Network Working Group
Internet-Draft
Expires: November 7, 2003

C. Newman
Sun Microsystems

May 9, 2003

Internet Message Access Protocol Internationalization
draft-ietf-imapext-i18n-00.txt

Status of this Memo

Copyright Notice

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 which improve international support including comparator negotiation for search, sort and thread, language negotiation for international error text, and translations for namespace prefixes.

Table of Contents

1. Conventions Used in this Document

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as defined in "Key words for use in RFCs to Indicate Requirement Levels" [1].

The formal syntax use the Augmented Backus-Naur Form (ABNF) [2] notation including the core rules defined in Appendix A of RFC 2234.

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.

2. Introduction

This specification defines two IMAP4rev1 [6] 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 [4] enables namespace translations.

The COMPARATOR extension allows the client to request a suitable comparator which will modify the behavior of the base specification's SEARCH command as well as the SORT and THREAD extensions [15]. This leverages the comparator registry [8].

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 [6] 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.

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 [3] 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.

A client that supports this extension MUST be prepared for a possible NAMESPACE response [4] from the server.

The LANGUAGE command is valid in not-authenticated, authenticated and selected states.

3.2 LANGUAGE Command

```
Arguments: Optional language range argument.

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

           A possible NAMESPACE response as defined by [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 the language matching the language range argument as described by section 2.5 of RFC 3066.

If the command succeeds, the server will return human-readable responses in the specified language starting with the tagged OK response to the LANGUAGE command. These responses will be in UTF-8 [7].

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

The client MUST NOT use MUL (Multiple languages) or UND (Undetermined) language tags and the server MUST return BAD if either tag is used. The special "*" 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 the language range does not match a known language tag exactly but does match a language by the rules of section 2.5 of [5], the server MUST send an untagged LANGUAGE response indicating the language selected.

If the language range argument is omitted, 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.

```
      < 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. Server MUST reply with BAD >

      C: A002 LANGUAGE MUL
      S: A002 BAD Invalid 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 with the tagged OK to the LANGUAGE
      command. >

      C: A004 LANGUAGE FR
      S: A004 OK La Language commande a ete execute avec success

      < 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: A005 LANGUAGE DE
      S: A005 NO Ce Language n'est pas supporte

      C: A006 LANGUAGE FR-CA
      S: * LANGUAGE (FR)
      S: A006 OK La Language commande a ete execute avec success

      C: A007 LANGUAGE "*"
      S: * LANGUAGE (FR)
      S: A007 OK La Language commande a ete execute avec success
```

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 the server supports the IMAP4 NAMESPACE command [4], the server MUST return an untagged NAMESPACE response when a language is negotiated. However the server MUST NOT return a NAMESPACE response if it is in not-authenticated state.

If as a result of the newly negotiated language, localized representations of the namespace prefixes are available, 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 [6] 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 FR-CA
S: * NAMESPACE (("" "/"))(("Other Users/" "/" "TRANSLATION"
     ("Autres Utilisateurs/"))) (("Public Folders/" "/"
     "TRANSLATION" ("R&Aok-pertoires Publics/")))
S: A001 OK La Language commande a ete executee avec success
```

3.5 Formal Syntax

The following syntax specification inherits ABNF [2] rules from IMAP4rev1 [6], IMAP4 Namespace [4], Tags for the Identification of Languages [5], and UTF-8 [7].

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

   mailbox-data    =/ language-data

   language-cmd    = "LANGUAGE" [SP lang-range-quoted]

   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 RFC 2342

   lang-range-quoted = astring
      ; Once any literal wrapper or quoting is removed, this follows
      ; the language-range rule in section 2.5 of RFC 3066

   lang-tag-quoted = astring
      ; Once any literal wrapper or quoting is removed, this follows
      ; the Language-Tag rule in section 2.1 of RFC 3066


   ; After the server is changed to a language other than i-default,
   ; the resp-text rule from RFC 3501 is replaced with the following:

   resp-text       = ["[" resp-text-code "]" SP ]
                     UTF8-TEXT-CHAR *(UTF8-TEXT-CHAR / "[")

