

Workgroup: EXTRA  
Internet-Draft:  
draft-ietf-extra-imap-inprogress  
Published: 30 January 2024  
Intended Status: Standards Track  
Expires: 2 August 2024  
Authors: M. Bettini  
Open-Xchange Oy  
**IMAP4 Response Code for Command Progress Notifications.**

## Abstract

This document defines a new IMAP untagged response code, "INPROGRESS", that provides structured numeric progress status indication for long-running commands.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 August 2024.

## Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

- [1. Introduction](#)
  - [2. Conventions Used in This Document](#)
  - [3. CAPABILITY Identification](#)
  - [4. The "INPROGRESS" Response Code](#)
  - [5. Formal Syntax](#)
  - [6. Security Considerations](#)
  - [7. IANA Considerations](#)
  - [8. Normative References](#)
- [Author's Address](#)

### 1. Introduction

Internet Message Access Protocol (IMAP) [[RFC9051](#)] commands can require a considerable amount of time to be completed by the server. In these cases, the client has no information about the progress of the commands. It is already possible to expose updates with a generic untagged response, like "\* OK Still on it, 57% done"; however, this does not provide a standard way to communicate with the client and allow it to inform the user of the progress of the long-running actions.

This document extends the Internet Message Access Protocol (IMAP) [[RFC9051](#)] with:

- \*a new "INPROGRESS" response code [[RFC5530](#)]. The new response code provides consistent means for a client to receive progress update notifications on command completion status.

- \*a new "INPROGRESS" capability [[RFC9051](#)]. The new capability informs the client that the server emits progress update notifications, via the "INPROGRESS" response code

### 2. Conventions Used in This Document

"Conventions" are basic principles or procedures. Document conventions are noted in this section.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

The word "can" (not "may") is used to refer to a possible circumstance or situation, as opposed to an optional facility of the protocol.

Conventions for notations are as in [[RFC9051](#)] and [[RFC5530](#)].

In examples, "C:" and "S:" indicate lines sent by the client and server, respectively. Note that each line includes the terminating CRLF.

### 3. CAPABILITY Identification

IMAP servers that support this extension **MUST** include "INPROGRESS" in the response list to the CAPABILITY command.

### 4. The "INPROGRESS" Response Code

The server **MAY** send the "INPROGRESS" Response Code to notify the client about the progress of the commands in execution, or simply to prevent the client from timing out and terminating the connection. The notifications **MAY** be sent for any IMAP command. If the server elects to send notifications, it is **RECOMMENDED** that these are sent every 10..15 seconds.

The response code is meant to appear embedded inside an untagged OK reply. The response code **MUST NOT** appear in a tagged response (the command has completed and further progress notifications make no sense).

The response code **MAY** embed a list of details, composed in order of:

1. CMD-TAG: the cmd-tag [[RFC9051](#)] that originated the long-running command. If the tag is not available, or if it contains the "]" character, it **MUST** be set to NIL. This still produces a usable notification, unless multiple commands are in flight simultaneously. A client can ensure reception of notifications with cmd-tag(s) by simply refraining from the use of character "]" in the originating command tags.
2. PROGRESS: a number indicating the number of items processed so far. The number **MUST** be non-negative and **SHOULD** be monotonically increasing. If the PROGRESS is not available, both PROGRESS and GOAL **MUST** be set to NIL.
3. GOAL: a number indicating the total number of items to be processed. The number **MUST** be positive and it **SHOULD NOT** change between successive notifications for the same command (i.e. for the same cmd-tag). This is the number that PROGRESS is expected to reach at the completion of the command and therefore it **MUST** be strictly greater than PROGRESS. If the GOAL is not known, it **MUST** be set to NIL.

If the response code does not embed a list of details, all details are to be interpreted as NIL.

