

# RFC 5023 : The Atom Publishing Protocol

Stéphane Bortzmeyer

<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 9 Octobre 2007

Date de publication du RFC : Octobre 2007

<http://www.bortzmeyer.org/5023.html>

---

Le nouveau protocole APP permet la publication et la mise à jour de documents Atom. Il vise à faciliter la publication sur le Web, notamment pour les blogs.

APP est clairement issu du monde des blogs. Si on veut mettre à jour un blog automatiquement, il existe plusieurs protocoles avec XML-RPC <<http://www.bortzmeyer.org/xml-rpc-for-blogs.html>> mais rien de normalisé. Ce rôle est celui principalement visé par APP. Mais APP permettra aussi beaucoup d'autres choses sur le Web, concurrençant aussi des protocoles comme WebDAV (RFC 4918<sup>1</sup>).

APP est un protocole de la famille REST, bâti au dessus de HTTP (RFC 2616).

APP repose sur les concepts de **ressources** et de **collections** (section 3). Une ressource est une information accessible par le Web. Dans APP, elle est fréquemment au format Atom (RFC 4287) et est alors nommée une "*entry resource*". Les ressources non-Atom sont nommées des "*media resource*". Elles sont en général « pointées » par une "*entry resource*" qui contient les méta-données les concernant et un lien vers la "*media resource*".

Les sections 7 et 8 du RFC spécifient également deux nouveaux formats XML, pour des documents décrivant les catégories (un mécanisme de classification des ressources) et les collections.

Ces ressources et ces collections sont identifiées par des URI (plus exactement par des IRI, RFC 3987). APP agit sur les collections et les ressources par l'intermédiaires de méthodes HTTP qui sont classiques pour les programmeurs REST (section 4.3 du RFC et des détails de protocole en section 5) :

- GET récupère une ressource ou une collection,

---

1. Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc4918.txt>

- PUT met à jour une ressource,
- POST modifie une collection, par exemple en créant une ressource,
- et DELETE, évidemment, détruit.

Le RFC précise bien que les serveurs APP jouissent d'une grande liberté d'action et que le client doit être prêt à tout, notamment si on utilise des méthodes HTTP dont l'action n'est pas définie par le RFC. Par exemple, DELETE sur une collection n'est pas spécifié mais peut être accepté par certains serveurs. Même chose pour le PUT sur une ressource pas encore existante, mentionné plus loin.

Voyons maintenant un exemple, tiré de la section 9 du RFC, montrant la création d'une ressource, obtenue par un POST sur l'URL d'une collection (ici, /edit/) :

```
[Envoyé par le client]
POST /edit/ HTTP/1.1
Host: example.org
User-Agent: Thingio/1.0
Authorization: Basic ZGFmZnk6c2VjZjZlJldA==
Content-Type: application/atom+xml;type=entry
Content-Length: 343
Slug: First Post

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <title>Atom-Powered Robots Run Amok</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <author><name>John Doe</name></author>
  <content>Some text.</content>
</entry>

[Répondu par le serveur]
HTTP/1.1 201 Created
Date: Fri, 7 Oct 2005 17:17:11 GMT
Content-Length: 430
Content-Type: application/atom+xml;type=entry;charset="utf-8"
Location: http://example.org/edit/first-post.atom
ETag: "c180de84f991g8"

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <title>Atom-Powered Robots Run Amok</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
  <updated>2007-08-01T16:46:18Z</updated>
  <author><name>John Doe</name></author>
  <content>Some text.</content>
  <link rel="edit"
    href="http://example.org/edit/first-post.atom"/>
</entry>
```

Le serveur renvoie le document Atom car il est libre de modifier ce qu'a envoyé le client (ici, la date a changé dans l'élément <updated>, peut-être parce que le serveur a d'avantage confiance dans sa propre horloge). C'est un des points importants d'APP.

Parmi les questions brûlantes d'APP figurait celle du *"lost update"* <<http://www.w3.org/1999/04/Editing/>>, la mise à jour perdue lorsque deux clients font presque en même temps un GET d'une ressource suivi d'un PUT avec la ressource modifiée. (L'excellent document du W3C indiqué plus haut, *"Editing the Web : Detecting the lost update problem using unreserved checkout"* <<http://www.w3.org/1999/04/Editing/>> explique très bien l'ensemble du problème et les solutions possibles.)

HTTP n'ayant pas d'état, et APP n'ayant pas d'option de verrouillage (contrairement à une autre extension de HTTP, WebDAV), la solution recommandée par notre RFC est celle du « PUT conditionnel » (section 9.5 du RFC) en utilisant la date ou bien l'"*etag*" et les en-têtes HTTP `If-Modified-Since` : ou `If-None-Match` :

Techniquement, c'est donc une solution assez simple mais la blogosphère a frémi <<http://www.tbray.org/ongoing/When/200x/2007/06/10/So-Lame>> sous les polémiques à ce sujet <<http://www.25hoursaday.com/weblog/2007/06/09/WhyGDataAPPFailsAsAGeneralPurposeEditingProtocolFor.aspx>>.

Parmi les verbes REST, on notera que PUT n'est utilisé que pour les mises à jour, pas les créations (le RFC 2616 autorise pourtant cet usage dans sa section 9.6). La question du « PUT pour créer » est toujours une question délicate dans le monde REST (voir <<http://bitworking.org/news/141/REST-Tips-Prefer-following-links-over-URI-construction>> ou bien <<http://www.megginson.com/blogs/quoderat/2005/04/03/post-in-rest-create-update-or-action/>>). Dans APP, comme l'espace de nommage est contrôlé par le serveur, il n'est pas prévu que le client puisse créer une ressource autrement qu'en passant par une collection (avec POST) et en laissant le serveur choisir un nom (le client peut toujours le guider avec le nouvel en-tête `Slug` : , décrit dans la section 9.7).

APP connaît déjà des extensions, la plus connue étant Gdata <<http://code.google.com/apis/gdata/index.html>>.

Parmi les mises en œuvres existantes d'APP, on notera, pour les serveurs :

- `mod_atom` <[http://www.tbray.org/ongoing/When/200x/2007/06/25/mod\\_atom](http://www.tbray.org/ongoing/When/200x/2007/06/25/mod_atom)>, un module Apache,
- `amplee` <<http://trac.defuze.org/wiki/amplee>>.

et pour les clients :

- GData Python Client Library <<http://code.google.com/p/gdata-python-client/>> est le client de référence de Google pour leur protocole Gdata, une extension d'APP,
- APP Test Client <<http://bitworking.org/projects/apptestclient/>>.

Pour en apprendre plus sur APP, on peut consulter l'excellent article "*Getting to know the Atom Publishing Protocol*" <<http://www.ibm.com/developerworks/library/x-atomppl/>>.