RFC 8620 : The JSON Meta Application Protocol (JMAP)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 6 septembre 2019

Date de publication du RFC : Juillet 2019

https://www.bortzmeyer.org/8620.html

Le protocole JMAP, "*JSON Meta Application Protocol*", permet de bâtir des mécanismes d'accès à des objets distants (par exemple des boîtes aux lettres, ou des agendas), en envoyant et recevant du JSON au dessus de HTTPS. Son principal « client » est le protocole « "*JMAP for mail*" », un concurrent d'IMAP normalisé dans le RFC 8621 [1].

Au début, JMAP était même conçu uniquement pour l'accès au courrier, comme l'indique son nom, qui évoque IMAP. Mais, dans le cadre de la normalisation de JMAP, est apparu le désir de séparer le protocole générique, adapté à toutes sortes d'objets distants, du protocole spécifique du courrier. D'où les deux RFC : ce RFC 8620 normalise le protocole générique, alors que le RFC 8621 normalise « "*JMAP for mail*" ». Dans le futur, d'autres protocoles fondés sur JMAP apparaitront peut-être, par exemple pour l'accès et la synchronisation d'un agenda (en concurrence avec le CalDAV du RFC 4791, donc).

Parmi les concepts techniques importants de JMAP, notons :
— Utilisation de JSON (RFC 8259) pour encoder les données, parce que tout le monde utilise JSON aujourd'hui,
— Transport sur HTTPS (RFC 2818 et RFC 7230), là encore comme tout le monde aujourd'hui, avec toutes les propriétés de sécurité de HTTPS (notamment, le client JMAP doit authentifier le serveur, typiquement via un certificat),
— Possibilité de regrouper ("*batching*") les requêtes en un seul envoi, pour les cas où la latence <https://www.bortzmeyer.org/latence.html> est élevée,
— Possibilité de notification par le serveur ("*push*"), pour éviter l'interrogation répétée par le client ("*polling*"),
— Un mécanisme de transfert incrémental des objets, pour limiter le débit sur le réseau.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8621.txt

Du fait de l'utilisation de JSON, il est bon de réviser le vocabulaire JSON, notamment le fait qu'un objet ("*object*") JSON est en fait un dictionnaire.

Outre les types de données de base de JSON, comme les booléens, les chaînes de caractères et les entiers, JMAP définit quelques types à lui, notamment le type `Id`, qui stocke l'identificateur d'un objet. Syntaxiquement, c'est une chaîne de caractères LDH ("*Letter-Digit-Hyphen*", lettres ASCII, chiffres et trait d'union). Par exemple, dans le RFC 8621, la première boîte aux lettres mentionnée a comme identificateur `MB23cfa8094c0f41e6`. Notre RFC crée aussi le type `Date`, puisque JSON ne normalise pas les dates. Ce type utilise le format du RFC 3339.

À ce stade, je peux avouer que j'ai fait un abus de langage. J'ai parlé de JSON mais en fait JMAP utilise un sous-ensemble de JSON, nommé I-JSON, et décrit dans le RFC 7493, afin d'éviter certaines ambiguités de JSON (section 8.4 de notre RFC). Tout le contenu échangé en JMAP doit être du I-JSON. D'ailleurs, si vous connaissez un logiciel libre qui vérifie qu'un texte JSON est du I-JSON, je suis preneur.

