RFC 3501 : INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 31 mai 2007

Date de publication du RFC : Mars 2003

http://www.bortzmeyer.org/3501.html

IMAP, protocole d'accès distant à des boîtes 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.

Les protocoles traditionnels du courrier électronique, notamment SMTP ne permettaient que d'acheminer le courrier jusqu'à destination. Pour le lire, pendant longtemps, la seule solution était d'accéder directement aux fichiers où étaient stockés les messages. Cette possibilité existe toujours mais le monde du courrier s'est enrichi de nouvelles fonctions, comme le stockage du courrier dans une base de données relationnelle ou comme la récupération du courrier à distance, en mode client/serveur. Pour cette récupération, le premier protocole a été POP, dans le RFC 918 [1] (aujourd'hui dans le RFC 1939). Et le plus courant est aujourd'hui IMAP, qui fait l'objet de notre RFC, qui remplace le RFC 2060.

POP est simple, très simple. Il est parfait lorsqu'on veut simplement télécharger les messages sur son poste. IMAP fonctionne sur un modèle différent, 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. Certaines de ces possibilités existaient déjà avec POP mais étaient peu utilisées. IMAP permet au contraire, via ses nombreux RFC, de tout faire à distance.

IMAP est donc un protocole complexe et notre RFC annonce la couleur en disant qu'il est une norme, pas un tutoriel. Pour apprendre IMAP, il vaut mieux ne pas commencer par le RFC. (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, et envoie des commandes sous forme de texte, comme avec SMTP. Il s'authentifie avec la commande `LOGIN`, choisit une boîte aux lettres avec la commande `SELECT`, récupère un message avec la commande `FETCH`. Voici un exemple de session. Notons tout de suite que chaque commande est précédée d'une **étiquette** ("*tag*", voir la section 2.2.1 du RFC) 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). Ici, le serveur utilise le logiciel Archiveopteryx <http://www.archiveopteryx.org/> :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc918.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
```

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
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
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"
```

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 :

```
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 est mis en œuvre dans de nombreux serveurs comme Dovecot, Courier <http://www.courier-mta.org/>, ou Archiveopteryx, déja cité.

Côté client, on trouve du IMAP dans beaucoup de logiciels, des "*webmails*" comme SquirrelMail, des MUA classiques comme mutt, des MUA en ligne de commande comme fetchmail, très pratique pour récupérer son courrier.

Il existe également des bibliothèques toutes faites pour programmer son client IMAP à son goût comme imaplib <http://docs.python.org/lib/module-imaplib.html> pour Python. Voici un exemple d'un court programme Python qui se connecte à un serveur IMAP, sélectionne les messages d'avril 2007 et les récupère. On note que la bibliothèque a choisi de rester très proche du vocabulaire du RFC :

```
import imaplib

connection = imaplib.IMAP4()
connection.login("stephane", "thisissecret")
connection.select() # Select the default mailbox
typee, data = connection.search(None,
                                '(since "1-Apr-2007")',
                                '(before "1-May-2007")')
for num in data[0].split():
    # Fetches the whole message
    typ, data = connection.fetch(num, '(RFC822)')
    print 'Message %s\n%s\n' % (num, data[0][1])
connection.close()
connection.logout()
```