Network Working Group
Request for Comments: 5182
Updates: 3501
Category: Standards Track

A. Melnikov
Isode Ltd.

March 2008

IMAP Extension for Referencing the Last SEARCH Result

Status of This Memo

Abstract

Many IMAP clients use the result of a SEARCH command as the input to perform another operation, for example, fetching the found messages, deleting them, or copying them to another mailbox.

This can be achieved using standard IMAP operations described in RFC 3501; however, this would be suboptimal. The server will send the list of found messages to the client; after that, the client will have to parse the list, reformat it, and send it back to the server. The client can't pipeline the SEARCH command with the subsequent command, and, as a result, the server might not be able to perform some optimizations.

This document proposes an IMAP extension that allows a client to tell a server to use the result of a SEARCH (or Unique Identifier (UID) SEARCH) command as an input to any subsequent command.

1. Introduction

Many IMAP clients use the result of a SEARCH command as the input to perform another operation, for example, fetching the found messages, deleting them, or copying them to another mailbox.

This document proposes an IMAP extension that allows a client to tell a server to use the result of a SEARCH (or UID SEARCH) command as an input to any subsequent command.

The SEARCH result reference extension defines a new SEARCH result option [IMAPABNF] "SAVE" that tells the server to remember the result of the SEARCH or UID SEARCH command (as well as any command based on SEARCH, e.g., SORT and THREAD [SORT]) and store it in an internal

variable that we will reference as the "search result variable". The client can use the "$" marker to reference the content of this internal variable. The "$" marker can be used instead of message sequence or UID sequence in order to indicate that the server should substitute it with the list of messages from the search result variable. Thus, the client can use the result of the latest remembered SEARCH command as a parameter to another command. The search result marker has several advantages:

* it avoids wasted bandwidth and associated delay;

* it allows the client to pipeline a SEARCH [IMAP4] command with a subsequent FETCH/STORE/COPY/SEARCH [IMAP4] or UID EXPUNGE [UIDPLUS] command;

* the client doesn't need to spend time reformatting the result of a SEARCH command into a message set used in the subsequent command;

* it allows the server to perform optimizations. For example, if the server can execute several pipelined commands in parallel (or out of order), presence of the search result marker can allow the server to decide which commands may or may not be executed out of order.

In absence of any other SEARCH result option, the SAVE result option also suppresses any SEARCH response that would have been otherwise returned by the SEARCH command.

1.1. Conventions Used in This Document

In examples, "C:" indicates lines sent by a client that is connected to a server. "S:" indicates lines sent by the server to the client.

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 [KEYWORDS].

Explanatory comments in examples start with // and are not part of the protocol.

2. Overview

2.1. Normative Description of the SEARCHRES Extension

The SEARCH result reference extension described in this document is present in any IMAP4 server implementation that returns "SEARCHRES" as one of the supported capabilities in the CAPABILITY command response. Any such server MUST also implement the [ESEARCH] extension.

Upon successful completion of a SELECT or an EXAMINE command (after the tagged OK response), the current search result variable is reset to the empty sequence.

A successful SEARCH command with the SAVE result option sets the value of the search result variable to the list of messages found in the SEARCH command. For example, if no messages were found, the search result variable will contain the empty list.

Any of the following SEARCH commands MUST NOT change the search result variable:

- a SEARCH command that caused the server to return the BAD tagged response,
- a SEARCH command with no SAVE result option that caused the server to return NO tagged response,
- a successful SEARCH command with no SAVE result option.

A SEARCH command with the SAVE result option that caused the server to return the NO tagged response sets the value of the search result variable to the empty sequence.

When a message listed in the search result variable is EXPUNGEd, it is automatically removed from the list. Implementors are reminded that if the server stores the list as a list of message numbers, it MUST automatically adjust them when notifying the client about expunged messages, as described in Section 7.4.1 of [IMAP4].

If the server decides to send a new UIDVALIDITY value while the mailbox is opened, this causes resetting of the search variable to the empty list.

Note that even if the "$" marker contains the empty list of messages, it must be treated by all commands accepting message sets as parameters as a valid, but non-matching list of messages. For example, the "FETCH $" command would return a tagged OK response and no FETCH responses. See also the Example 5 below.