JMAP nécessite que le client se connecte au serveur (section 2 du RFC, sur la session). Pour cela, il lui faut l'URL du serveur (rappelez-vous que JMAP tourne sur HTTPS), qui peut être obtenu manuellement, ou par un processus de découverte décrit plus loin. Et il faut évidemment les moyens d'authentification (par exemple nom et phrase de passe), ceux-ci n'étant pas précisés dans notre RFC. Un mécanisme unique avait été prévu mais avait suscité trop de controverses à l'IETF. Finalement, les mécanismes utilisables sont ceux habituels de HTTPS (enregistrés à l'IANA `<https://www.iana.org/assignments/http-authschemes/>`). Lors de la connexion, le serveur va envoyer un objet JSON décrivant entre autres ses capacités, comme la taille maximale des objets téléversés, ou comme les extensions gérées (comme le « *"JMAP for mail"* » du RFC 8621). Voici un exemple (tiré du RFC, car je n'ai pas trouvé de serveur JMAP où je pouvais avoir facilement un compte pour tester, si vous en avez un, n'hésitez pas à m'écrire) :

```
   {
     "capabilities": {
       "urn:ietf:params:jmap:core": {
         "maxSizeUpload": 50000000,
         "maxSizeRequest": 10000000,
...
         "collationAlgorithms": [
           "i;ascii-numeric",
           "i;ascii-casemap",
           "i;unicode-casemap"
         ]
       },
       "urn:ietf:params:jmap:mail": {}
       "urn:ietf:params:jmap:contacts": {},
       "https://example.com/apis/foobar": {
         "maxFoosFinangled": 42
       }
     },
     "accounts": {
       "A13824": {
         "name": "john@example.com",
         "isPersonal": true,
         "isReadOnly": false,
         "accountCapabilities": {
           "urn:ietf:params:jmap:mail": {
             "maxMailboxesPerEmail": null,
             "maxMailboxDepth": 10,
             ...
           },
           "urn:ietf:params:jmap:contacts": {
             ...
           }
```

```
...
      "apiUrl": "https://jmap.example.com/api/",
...
```

Notez que ce serveur annonce qu'il sait faire du JMAP pour le courrier (RFC 8621, cf. la ligne `urn:ietf:params:jmap:mail`) et qu'il a également une extension privée, `https://example.com/apis/foobar`. Les capacités publiques sont dans un registre IANA <`https://www.iana.org/assignments/jmap/jmap.xml#jmap-capabilities`>. On peut ajouter des capacités par la procédure (cf. RFC 8126) « Spécification nécessaire » pour les capacités marquées « fréquentes » ("*common*"), et par la procédure « Examen par un expert » pour les autres.

La méthode standard pour découvrir le serveur JMAP, si on n'en connait pas l'URL, est d'utiliser un enregistrement SRV (RFC 2782, mais voir aussi le RFC 6186) puis un URL bien connu. Imaginons que le domaine soit `example.net`. On cherche le SRV pour `_jmap._tcp.example.net`. (`jmap` a donc été ajouté au registre des services <`https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml`>.) On récupère alors le nom du serveur et le port (a priori, ce sera 443, le port standard de HTTPS). Et on n'a plus qu'à se connecter à l'URL bien connu (RFC 8615), à `https://${hostname}[:${port}]/.well-known/jmap`. `jmap` figure à cet effet dans le registre des URL bien connus <`https://www.iana.org/assignments/well-known-uris/well-known-uris.xml#well-known-uris-1`>. (Notez que l'étape SRV est facultative, certains clients iront directement au `/.well-known/jmap`.) Ainsi, si vous utilisez JMAP pour le courrier, et que votre adresse est `gerard@example.net`, vous partez du domaine `example.net` et vous suivez l'algorithme ci-dessus. (Je ne sais pas pourquoi JMAP n'utilise pas plutôt le WebFinger du RFC 7033.)

Puisqu'on utilise le DNS pour récupérer ces enregistrements SRV, il est évidemment recommandé de déployer DNSSEC.