The server can provide the progress notifications details with different degrees of completeness:

- bare keepalive
  - \* OK [INPROGRESS] Hang in there..
- keepalive with indication of the command tag
  - \* OK [INPROGRESS ("tag" NIL NIL)] Hang in there..
- progress indication with unknown GOAL
  - \* OK [INPROGRESS ("tag" 175 NIL)] Processed 175 items so far
- progress indication with the indication of the GOAL
  - \* OK [INPROGRESS ("tag" 175 1000)] Processed 17% of the items

Examples:

```
C: A001 search text "this will be slow"
  [13 seconds later]
S: * OK [INPROGRESS ("001" 454 1000)] Processed 45% of the items
  [14 seconds later]
S: * OK [INPROGRESS ("001" 999 1000)] Processed 99% of the items
  [5 second later]
S: * SEARCH 447 735
S: A001 OK Search completed (23.387 + 0.004 + 0.017 secs).
```

```
C: A003 COPY 2000:4000 Meeting-Minutes
  [12 seconds later]
S: * OK [INPROGRESS ("A003" 175 2001)] Still working on this...
  [14 seconds later]
S: * OK [INPROGRESS ("A003" 440 2001)] Still working on this...
  [13 seconds later]
S: * OK [INPROGRESS ("A003" 987 2001)] Still working on this...
  [14 seconds later]
S: * OK [INPROGRESS ("A003" 1388 2001)] Still working on this...
  [14 seconds later]
S: * OK [INPROGRESS ("A003" 1876 2001)] Still working on this...
  [9 seconds later]
S: A003 OK Copy completed
```

PROGRESS and GOAL **SHOULD** be counts of the kind of item being processed - in most cases, messages counts. If that is not possible, the counts **SHOULD** be percentages, with progress varying between 0 and 99 and goal fixed at 100.

The command is completed when PROGRESS equals GOAL. In this case, the server **SHOULD NOT** send a further progress notification before the proper command completion.

If the command completes before the first server notification deadline, there will be no notifications at all. The client **MUST**

assume PROGRESS to be 0 and GOAL to be unknown until the server issues a notification for the command.

While the server **SHOULD** keep GOAL constant and PROGRESS monotonically increasing, there are circumstances where this might not be possible. The client **MUST** be prepared to handle cases where the server cannot keep GOAL constant and/or PROGRESS monotonically increasing. When the GOAL changes or the PROGRESS goes backward, the **RECOMMENDED** interpretation is that the previous GOAL has been reached, but the server discovered that further (long-running) work is required (either with known or unknown new GOAL),

The client **MAY** disregard progress notifications entirely or process them only in relation with specific commands. If a User Interface is involved, it is the client's duty to decide which of these commands are blocking on the user experience, since this may differ based on implementation details.

Also, the client **MUST NOT** consider the values to be authoritative for any other use than evaluating the progress of the commands. E.g.: the client must not use the GOAL field in place of the proper output of a SEARCH command to know the number of messages in a folder.

## 5. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) [[RFC5234](#)] notation. Elements not defined here can be found in the formal syntax of the ABNF [[RFC5234](#)] and IMAP [[RFC9051](#)] specifications.

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

```
inprogress-tag           = quoted / nil
inprogress-state-unknown = nil      SP nil
inprogress-state-counting = number SP nil
inprogress-state-known-goal = number SP nz-number

inprogress-state = inprogress-state-unknown
                  / inprogress-state-counting
                  / inprogress-state-known-goal

resp-text-code =/ "INPROGRESS" [ SP "(" inprogress-tag SP
                               inprogress-state ")" ]
```

## 6. Security Considerations

The details of the response code are not expected to disclose any information that isn't currently available from commands output. The progress details could be obtained anyway by a series of sending commands with different workloads - either by constructing data sets or searching in the appropriate way into them.

The client must protect itself against data sent by a malicious server. Specifically, the client should guard against values that can cause arithmetic exceptions, like  $GOAL = 0$ ,  $GOAL/VALUE < 0$ ,  $GOAL/VALUE \geq 2^{32}$ . (these are not possible within a correct implementation of the ABNF syntax above), and  $VALUE > GOAL$ .

## 7. IANA Considerations

IANA is requested to add "INPROGRESS" to the "IMAP Response Codes" registry located at <https://www.iana.org/assignments/imap-response-codes>, with a reference to this document.

IANA is requested to add "INPROGRESS" to the "IMAP Capabilities" registry located at <https://www.iana.org/assignments/imap-capabilities>, with a reference to this document.

## 8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.
- [RFC5530] Gulbrandsen, A., "IMAP Response Codes", RFC 5530, DOI 10.17487/RFC5530, May 2009, <https://www.rfc-editor.org/info/rfc5530>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.
- [RFC9051] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/info/rfc9051>.

**Author's Address**

Marco Bettini  
Open-Xchange Oy  
Lars Sonckin kaari 10  
FI-02600 Espoo  
Finland

Email: [marco.bettini@open-xchange.com](mailto:marco.bettini@open-xchange.com)