RFC 8909 : Registry Data Escrow Specification

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 novembre 2020

Date de publication du RFC : Novembre 2020

https://www.bortzmeyer.org/8909.html

Prenons un registre, par exemple un registre de noms de domaine. Des tas de gens comptent sur lui. Si ce registre disparait en emportant sa base de données, ce serait une catastrophe. Il est donc nécessaire de garder les données à part, ce qu'on nomme un **séquestre** ("*escrow*"). Ce RFC décrit un format standard pour les données de séquestre, indépendant du type de registre (même si la demande prioritaire est pour les registres de noms de domaine). Il s'agit d'un format générique, qui sera complété par des spécifications pour les différents types de registre, par exemple le RFC 9022 [1] pour les registres de noms de domaine.

L'idée (section 1 du RFC) est donc que le registre dépose à intervalles réguliers une copie de sa base de données auprès de l'opérateur de séquestre. Il ne s'agit pas d'une sauvegarde. La sauvegarde est conçue pour faire face à des accidents comme un incendie. Le séquestre, lui, est conçu pour faire face à une disparition complète du registre, par exemple une faillite si le registre est une entreprise commerciale. Ou bien une redélégation complète du domaine à un autre registre. Les sauvegardes, non standardisées et liées à un système d'information particulier, ne peuvent pas être utilisées dans ces cas. D'où l'importance d'un format standard pour un séquestre, précisément ce que normalise notre RFC. Lors d'une défaillance complète de l'ancien registre, l'opérateur de séquestre transmettra les données de séquestre au nouveau registre, qui pourra l'importer dans son système d'information, qui sera alors prêt à prendre le relais. Pour un registre de noms de domaine, le dépôt des données de séquestre devra comprendre la liste des noms, les titulaires et contacts pour chacun des noms, les serveurs de noms, les enregistrements DS (utilisés pour DNSSEC), etc. Par contre, certaines informations ne pourront pas être incluses, comme les parties privées des clés utilisées pour DNSSEC (qui sont peut-être enfermées dans un HSM) et le nouveau registre aura donc quand même un peu de travail de réparation.

Ainsi, la convention <https://www.afnic.fr/fr/ressources/documents-de-reference/cadre-legal/convention-etat-afnic-pour-la-gestion-du-fr/> entre l'État et l'AFNIC pour la gestion du `.fr` prévoit dans sa section 5 que « L'Office d'enregistrement [sic] s'engage à mettre en place et à maintenir sur le sol français un séquestre de données quotidien <https://www.afnic.fr/medias/documents/Cadre_legal/Convention_Etat-AFNIC-2019.pdf> ». Même chose pour les TLD sous contrat avec l'ICANN. Le "*Base Registry Agreement*" <https://www.icann.org/resources/pages/registries/registries-agreements-en> impose un séquestre dans sa spécification 2 « "*DATA ESCROW REQUIREMENTS*" ». Elle dit « "*Deposit[Caractère Unicode non montré [2]]s Format.*

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc9022.txt
2. Car trop difficile à faire afficher par LaTeX

Registry objects, such as domains, contacts, name servers, registrars, etc. will be compiled into a file constructed as described in draft-arias-noguchi-registry-data-escrow, see Part A, Section 9, reference 1 of this Specification and draft-arias-noguchi-dnrd-objects-mapping, see Part A, Section 9, reference 2 of this Specification (collectively, the [Caractère Unicode non montré]DNDE Specification[Caractère Unicode non montré])." ≫. [Le document `draft-arias-noguchi-registry-data-escrow` est devenu ce RFC 8909.] L'ICANN reçoit actuellement 1 200 séquestres, pour chaque TLD qu'elle régule, les dépôts ayant lieu une fois par semaine.

Ces exigences sont tout à fait normales : la disparition d'un registre de noms de domaine, s'il n'y avait pas de séquestre, entrainerait la disparition de tous les noms, sans moyen pour les titulaires de faire valoir leurs droits. Si `.org` disparaissait sans séquestre, `bortzmeyer.org` n'existerait plus, et si un registre prenait le relais avec une base de données vide, rien ne me garantit d'obtenir ce nom, l'adresse de ce blog devrait donc changer. Cette question de **continuité** de l'activité de registre, même si des organisations disparaissent, est la motivation pour ce RFC (section 3).