   UTF8-TEXT-CHAR  = %x20-%x5A / %x5C-%x7E / UTF8-2 / UTF8-3 / UTF8-4
      ; UTF-8 excluding 7-bit control characters and "["
```

4. COMPARATOR Extension

IMAP4rev1 [6] includes the SEARCH command which can be used to locate messages matching criteria including human-readable text. The SORT extension [15] 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.

This section defines an IMAP extension to negotiate use of comparators [8] to internationalize IMAP SEARCH, SORT and THREAD. The IMAP extension consists of a new command to determine or change the active comparator, a new response to indicate the active comparator and possibly other available comparators, SEARCH and SORT keys which can be used to change comparators on-the-fly, and an additional response code to indicate that the failure of a SEARCH or SORT command was due to an invalid comparator.

The term "default comparator" refers to the comparator which is used by SEARCH and SORT absent any negotiation using the COMPARATOR command or SEARCH/SORT key. The term "active comparator" refers to the comparator which will be used within a session by SEARCH and SORT absent use of the COMPARATOR SEARCH/SORT key. The COMPARATOR command is used to change the active comparator. The term "selected comparator" refers to the comparator selected by the most recent COMPARATOR SEARCH/SORT key within the context of the current SEARCH/SORT program or the active comparator if there is no COMPARATOR SEARCH/SORT key yet seen in context.

The selected 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 selected comparator applies to the following SORT keys: "CC", "FROM", "SUBJECT" and "TO". If the server advertises the THREAD=ORDEREDSUBJECT, then the active comparator applies to the ORDEREDSUBJECT threading algorithm.

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.

4.1 COMPARATOR Extension Requirements

IMAP servers that support this extension MUST list the keyword COMPARATOR in their CAPABILITY data once IMAP enters authenticated state.

A server that advertises this extension MUST implement the en;ascii-casemap and i;octet comparators. A server intended to be deployed globally MUST implement the i;basic-uca=3.1.1-uv=3.2 comparator.

A server that advertises this extension MUST use a registered case-insensitive comparator which supports the substring matching function as the default comparator. If the server also advertises the SORT or THREAD=ORDEREDSUBJECT extensions, then the default comparator MUST also support the ordering function. 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.

A server that advertises this extension MUST support UTF-8 as a SEARCH charset.

The COMPARATOR command is valid in authenticated and selected states.

4.2 Comparators and Charsets

For SEARCH, SORT and THREAD operations that apply to message headers, the server is responsible for removing the MIME header encoding [10] and converting the text of any known charsets to UTF-8 prior to applying the comparator algorithm. Unknown charsets should never match when using the SEARCH command, and will sort together with invalid comparator input for the SORT and THREAD commands.

When message text is in a known charset other than UTF-8, the server is responsible for converting that text to UTF-8 prior to applying the comparator. When message text is in an unknown charset, then the text should be skipped by the SEARCH command unless the comparator is i;octet.

4.3 COMPARATOR Command

```
Arguments: Optional comparator order arguments.

Response:  A possible COMPARATOR response (see Section 4.4).

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

The COMPARATOR command is used to determine or change the active comparator. When issued with no arguments, it will result in a COMPARATOR response indicating the currently active comparator. When issued with one or more comparator order argument, it will change the active comparator if any comparator matches any argument. The COMPARATOR response will list other matching comparators if more than one matches the specified patterns.

When the single argument "*" is used with the COMPARATOR command, it will set the active comparator to be the default comparator.

4.4 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.

4.5 COMPARATOR SEARCH and SORT Key

The COMPARATOR SEARCH key takes a comparator order argument. That argument will select the comparator to use for subsequent SEARCH keys in the search program. The COMPARATOR SORT key works in a similar fashion except that it applies to subsequent SORT keys in the SORT criterion.

If no comparator matches the pattern specified by the COMPARATOR SEARCH or SORT key, then the SEARCH or SORT command will fail with a [BADCOMPARATOR] response code. This error code is also returned if a comparator is found, but it does not support the necessary function (substring matching for SEARCH, or ordering for SORT).

If an input string provided as part of a SEARCH program causes an error when used with the selected comparator, the SEARCH command will fail with a [BADMATCH] response code.

