RFC 9727 : api-catalog: A Well-Known URI and Link Relation to Help Discovery of APIs

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 juin 2025

Date de publication du RFC : Juin 2025

https://www.bortzmeyer.org/9727.html

Les API réseau, c'est très bien. Tout le monde en fait aujourd'hui, typiquement bâties au-dessus de HTTP. Mais il est parfois difficile de s'y retrouver parmi toutes les API. « Je suis sûr que l'INSEE avait une API pour faire ça mais où est-elle exactement et où est sa documentation ? La solution pour cela ? Les catalogues d'API, rassemblant nom, description, URL, etc, des API. Il ne reste plus qu'à trouver le catalogue...Ce RFC fournit pour cela un URL standard et un mécanisme de lien. Il fournit également un format standard pour écrire ce catalogue, bâti sur Linkset (RFC 9264 [1], section 4.2).

Comme disait M. de la Palice, et comme le rappelle la section 1 du RFC, pour utiliser une API, il faut d'abord la découvrir. Cela veut dire apprendre son existence, quel va être l'URL à appeler (le "*endpoint*"), les paramètres à passer, quelles sont les conditions d'utilisation (gratuite ? nombre maximal de requêtes par jour ?), etc. Il existe des formats pour cela (le plus connu étant celui d'OpenAPI) mais il reste à trouver les fichiers de ce format. Pour faciliter cette tâche du développeur ou de la développeuse qui utilisera l'API, l'éditeur ("*publisher*" dans le RFC) qui publie l'API va fabriquer un joli catalogue, et le rendre lui-même découvrable. Un catalogue est une liste organisée des API offertes par une organisation donnée.

Première solution décrite par notre RFC (section 2) : un URL bien connu, et donc prévisible, `https://www.example.n` (si l'éditeur est en `www.example.net`). Ces URL bien connus sont normalisés dans le RFC 8615. Évidemment, une redirection HTTP est possible. `api-catalog` a été ajouté au registre des URL bien connus <https://www.iana.org/assignments/well-known-uris/well-known-uris.xml#well-known-uris-1>.

Deuxième solution, un lien (section 3) `api-catalog`. Les liens, vous connaissez, et ils sont normalisés dans le RFC 8288. Ils peuvent se représenter de plusieurs façons, en HTML :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc9264.txt`

```
<!DOCTYPE HTML>
<html>
   <head>
         <title>Welcome to Example Publisher</title>
   </head>
   <body>
         <p>
          <a href="my_api_catalog.json" rel="api-catalog">
           Example Publisher's APIs
          </a>
         </p>
         <p>(remainder of content)</p>
   </body>
</html>
```

Ou bien en HTTP, dans la réponse :

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Location: /index.html
Link: </my_api_catalog.json>; rel=api-catalog
Content-Length: 356
```

Désormais, `api-catalog` figure dans le registre des types de liens `<https://www.iana.org/assignments/link-relations/link-relations.xml#link-relations-1>`.

Cette norme concerne la découverte du catalogue, mais aussi la syntaxe de son contenu. La section 4 du RFC donne des indications sur le contenu du catalogue, sa sémantique (inclure la politique d'utilisation, lien vers une spécification formelle, etc) et sa syntaxe, le format Linkset du RFC 9264. Des exemples figurent dans l'annexe A du RFC mais vous pouvez aussi regarder le le catalogue des mes API `</.well-known/api-catalog>` (en construction). Voici le premier exemple de catalogue donné dans le RFC, utilisant les relations du RFC 8631 :

```
{
    "linkset": [
    {
      "anchor": "https://developer.example.com/apis/foo_api",
      "service-desc": [
        {
          "href": "https://developer.example.com/apis/foo_api/spec",
          "type": "application/yaml"
        }
      ],
      "service-doc": [
        {
          "href": "https://developer.example.com/apis/foo_api/doc",
          "type": "text/html"
        }
      ],
      "service-meta": [
        {
          "href": "https://developer.example.com/apis/foo_api/policies",
          "type": "text/xml"
        }
      ]
```

```
    },
    {
      "anchor": "https://developer.example.com/apis/bar_api",
      "service-desc": [
        {
          "href": "https://developer.example.com/apis/bar_api/spec",
          "type": "application/yaml"
        }
      ],
      "service-doc": [
        {
          "href": "https://developer.example.com/apis/bar_api/doc",
          "type": "text/plain"
        }
      ]
    }
  ]
}
```

D'autres formats que Linkset sont possible (mais Linkset doit être présent), par exemple via la négociation de contenu. Quelques formats possibles cités par le RFC :

— APIs.json <http://apisjson.org/>,
— RESTDesc <https://restdesc.org/>,
— HAL <https://datatracker.ietf.org/doc/draft-kelly-json-hal/>,
— WebAPI <https://webapi-discovery.github.io/rfcs/rfc0001.html>.

Ce catalogue peut, et ce serait souhaitable, être écrit automatiquement par le cadriciel qu'on utilise pour développer ses API.

Et, sinon, où est-ce qu'on publie cette API ? La méthode recommandée (section 5) est de la publier à plusieurs endroits. Si on gère `www.example.net` et que les URL des API sont en `api.example.com`, le RFC recommande de les publier aux deux endroits. Rappelez-vous que les redirections HTTP sont permises, donc vous pouvez n'avoir qu'un seul exemplaire du catalogue, ce qui simplifie sa maintenance. Car, puisqu'on parle de maintenance, un gros défi sera clairement de garder ce catalogue à jour au fur et à mesure de l'évolution des API. . .(La section 8, sur la sécurité, revient sur ce problème.)

Je n'ai pas trouvé d'exemple réel dans le Web, même chez Vodafone, où travaille l'auteur. Mais j'ai fait un catalogue pour mes propres API </.well-known/api-catalog>. En l'absence d'outils de test et de validation pour les catalogues, il a peut-être quelques erreurs. (Si vous lisez la section 2 du RFC, vous verrez que mon serveur HTTP ne répond pas aux requêtes HEAD avec un lien dans l'en-tête, un `Link:`. Il devrait.)