Un exemple de service de « registre de secours » ("*third-party beneficiary*", dans la section 2 du RFC, on pourrait dire aussi "*backup registry*") est l'EBERO ("*Emergency Back-End Registry Operator*") de l'ICANN, décrit dans les « "*Registry Transition Processes*" `<https://www.icann.org/resources/pages/transition-processes-2013-04-22-en>` ».

Ce RFC ne spécifie qu'un format de données. Il ne règle pas les questions politiques (faut-il faire un séquestre, à quel rythme, qui doit être l'opérateur de séquestre, qui va désigner un registre de secours, etc).

Passons donc à la technique (section 5). Le format est du XML. L'élément racine est `<deposit>` (dépôt). Un attribut `type` indique si ce dépôt est complet (la totalité des données) ou bien incrémental (différence avec le dépôt précédent). L'espace de noms est `urn:ietf:params:xml:ns:rde-1.0`, enregistré à l'IANA `<https://www.iana.org/assignments/xml-registry/xml-registry.xml#ns>` (voir la section 8). RDE signifie "*Registry Data Escrow*". Parmi les éléments obligatoires sous `<deposit>`, il y a :

- `<watermark>` qui indique le moment où ce dépôt a été fait, au format du RFC 3339 (section 4). On peut traduire "*watermark*" par jalon, point de synchronisation, point d'étape ou marque.
- `<rdeMenu>`, diverses métadonnées.
- `<contents>` contient les données (rappelez-vous que ce RFC est générique, le format exact des données, qui dépend du type de registre, sera dans un autre RFC).
- `<deletes>` ne sert que pour les dépôts incrémentaux, et indique les objets qui ont été détruits depuis la dernière fois.
- Eh non, il n'y a pas d'élément `<adds>` ; dans un dépôt incrémental, les éléments ajoutés sont sous `<contents>`.

Voici un exemple très partiel d'un dépôt complet :

```
<?xml version="1.0" encoding="UTF-8"?>
<d:deposit xmlns:d="urn:ietf:params:xml:ns:rde-1.0"
   xmlns:myobj="urn:example:my-objects-1.0"
   type="FULL" id="20201006" resend="0">
  <d:watermark>2020-10-06T13:12:15Z</d:watermark>
  <d:rdeMenu>
    <d:version>1.0</d:version>
    <d:objURI>urn:example:my-objects-1.0</d:objURI>
  </d:rdeMenu>
  <d:contents>
    <myobj:object>
      <myobj:value>42</myobj:value>
    </myobj:object>
  </d:contents>
</d:deposit>
```

Dans cet exemple, le fichier XML contient un dépôt complet, fait le 6 octobre 2020, et dont le contenu est un seul élément, de valeur 42. (Pour un vrai dépôt lors d'un séquestre d'un registre de noms de domaine, on aurait, au lieu de `<myobj:object>`, les domaines, leurs titulaires, etc.) Vous avez des exemples un peu plus réalistes dans les sections 11 à 13 du RFC.

Attention si vous utilisez des dépôts incrémentaux, l'ordre des `<contents>` et `<deletes>` compte : il faut mettre les `<deletes>` en premier.

La syntaxe complète est décrite dans la section 6 du RFC, en XML Schema. Du fait de l'utilisation de XML, l'internationalisation est automatique, on peut par exemple mettre les noms des contacts en Unicode (section 7).

Le format décrit dans ce RFC est seulement un format. Des tas de questions subsistent si on veut un système de séquestre complet, il faut par exemple spécifier un mécanisme de transport des données entre le registre et l'opérateur de séquestre (par exemple avec SSH, ou bien un `POST` HTTPS). Pour l'ICANN, c'est spécifié dans l'"*Internet-Draft*" `draft-lozano-icann-registry-interfaces`.