Une fois qu'on a récupéré le premier objet JSON décrit plus haut, on utilise la propriété (le membre, pour parler JSON) `apiUrl` de cet objet pour faire les requêtes suivantes (section 3 du RFC). On utilise la méthode HTTP `POST`, le type MIME `application/json`, et on envoie des requêtes en JSON, qui seront suivies de réponses du serveur, également en JSON. Les méthodes JMAP (à ne pas confondre avec les méthodes HTTP comme `GET` ou `POST`) sont écrites sous la forme `Catégorie/Méthode`. Il existe une catégorie `Core` pour les méthodes génériques de JMAP et chaque protocole utilisant JMAP définit sa (ou ses) propre(s) catégorie(s). Ainsi, le RFC 8621 définit les catégories `Mailbox`, `Email` (un message), etc. Comme `Core` définit une méthode `echo` (section 4, le ping de JMAP), qui ne fait que renvoyer les données, sans les comprendre, un exemple de requête/réponse peut être :

```
[[ "Core/echo", {
      "hello": true,
      "high": 5
}, "b3ff" ]]
```

```
[[ "Core/echo", {
      "hello": true,
      "high": 5
}, "b3ff" ]]
```

(Oui, la réponse - le second paragraphe - est identique à la question.)

En cas d'erreur, le serveur devrait renvoyer un objet décrivant le problème, en utilisant la syntaxe du RFC 7807. Une liste des erreurs connues figure dans un registre IANA <https://www.iana.org/assignments/jmap/jmap.xml#jmap-error-codes>.

Il existe des noms de méthodes standard qu'on retrouve dans toutes les catégories, comme `get`. Si on a une catégorie `Foo` décrivant un certain type d'objets, le client sait qu'il pourra récupérer les objets de ce type avec la méthode `Foo/get`, les modifier avec `Foo/set` et récupérer uniquement les modifications incrémentales avec `Foo/changes`. La section 5 du RFC décrit ces méthodes standard.

Une méthode particulièrement utile est `query` (section 5.5). Elle permet au client de demander au serveur de faire une recherche et/ou un tri des objets. Au lieu de tout télécharger et de faire recherche et tri soi-même, le client peut donc sous-traiter cette opération potentiellement coûteuse. Cette méthode est une de celles qui permet de dire que JMAP est bien adapté aux machines clientes disposant de faibles ressources matérielles, et pas très bien connectées. Le RFC cite (section 5.7) un type (imaginaire) `Todo` décrivant des tâches à accomplir, et l'exemple avec `query` permet d'illustrer le membre `filter` de la méthode, pour indiquer les critères de sélection :

```
[[ "Todo/query", {
       "accountId": "x",
       "filter": {
              "operator": "OR",
              "conditions": [
                         { "hasKeyword": "music" },
                         { "hasKeyword": "video" }
      ]
       }
    }
]]
```

Comme beaucoup de méthodes JMAP, `query` peut imposer un travail important au serveur. Un client maladroit, ou cherchant déliberement à créer une attaque par déni de service pourrait planter un serveur trop léger. Les serveurs JMAP doivent donc avoir des mécanismes de protection, comme une limite de temps passé sur chaque requête.

On l'a dit, un des intérêts de JMAP est la possibilité d'obtenir des notifications du serveur, sans obliger le client à vider sa batterie en interrogeant périodiquement le serveur. La section 7 du RFC détaille ce mécanisme. Deux alternatives pour le client : garder la connexion HTTPS ouverte en permanence, pour y recevoir ces notifications, ou bien utiliser un service tiers comme celui de Google. Notons que ces notifications, par leur seule existence, même si le canal est chiffré, peuvent révéler des informations. Comme noté dans la revue du protocole par la direction Sécurité à l'IETF ""*I.e., if someone can see that wikileaks smtp server sends email to corporate smtp server, but the smtp traffic is encrypted so they do not know the recipient of the email, but then few seconds later see push event notification stream going to the Joe's laptop indicating something has happened in his mail box, they can find out the who the recipient was.*"".

Il existe une page « officielle » présentant le protocole <https://jmap.io/> et plusieurs mises en oeuvre <https://jmap.io/software.html> (actuellement, la plupart sont, expérimentales et/ou en cours de développement). C'est une des raisons pour lesquelles je ne présente pas ici d'essais réels. Notez toutefois que Fastmail a du JMAP en production.