headers

There is now an IETF standard for "Internationalizing Domain Names in Applications (IDNA)" [RFC3490]. While IMAP clients are free to support this standard, an argument can be made that it would be helpful to simple clients if the IMAP server could perform this conversion (the same argument would apply to MIME header encoding

[RFC2047]). However, it would be unwise to move forward with such work until the work in progress to define the format of international email addresses is complete.

6. IANA Considerations

IANA added LANGUAGE, I18NLEVEL=1, and I18NLEVEL=2 to the IMAP4 Capabilities Registry.

7. Security Considerations

The LANGUAGE extension makes a new command available in "Not Authenticated" state in IMAP. Some IMAP implementations run with root privilege when the server is in "Not Authenticated" state and do not revoke that privilege until after authentication is complete. Such implementations are particularly vulnerable to buffer overflow security errors at this stage and need to implement parsing of this command with extra care.

A LANGUAGE command issued prior to activation of a security layer is subject to an active attack that suppresses or modifies the negotiation, and thus makes STARTTLS or authentication error messages more difficult to interpret. This is not a new attack as the error messages themselves are subject to active attack. Clients MUST re-issue the LANGUAGE command once a security layer is active, in order to prevent this attack from impacting subsequent protocol operations.

LANGUAGE, I18NLEVEL=1, and I18NLEVEL=2 extensions use the UTF-8 charset; thus, the security considerations for UTF-8 [RFC3629] are relevant. However, neither uses UTF-8 for identifiers, so the most serious concerns do not apply.

8. Acknowledgements

The LANGUAGE extension is based on a previous document by Mike Gahrns, a substantial portion of the text in that section was written by him. Many people have participated in discussions about an IMAP Language extension in the various fora of the IETF and Internet working groups, so any list of contributors is bound to be incomplete. However, the authors would like to thank Andrew McCown for early work on the original proposal, John Myers for suggestions regarding the namespace issue, along with Jutta Degener, Mark Crispin, Mark Pustilnik, Larry Osterman, Cyrus Daboo, Martin Duerst, Timo Sirainen, Ben Campbell, and Magnus Nystrom for their many suggestions that have been incorporated into this document.

Initial discussion of the I18NLEVEL=2 extension involved input from Mark Crispin and other participants of the IMAP Extensions WG.

9. Relevant Sources of Documents for Internationalized IMAP Implementations

This is a non-normative list of sources to consider when implementing i18n-aware IMAP software.

- The LANGUAGE and I18NLEVEL=2 extensions to IMAP (this specification).
- The 8-bit rules for mailbox naming in Section 5.1 of RFC 3501.
- The Mailbox International Naming Convention in Section 5.1.3 of RFC 3501.
- MIME [RFC2045] for message bodies.
- MIME header encoding [RFC2047] for message headers.
- The IETF EAI working group.
- MIME Parameter Value and Encoded Word Extensions [RFC2231] for filenames. Quality IMAP server implementations will automatically combine multipart parameters when generating the BODYSTRUCTURE. There is also some deployed non-standard use of MIME header encoding inside double quotes for filenames.
- IDNA [RFC3490] and punycode [RFC3492] for domain names (currently only relevant to IMAP clients).
- The UTF-8 charset [RFC3629].
- The IETF policy on Character Sets and Languages [RFC2277].

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2342] Gahrns, M. and C. Newman, "IMAP4 Namespace", RFC 2342, May 1998.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC4422] Melnikov, A., Ed., and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC4466] Melnikov, A. and C. Daboo, "Collected Extensions to IMAP4 ABNF", RFC 4466, April 2006.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

[RFC4647] Phillips, A. and M. Davis, "Matching of Language Tags", BCP 47, RFC 4647, September 2006.

[RFC4790] Newman, C., Duerst, M., and A. Gulbrandsen, "Internet Application Protocol Collation Registry", RFC 4790, March 2007.

[SORT] Crispin, M. and K. Murchison, "Internet Message Access Protocol - SORT and THREAD Extensions", RFC 5256, June 2008.

[UCM] Crispin, M., "i;unicode-casemap - Simple Unicode Collation Algorithm", RFC 5051, October 2007.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

11. Informative References

[RFC2231] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, November 1997.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[METADATA] Daboo, C., "IMAP METADATA Extension", Work in Progress, April 2008.

[IMAP-EAI] Resnick, P., and C. Newman, "IMAP Support for UTF-8", Work in Progress, November 2007.

Authors' Addresses

Chris Newman
Sun Microsystems
3401 Centrelake Dr., Suite 410
Ontario, CA 91761
US

EMail: chris.newman@sun.com

Arnt Gulbrandsen
Oryx Mail Systems GmbH
Schweppermannstr. 8
D-81671 Muenchen
Germany

EMail: arnt@oryx.com
Fax: +49 89 4502 9758

Alexey Melnikov
Isode Limited
5 Castle Business Village, 36 Station Road,
Hampton, Middlesex, TW12 2BX, UK

EMail: Alexey.Melnikov@isode.com

Full Copyright Statement

Intellectual Property