Note that even if the "$" marker contains the empty list of messages, it must be treated as a valid but non-matching list of messages, by all commands that accept message sets as parameters.

Implementation note: server implementors should note that "$" can reference IMAP message sequences or UID sequences, depending on the context where it is used. For example, the "$" marker can be set as a result of a SEARCH (SAVE) command and used as a parameter to a UID FETCH command (which accepts a UID sequence, not a message sequence), or the "$" marker can be set as a result of a UID SEARCH (SAVE) command and used as a parameter to a FETCH command (which accepts a message sequence, not a UID sequence).

2.2. Examples

1) The following example demonstrates how the client can use the result of a SEARCH command to FETCH headers of interesting messages:

Example 1:

```
C: A282 SEARCH RETURN (SAVE) FLAGGED SINCE 1-Feb-1994
    NOT FROM "Smith"
S: A282 OK SEARCH completed, result saved
C: A283 FETCH $ (UID INTERNALDATE FLAGS RFC822.HEADER)
S: * 2 FETCH (UID 14 ...
S: * 84 FETCH (UID 100 ...
S: * 882 FETCH (UID 1115 ...
S: A283 OK completed
```

The client can also pipeline the two commands:

Example 2:

```
C: A282 SEARCH RETURN (SAVE) FLAGGED SINCE 1-Feb-1994
    NOT FROM "Smith"
C: A283 FETCH $ (UID INTERNALDATE FLAGS RFC822.HEADER)
S: A282 OK SEARCH completed
S: * 2 FETCH (UID 14 ...
S: * 84 FETCH (UID 100 ...
S: * 882 FETCH (UID 1115 ...
S: A283 OK completed
```

2) The following example demonstrates that the result of one SEARCH command can be used as input to another SEARCH command:

Example 3:

```
C: A300 SEARCH RETURN (SAVE) SINCE 1-Jan-2004
    NOT FROM "Smith"
S: A300 OK SEARCH completed
C: A301 UID SEARCH UID $ SMALLER 4096
S: * SEARCH 17 900 901
S: A301 OK completed
```

Note that the second command in Example 3 can be replaced with:

```
C: A301 UID SEARCH $ SMALLER 4096
```

and the result of the command would be the same.

3) The following example shows that the "$" marker can be combined with other message numbers using the OR SEARCH criterion.

Example 4:

```
C: P282 SEARCH RETURN (SAVE) SINCE 1-Feb-1994
    NOT FROM "Smith"
S: P282 OK SEARCH completed
C: P283 SEARCH CHARSET UTF-8 (OR $ 1,3000:3021) TEXT {8}
C: YYYYYYYY
S: * SEARCH 882 1102 3003 3005 3006
S: P283 OK completed
```

Note: Since this document format is restricted to 7-bit ASCII text, it is not possible to show actual UTF-8 data. The "YYYYYYYY" is a placeholder for what would be 8 octets of 8-bit data in an actual transaction.

4) The following example demonstrates that a failed SEARCH sets the search result variable to the empty list.

Example 5:

```
C: B282 SEARCH RETURN (SAVE) SINCE 1-Feb-1994
    NOT FROM "Smith"
S: B282 OK SEARCH completed
C: B283 SEARCH CHARSET KOI8-R (OR $ 1,3000:3021) TEXT {4}
C: XXXX
S: B283 NO [BADCHARSET UTF-8] KOI8-R is not supported
//After this command the saved result variable contains
//no messages.  A client that wants to reissue the B283
//SEARCH command with another CHARSET would have to reissue
//the B282 command as well.  One possible workaround for
//this is to include the desired CHARSET parameter
//in the earliest SEARCH RETURN (SAVE) command in a
//sequence of related SEARCH commands.
//A better approach might be to always use CHARSET UTF-8
//instead.
```

Note: Since this document format is restricted to 7-bit ASCII text, it is not possible to show actual KOI8-R data. The "XXXX" is a placeholder for what would be 4 octets of 8-bit data in an actual transaction.

5) The following example demonstrates that it is not an error to use the "$" marker when it contains no messages.

