RFC 9051 : Internet Message Access Protocol (IMAP) - Version 4rev2

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 3 septembre 2021

Date de publication du RFC : Août 2021

https://www.bortzmeyer.org/9051.html

IMAP, protocole d'accès distant à des boites aux lettres n'est pas juste un protocole, c'est tout un écosystème, décrit dans de nombreux RFC. Celui-ci est le socle de la version actuelle d'IMAP, la « 4rev2 ». Elle remplace la précédente version, « 4rev1 » (normalisée dans le RFC 3501 [1]). Le principal changement est sans doute une internationalisation plus poussée pour les noms de boites aux lettres et les en-têtes des messages.

Les protocoles traditionnels du courrier électronique, notamment SMTP ne permettaient que d'acheminer le courrier jusqu'à destination. Pour le lire, la solution la plus courante aujourd'hui est en mode client/serveur, avec IMAP, qui fait l'objet de notre RFC. IMAP ne permet pas d'envoyer du courrier, pour cela, il faut utiliser le RFC 6409. IMAP fonctionne sur un modèle où les messages ont plutôt vocation à rester sur le serveur, et où les clients peuvent, non seulement les récupérer mais aussi les classer, les détruire, les marquer comme lus, etc. IMAP permet de tout faire à distance.

Pour apprendre IMAP, il vaut mieux ne pas commencer par le RFC, qui est très riche. (La lecture du RFC 2683 est recommandée aux implémenteurs.) Pourtant, le principe de base est simple. IMAP est client/serveur. Le client se connecte en TCP au serveur, sur le port 143, et envoie des commandes sous forme de texte, comme avec SMTP. Il s'authentifie avec la commande `LOGIN`, choisit une boite aux lettres avec la commande `SELECT`, récupère un message avec la commande `FETCH`. IMAP peut aussi utiliser du TLS implicite, sur le port 993. Voici un exemple de session. Notons tout de suite, c'est un point très important du protocole IMAP, que chaque commande est précédée d'une **étiquette** ("*tag*", voir les sections 2.2.1 et 5.5 du RFC) arbitraire qui change à chaque commande et qui permet d'associer une réponse à une commande, IMAP permettant d'envoyer plusieurs commandes sans attendre de réponse. Les réponses portent l'étiquette correspondante, ou bien un astérisque s'il s'agit d'un message du serveur non sollicité. Ici, le serveur utilise le logiciel Archiveopteryx <https://archiveopteryx.org/> :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3501.txt

```
% telnet imap.example.org imap
Trying 192.0.2.23...
Connected to imap.example.org.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 AUTH=CRAM-MD5 AUTH=DIGEST-MD5 AUTH=PLAIN COMPRESS=DEFLATE ID LITERAL+ STARTTLS] i
com1 LOGIN stephane vraimentsecret
com1 OK [CAPABILITY IMAP4rev1 ACL ANNOTATE BINARY CATENATE CHILDREN COMPRESS=DEFLATE CONDSTORE ESEARCH ID ID
com2 SELECT INBOX
* 189 EXISTS
* 12 RECENT
* OK [UIDNEXT 1594] next uid
* OK [UIDVALIDITY 1] uid validity
* FLAGS (\Deleted \Answered \Flagged \Draft \Seen)
* OK [PERMANENTFLAGS (\Deleted \Answered \Flagged \Draft \Seen \*)] permanent flags
com2 OK [READ-WRITE] done
```

Le serveur préfixe ses réponses par OK si tout va bien, NO s'il ne pas réussi à faire ce qu'on lui demande et BAD si la demande était erronnée (commande inconnue, par exemple). Ici, avec l'étiquette A2 et un Dovecot :

```
A2 CAVAPAS
A2 BAD Error in IMAP command CAVAPAS: Unknown command (0.001 + 0.000 secs).
```