4.6 Formal Syntax

The following syntax specification inherits ABNF [2] rules from IMAP4rev1 [6], and Internet Application Protocol Comparator Registry [8].

```
   command-auth      =/ comparator-cmd

   mailbox-data      =/ comparator-data

   search-key        =/ comparator-key

   sort-criterion    =/ comparator-key
      ; this only applies to servers which advertise both
      ; the COMPARATOR and SORT extensions.

   resp-text-code    =/ "BADCOMPARATOR" / "BADMATCH"

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

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

   comparator-key    = "COMPARATOR" SP comp-order-quoted

   comp-name-quoted  = astring
      ; Once any literal wrapper or quoting is removed,
      ; this follows the comparator-name rule

   comp-order-quoted = astring
      ; Once any literal wrapper or quoting is removed,
      ; this follows the comparator-order rule

   comp-sel-quoted   = astring
      ; Once any literal wrapper or quoting is removed,
      ; this follows the comparator-sel rule
```

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 future standards work.

5.1 UTF-8 Userids and Passwords

IMAP4rev1 presently restricts the userid and password fields of the LOGIN command to US-ASCII. Because the ability to enter a userid and

password is necessary to use IMAP at all for most authentication mechanisms, the potential lack of input methods for non-ASCII text is a serious interoperability concern. However, because of the visibility of these fields to end-users, it is expected that pressure to support UTF-8 login names and passwords will eventually become irresistable. This specification defers such work until the SASL-related profile of stringprep [12] is published as an RFC, and the impact on ACLs and email addresses has been assessed.

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 use of any 8-bit in the "userid" or "password" field.

5.2 UTF-8 Mailbox Names

There appears to be rough concensus in the IMAP community that the modified UTF-7 mailbox naming convention was a mistake and we should have used UTF-8 instead. However, the preliminary design discussions to create a transitional mechanism for UTF-8 mailbox names suggests that the cost of supporting both a UTF-8 mechanism and a modified UTF-7 convention for mailbox names during a transition period might exceed the benefit of the eventual goal. Because of this controversy, UTF-8 mailbox name mechanisms are not included in this specification.

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.

5.3 UTF-8 Domains, Addresses and Mail Headers

There is now an IETF standard for Internationalizing Domain Names in Applications [13]. While IMAP clients are free to support this standard and convert punycode [14] to UTF-8 at display time, 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 [10]). 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

When this is published as an RFC, the following IMAP extensions will be registered:

Capability Name	Reference
LANGUAGE	[RFC XXXX]
COMPARATOR	[RFC XXXX]

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 which 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, so this does not impact subsequent protocol operations.

Both the LANGUAGE and COMPARATOR extensions use the UTF-8 charset, thus the security considerations for UTF-8 [7] are relevent. 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 Internet draft by Mike Gahrns and Alexey Melnikov, a substantial portion of the text in that section was written by them. 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 and Larry Osterman for their many suggestions that have been incorporated into this document.

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

9. Relevant Standards for i18n IMAP Implementations

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

- o The LANGUAGE and COMPARATOR extensions to IMAP (this specification).
- o The 8-bit rules for mailbox naming in section 5.1 of RFC 3501.
- o The Mailbox International Naming Convention in section 5.1.3 of RFC 3501.
- o MIME [9] for message bodies.
- o MIME header encoding [10] for message headers.
- o MIME Parameter Value and Encoded Word Extensions [11] 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.
- o IDNA [13] and punycode [14] for domain names (presently only relevant to IMAP clients).
- o The UTF-8 charset [7].
- o The IETF policy on Character Sets and Languages [3].

10. Open Issues

1. Should there be an explicit request for the list of comparators in the COMPARATOR response code?

2. Should we add a command to fetch the lookup tables associated with a comparator?

3. Need examples for COMPARATOR extension.

Normative References

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

[2] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

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

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

[5] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

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

[7] Yergeau, F., "UTF-8, a transformation format of ISO 10646", draft-yergeau-rfc2279bis-04 (work in progress), February 2003.

[8] Newman, C., "Internet Application Protocol Comparator Registry", draft-newman-i18n-comparator-00 (work in progress), May 2003.

Informative References

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

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

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

[12] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

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

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

[15] Crispin, M. and K. Murchison, "INTERNET MESSAGE ACCESS PROTOCOL - SORT AND THREAD EXTENSION", draft-ietf-imapext-sort-12 (work in progress), March 2003.

Author's Address

Chris Newman
Sun Microsystems
1050 Lakes Drive
West Covina, CA 91790
US

EMail: chris.newman@sun.com

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.