Example 6:

```
C: E282 SEARCH RETURN (SAVE) SINCE 28-Oct-2006
    NOT FROM "Eric"
C: E283 COPY $ "Other Messages"
//The "$" contains no messages
S: E282 OK SEARCH completed
S: E283 OK COPY completed, nothing copied
```

2.3. Multiple Commands in Progress

Use of a SEARCH RETURN (SAVE) command followed by a command using the "$" marker creates direct dependency between the two commands. As directed by Section 5.5 of [IMAP4], a server MUST execute the two commands in the order they were received. (A server capable of out-of-order execution can in some cases execute the two commands in parallel, for example, if a SEARCH RETURN (SAVE) is followed by "SEARCH $", the search criteria from the first command can be directly substituted into the second command.)

A client supporting this extension MAY pipeline a SEARCH RETURN (SAVE) command with one or more command using the "$" marker, as long as this doesn't create an ambiguity, as described in Section 5.5 of [IMAP4].

Example 7:

```
C: F282 SEARCH RETURN (SAVE) KEYWORD $Junk
C: F283 COPY $ "Junk"
C: F284 STORE $ +FLAGS.Silent (\Deleted)
S: F282 OK SEARCH completed
S: F283 OK COPY completed
S: F284 OK STORE completed
```

Example 8:

```
C: G282 SEARCH RETURN (SAVE) KEYWORD $Junk
C: G283 SEARCH RETURN (ALL) SINCE 28-Oct-2006
    FROM "Eric"
//The server can execute the two SEARCH commands
//in any order, as they don't have any dependency.
//Note that the second command is making use of
//the [ESEARCH] extension.
S: * ESEARCH (TAG "G283") ALL 3:15,27,29:103
S: G283 OK SEARCH completed
S: G282 OK SEARCH completed
```

The following example demonstrates that the result of the second SEARCH always overrides the result of the first.

Example 9:

```
C: H282 SEARCH RETURN (SAVE) KEYWORD $Junk
C: H283 SEARCH RETURN (SAVE) SINCE 28-Oct-2006
    FROM "Eric"
S: H282 OK SEARCH completed
S: H283 OK SEARCH completed
```

2.4. Interaction with ESEARCH Extension

Servers that implement the extension defined in this document MUST implement [ESEARCH] and conform to additional requirements listed in this section.

The SAVE result option doesn't change whether the server would return items corresponding to MIN, MAX, ALL, or COUNT [ESEARCH] result options.

When the SAVE result option is combined with the MIN or MAX [ESEARCH] result option, and none of the other ESEARCH result options are present, the corresponding MIN/MAX is returned (if the search result is not empty), but the "$" marker would contain a single message as returned in the MIN/MAX return item.

If the SAVE result option is combined with both MIN and MAX result options, and none of the other ESEARCH result options are present, the "$" marker would contain one or two messages as returned in the MIN/MAX return items.

If the SAVE result option is combined with the ALL and/or COUNT result option(s), the "$" marker would always contain all messages found by the SEARCH or UID SEARCH command. (Note that the last rule might affect ESEARCH implementations that optimize how the COUNT result is constructed.)

The following table summarizes the additional requirement on ESEARCH server implementations described in this section.

Combination of Result option	"$" marker value
SAVE MIN	MIN
SAVE MAX	MAX
SAVE MIN MAX	MIN & MAX
SAVE * [m]	all found messages

where '*' means "ALL" and/or "COUNT"
'[m]' means optional "MIN" and/or "MAX"

The following example demonstrates behavioral difference for different combinations of ESEARCH result options. Explanatory comments start with // and are not part of the protocol:

Example 10:

```
          C: C282 SEARCH RETURN (ALL) SINCE 12-Feb-2006
              NOT FROM "Smith"
          S: * ESEARCH (TAG "C283") ALL 2,10:15,21
        //$ value hasn't changed
          S: C282 OK SEARCH completed

          C: C283 SEARCH RETURN (ALL SAVE) SINCE 12-Feb-2006
              NOT FROM "Smith"
          S: * ESEARCH (TAG "C283") ALL 2,10:15,21
        //$ value is 2,10:15,21
          S: C283 OK SEARCH completed

          C: C284 SEARCH RETURN (SAVE MIN) SINCE 12-Feb-2006
              NOT FROM "Smith"
          S: * ESEARCH (TAG "C284") MIN 2
        //$ value is 2
          S: C284 OK SEARCH completed

          C: C285 SEARCH RETURN (MAX SAVE MIN) SINCE
              12-Feb-2006 NOT FROM "Smith"
          S: * ESEARCH (TAG "C285") MIN 2 MAX 21
        //$ value is 2,21
          S: C285 OK SEARCH completed

          C: C286 SEARCH RETURN (MAX SAVE MIN COUNT)
              SINCE 12-Feb-2006 NOT FROM "Smith"
          S: * ESEARCH (TAG "C286") MIN 2 MAX 21 COUNT 8
        //$ value is 2,10:15,21
          S: C286 OK SEARCH completed

          C: C286 SEARCH RETURN (ALL SAVE MIN) SINCE
              12-Feb-2006 NOT FROM "Smith"
          S: * ESEARCH (TAG "C286") MIN 2 ALL 2,10:15,21
        //$ value is 2,10:15,21
          S: C286 OK SEARCH completed
```

2.5. Refusing to Save Search Results

In some cases, the server MAY refuse to save a SEARCH (SAVE) result, for example, if an internal limit on the number of saved results is reached.

In this case, the server MUST return a tagged NO response containing the NOTSAVED response code and set the search result variable to the empty sequence, as described in Section 2.1.

3. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [ABNF]. Non-terminals referenced but not defined below are as defined in [IMAP4] or [IMAPABNF].

Except as noted otherwise, all alphabetic characters are case-insensitive. The use of upper- or lower-case characters to define token strings is for editorial clarity only. Implementations MUST accept these strings in a case-insensitive fashion.

```
capability          =/ "SEARCHRES"
                       ;; capability is defined in [IMAP4]

sequence-set        =/ seq-last-command
                       ;; extends sequence-set to allow for
                       ;; "result of the last command" indicator.

seq-last-command    = "$"

search-return-opt   = "SAVE"
                       ;; conforms to generic search-return-opt
                       ;; syntax defined in [IMAPABNF]

resp-text-code      =/ "NOTSAVED"
                       ;; <resp-text-code> from [IMAP4]
```

4. Security Considerations

This extension requires the server to keep additional state, that may be used to simplify Denial of Service attacks. In order to minimize damage from such attacks, server implementations MAY limit the number of saved searches they allow across all connections at any given time and return the tagged NO response containing the NOTSAVED response code (see Section 2.5) to a SEARCH RETURN (SAVE) command when this limit is exceeded.

Apart from that, it is believed that this extension doesn't raise any additional security concerns not already discussed in [IMAP4].

5. IANA Considerations

This document defines the "SEARCHRES" IMAP capability. IANA has added it to the IMAP4 Capabilities Registry, which is currently located at:

http://www.iana.org/assignments/imap4-capabilities

6. Acknowledgments

The author would like to thank Mark Crispin, Cyrus Daboo, and Curtis King for remembering that this document had to be written, as well as for comments and corrections received.

The author would also like to thank Dave Cridland, Mark Crispin, Chris Newman, Dan Karp, and Spencer Dawkins for comments and corrections received.

Valuable comments, both in agreement and in dissent, were received from Arnt Gulbrandsen.

7. References

7.1. Normative References

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

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

[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.

[ESEARCH] Melnikov, A. and D. Cridland, "IMAP4 Extension to SEARCH Command for Controlling What Kind of Information Is Returned", RFC 4731, November 2006.

7.2. Informative References

[UIDPLUS] Crispin, M., "Internet Message Access Protocol (IMAP) - UIDPLUS extension", RFC 4315, December 2005.

[SORT] Crispin, M. and K. Murchison, "INTERNET MESSAGE ACCESS PROTOCOL - SORT AND THREAD EXTENSIONS", Work in Progress, Septemeber 2007.

Author's Address

Alexey Melnikov
Isode Ltd.
5 Castle Business Village,
36 Station Road,
Hampton, Middlesex,
TW12 2BX, United Kingdom

EMail: Alexey.Melnikov@isode.com

Full Copyright Statement

Intellectual Property