Il faut également se préoccuper de tout ce qui concerne la sécurité. La section 9 du RFC rappelle que, si on veut que les données de séquestre soient confidentielles (pour un registre de noms de domaine, les coordonnées des titulaires et contacts sont certainement en bonne partie des données personnelles, cf. section 10), il faut chiffrer la communication entre le registre et l'opérateur de séquestre. Et il faut bien sûr tout authentifier. L'opérateur de séquestre doit vérifier que le dépôt vient bien du registre et n'est pas un dépôt injecté par un pirate, le registre doit vérifier qu'il envoie bien le dépôt à l'opérateur de séquestre et pas à un tiers. Comme exemple des techniques qui peuvent être utilisées pour atteindre ce but, l'ICANN cite OpenPGP (RFC 4880) : « "*Files processed for compression and encryption will be in the binary OpenPGP format as per OpenPGP Message Format - RFC 4880, see Part A, Section 9, reference 3 of this Specification. Acceptable algorithms for Public-key cryptography, Symmetric-key cryptography, Hash and Compression are those enumerated in RFC 4880, not marked as deprecated in OpenPGP IANA Registry, see Part A, Section 9, reference 4 of this Specification, that are also royalty-free.*" ».

Comme le séquestre ne servirait qu'en cas de terminaison complète du registre, on peut penser que le registre actuel n'est pas très motivé pour assurer ce service (et c'est pour cela qu'il doit être imposé). Il y a un risque de négligence (voire de malhonnêté) dans le production des dépôts. Voici pourquoi l'opérateur de séquestre **doit** tester que les dépôts qu'il reçoit sont corrects. Au minimum, il doit vérifier leur syntaxe. Ici, on va se servir de xmllint pour cela. D'abord, on utilise un schéma pour les données spécifiques de notre type de registre. Ici, il est très simple, uniquement des données bidon :

```
% cat myobj.xsd
<?xml version="1.0" encoding="utf-8"?>

<schema targetNamespace="urn:example:my-objects-1.0"
xmlns:myobj="urn:example:my-objects-1.0"
xmlns:rde="urn:ietf:params:xml:ns:rde-1.0"
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

    <annotation>
      <documentation>
        Test
      </documentation>
    </annotation>

    <element name="object" type="myobj:objectType" substitutionGroup="rde:content"/>
```

```
    <complexType name="objectType">
      <complexContent>
<extension base="rde:contentType">
  <sequence>
    <element name="value" type="integer"/>
  </sequence>
      </extension>
      </complexContent>
    </complexType>

</schema>
```

Ensuite, on écrit un petit schéma qui va importer les deux schémas, le nôtre (ci-dessus), spécifique à un type de registre, et le schéma du RFC, générique :

```
% cat wrapper.xsd
<?xml version="1.0" encoding="utf-8"?>

<schema targetNamespace="urn:example:tmp-1.0"
xmlns="http://www.w3.org/2001/XMLSchema">

  <import namespace="urn:ietf:params:xml:ns:rde-1.0" schemaLocation="rde.xsd"/>
  <import namespace="urn:example:my-objects-1.0" schemaLocation="myobj.xsd"/>

</schema>
```

Ensuite, on produit le dépôt :

```
% cat test.xml
<?xml version="1.0" encoding="UTF-8"?>
<d:deposit xmlns:d="urn:ietf:params:xml:ns:rde-1.0"
   xmlns:myobj="urn:example:my-objects-1.0"
   type="FULL" id="20201006" resend="0">
  <d:watermark>2020-10-06T13:12:15Z</d:watermark>
  <d:rdeMenu>
    <d:version>1.0</d:version>
    <d:objURI>urn:example:my-objects-1.0</d:objURI>
  </d:rdeMenu>
  <d:contents>
    <myobj:object>
      <myobj:value>42</myobj:value>
    </myobj:object>
  </d:contents>
</d:deposit>
```

Et on n'a plus qu'à valider ce dépôt :

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

Si les données étaient incorrectes (dépôt mal fait), xmllint nous préviendrait :

```
% xmllint --noout --schema wrapper.xsd test.xml
test.xml:12: element value: Schemas validity error : Element '{urn:example:my-objects-1.0}value': 'toto' is not
test.xml fails to validate
```

Idéalement, il faudrait même que l'opérateur de séquestre teste un chargement complet des données dans un autre logiciel de gestion de registre (oui, je sais, c'est beaucoup demander). Bon, s'il vérifie la syntaxe, c'est déjà ça.