Valider du XML : exemple EPP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 octobre 2022. Dernière mise à jour le 21 octobre 2022

https://www.bortzmeyer.org/validate-epp.html

Cet article n'aura probablement pas beaucoup de lecteurs car peu de gens utilisent le protocole EPP. Ce dernier sert uniquement à la communication entre registres et bureaux d'enregistrement. Il utilise le format XML, et est décrit via un langage de schéma, ce qui permet sa validation par un programme. Comme ce n'est pas tout à fait évident, je montre ici comment on peut faire cette validation.

EPP est normalisé dans le RFC 5730 [1]. Mais attention, cela ne normalise que le cœur du protocole. EPP est un protocole d'avitaillement d'objets <https://www.afnic.fr/observatoire-ressources/papier-expert/que-se-passe-t-il-quand-on-enregistre-un-nom-de-domaine/> et il peut manipuler différents types d'objets, par exemple des noms de domaine. Il faut donc ajouter un RFC par type d'objet (pour les noms de domaine, c'est le RFC 5731) et à chaque fois un nouvel espace de noms. Et EPP a diverses extensions, avec à chaque fois un espace de noms. Tout cela est formellement décrit en XML Schema. Avantage : on peut valider un document EPP automatiquement et s'assurer qu'il est conforme à ce qu'on attend, avant de le traiter.

On va utiliser ici xmllint, qu'on trouve dans tous les bons systèmes d'exploitation. xmllint exige qu'on lui passe un fichier unique pour le schéma, donc je vous ai fait un joli fichier XML Schema qui inclut toutes les extensions auxquelles j'ai pensé, . Ensuite, on doit récupérer à l'IANA <https://www.iana.org/assignments/xml-registry/xml-registry.xml#schema> tous les fichiers `.xsd` concernant les différents schémas qu'on utilise. Cela peut se faire avec le script Python et cette boucle shell :

```
for schema in $(./extract-xsd.py epp-wrapper.xsd); do
    wget https://www.iana.org/assignments/xml-registry/schema/${schema}
done
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5730.txt

On peut ensuite valider un document EPP. Prenons celui-ci comme exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cela donne :

```
% xmllint --noout --schema epp-wrapper.xsd test.xml
test.xml validates
```

Parfait. Un programme qui va traiter les données peut, s'il a validé, être tranquille. Il sait par exemple qu'il aura une et une seule commande dans l'élément `<epp>`.

Notez que c'est en écrivant cet article qu'une faille a été trouvée `<https://www.rfc-editor.org/errata/eid7176>` dans le RFC 9167.

Si maintenant on prend un document EPP invalide (ajout d'un `<foobar>`) :

```
% xmllint --noout --schema epp-wrapper.xsd test-invalid.xml
test-invalid.xml:12: element foobar: Schemas validity error : Element '{urn:ietf:params:xml:ns:epp-1.0}fooba
test-invalid.xml fails to validate
```

Parfait encore, le document invalide est rejeté.

Si on utilise une extension à EPP comme celle pour DNSSEC du RFC 5910 :

```
<?xml version="1.0" encoding="utf-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <update>
      <domain:update
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
      </domain:update>
```

```
    </update>
    <extension>
      <secDNS:create xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
        <secDNS:dsData>
          <secDNS:keyTag>12345</secDNS:keyTag>
          <secDNS:alg>3</secDNS:alg>
          <secDNS:digestType>1</secDNS:digestType>
          <secDNS:digest>49FD46E6C4B45C55D4AC</secDNS:digest>
          <!-- <secDNS:keyData>, la clé elle-même, est *facultatif* -->
        </secDNS:dsData>
      </secDNS:create>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

La validation sera possible grâce à tous les schémas chargés :

```
% xmllint --noout --schema epp-wrapper.xsd test-dnssec.xml
test-dnssec.xml validates
```