Si le serveur n'est accessible qu'en TLS implicite (TLS démarrant automatiquement au début, sans `STARTTLS`), on peut utiliser un client TLS en ligne de commande, comme celui de GnuTLS :

```
% gnutls-cli -p 993 imap.maison.example
- Certificate[0] info:
- subject 'CN=imap.maison.example', issuer 'CN=CAcert Class 3 Root,OU=http://www.CAcert.org,O=CAcert Inc.',
 ...
- Status: The certificate is trusted.
- Description: (TLS1.3)-(ECDHE-SECP256R1)-(RSA-PSS-RSAE-SHA256)-(AES-256-GCM)

[À partir d'ici, on fait de l'IMAP]

* OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE LITERAL+ AUTH=PLAIN] Dovecot (Debian) reac
A1 LOGIN stephane vraimenttropsecret
A1 OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE SORT SORT=DISPLAY THREAD=REFERENCES THREA
A2 SELECT INBOX
...
* 0 EXISTS
* 0 RECENT
A2 OK [READ-WRITE] Select completed (0.002 + 0.000 + 0.001 secs).
...
A14 FETCH 1 (BODY.PEEK[HEADER.FIELDS (SUBJECT)])
* 1 FETCH (BODY[HEADER.FIELDS (SUBJECT)] {17}
Subject: Test

)
A14 OK Fetch completed (0.007 + 0.000 + 0.006 secs).
```

Les commandes IMAP, comme `FETCH` dans l'exemple ci-dessus, sont décrites dans la section 6 du RFC. Pour tester son serveur IMAP, on peut aussi utiliser fetchmail et lui demander d'afficher toute la session avec `-v -v` :

```
 % fetchmail -v -v -u MYLOGIN imap.1and1.fr
Enter password for MYLOGIN@imap.1and1.fr:
fetchmail: 6.3.6 querying imap.1and1.fr (protocol auto) at Tue Apr 15 12:00:27 2008: poll started
fetchmail: 6.3.6 querying imap.1and1.fr (protocol IMAP) at Tue Apr 15 12:00:27 2008: poll started
Trying to connect to 212.227.15.141/143...connected.
fetchmail: IMAP< * OK IMAP server ready H mimap3 65564
fetchmail: IMAP> A0001 CAPABILITY
fetchmail: IMAP< * CAPABILITY IMAP4rev1 LITERAL+ ID STARTTLS CHILDREN QUOTA IDLE NAMESPACE UIDPLUS UNSELECT SORT
fetchmail: IMAP< A0001 OK CAPABILITY finished.
fetchmail: Protocol identified as IMAP4 rev 1
fetchmail: IMAP> A0002 STARTTLS
fetchmail: IMAP< A0002 OK Begin TLS negotiation.
fetchmail: Issuer Organization: Thawte Consulting cc
fetchmail: Issuer CommonName: Thawte Premium Server CA
fetchmail: Server CommonName: imap.1and1.fr
fetchmail: imap.1and1.fr key fingerprint: 93:13:99:6A:3F:23:73:C3:00:37:4A:39:EE:22:93:AB
fetchmail: IMAP> A0003 CAPABILITY
fetchmail: IMAP< * CAPABILITY IMAP4rev1 LITERAL+ ID CHILDREN QUOTA IDLE NAMESPACE UIDPLUS UNSELECT SORT AUTH=LOG
fetchmail: IMAP< A0003 OK CAPABILITY finished.
fetchmail: Protocol identified as IMAP4 rev 1
fetchmail: imap.1and1.fr: upgrade to TLS succeeded.
fetchmail: IMAP> A0004 LOGIN "m39041005-1" *
fetchmail: IMAP< A0004 OK LOGIN finished.
fetchmail: selecting or re-polling default folder
fetchmail: IMAP> A0005 SELECT "INBOX"
```

En IMAP, un message peut être identifié par un UID (”*Unique Identifier*”) ou par un numéro. Le numéro n'est pas global, il n'a de sens que pour une boite donnée, et il peut changer, par exemple si on détruit des messages. L'UID, lui, est stable, au moins pour une session (et de préférence pour toutes). Les messages ont également des attributs (”*flags*”). Ceux qui commencent par une barre oblique inverse comme `\Seen` (indique que le message a été lu), `\Answered` (on y a répondu), etc, sont obligatoires, et il existe également des attributs optionnels commençant par un dollar comme `$Junk` (spam). Les fonctions de recherche d'IMAP pourront utiliser ces attributs comme critères de recherche. Les attributs optionnels (”*keywords*”) sont listés dans un registre IANA `<https://www.iana.org/assignments/imap-jmap-keywords/imap-jmap-keywords.xml#imap-keywords-1>`, et on peut en ajouter (le RFC 5788 explique comment).

IMAP a des fonctions plus riches, notamment la possibilité de chercher dans les messages (section 6.4.4 du RFC), ici, on extrait les messages de mai 2007, puis on récupère le sujet du premier message, par son numéro :

```
com10 SEARCH since 1-May-2007 before 31-May-2007
* SEARCH 12
com10 OK done
com11 FETCH 1 (BODY.PEEK[HEADER.FIELDS (SUBJECT)])
* 1 FETCH (BODY[HEADER.FIELDS (Subject)] {17}
Subject: Rapport sur les fonctions de vue dans Archiveopteryx

)
com11 OK done
```

IMAP 4rev2 est complètement internationalisé et, par exemple, peut gérer des en-têtes en UTF-8, comme décrits dans le RFC 6532, ce qui est une des améliorations par rapport à 4rev1.

Vous noterez dans les réponses `CAPABILITY` montrées plus haut, que le serveur indique la ou les versions d'IMAP qu'il sait gérer. Il peut donc annoncer 4rev1 et 4rev2, s'il est prêt à gérer les deux. Le

client devra utiliser la commande ENABLE pour choisir. S'il choisit 4rev2, le serveur pourra utiliser les nouveautés de 4rev2, notamment dans le domaine de l'internationalisation (noms de boites en UTF-8, alors que 4rev1 savait tout juste utiliser un bricolage basé sur UTF-7).

IMAP 4rev2 est compatible avec 4rev1 (un logiciel de la précédente norme peut interagir avec un logiciel de la nouvelle). Il peut même interagir avec l'IMAP 2 du RFC 1776 (IMAP 3 n'a jamais été publié <https://datatracker.ietf.org/doc/draft-ietf-imap-imap2bis/>), cf. RFC 2061.

L'annexe E résume les principaux changements depuis le RFC 3501, qui normalisait IMAP 4rev1 :

— Messages plus grands (taille stockée sur 63 bits et plus seulement 32). Un serveur qui accepte les messages de plus de quatre gigoctets devra donc les masquer aux clients 4rev1 (ou bien trouver une autre stratégie).
— UTF-8 pour les noms de boites et les sujets des messages.
— Incorporation de nombreuses extensions précédemment séparées, trop nombreuses pour que je les cite toutes. Il y a par exemple le BINARY du RFC 3516 ou le LIST-EXTENDED du RFC 5258.
— Beaucoup de clarifications du texte.
— Abandon de certaines choses, aussi, comme le code de réponse UNSEEN de la commande SELECT.
— Application des progrès de la cryptanalyse, donc remplacement de MD5 par SHA-256.
— Suppression de la convention du préfixe « X- », en application du RFC 6648.

La réalisation d'un client ou d'un serveur IMAP soulève plein de problèmes pratiques, que la section 5 de notre RFC traite. Par exemple, les noms des boites aux lettres peuvent être en Unicode, plus précisément le sous-ensemble d'Unicode du RFC 5198 (cela n'était pas possible en 4rev1). Un logiciel doit donc s'attendre à rencontrer de tels noms.

IMAP est mis en œuvre dans de nombreux serveurs comme Dovecot, Courier <http://www.courier-mta.org/>, ou Archiveopteryx <https://archiveopteryx.org/>, déjà cité (mais qui semble abandonné). Mais, comme vous l'avez vu dans les exemples de session IMAP cités plus haut, la version de notre RFC, « 4rev2 » n'a pas encore forcément atteint tous les logiciels.

Côté client, on trouve du IMAP dans beaucoup de logiciels, des "*webmails*", des MUA classiques comme mutt, des MUA en ligne de commande comme fetchmail, très pratique pour récupérer son courrier. (Si vous écrivez un logiciel IMAP à partir de zéro, ce RFC recommande la lecture préalable du RFC 2683.)

Il existe également des bibliothèques toutes faites pour programmer son client IMAP à son goût comme imaplib <https://docs.python.org/3/library/imaplib.html> pour Python. Voici un exemple d'un court programme Python qui se connecte à un serveur IMAP, sélectionne tous les messages et les récupère. On note que la bibliothèque a choisi de rester très proche du vocabulaire du RFC :

```
import imaplib

# Unlike what the documentation says, "host" has no proper default.
connection = imaplib.IMAP4_SSL(host='localhost')
connection.login("stephane", "thisissecret")
connection.select() # Select the default mailbox
typee, data = connection.search(None, "ALL")
for num in data[0].split():
    # Fetches the whole message
    # "RFC822" est le terme traditionnel mais le format des messages
    # est désormais dans le RFC 5322.
    typ, data = connection.fetch(num, '(RFC822)')
    print('Message %s\n%s\n' % (num, data[0][1]))
connection.close()
connection.logout()
```