RFC 8334 : Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 mars 2018

Date de publication du RFC : Mars 2018

https://www.bortzmeyer.org/8334.html

Les registres de noms de domaine ont parfois des périodes d'enregistrement spéciales, par exemple lors de la phase de lancement d'un nouveau domaine d'enregistrement, ou bien lorsque les règles d'enregistrement changent. Pendant ces périodes, les conditions d'enregistrement ne sont pas les mêmes que pendant les périodes « standards ». Les registres qui utilisent le protocole EPP pour l'enregistrement peuvent alors utiliser les extensions EPP de ce nouveau RFC pour gérer ces périodes spéciales.

Un exemple de période spéciale est l'ouverture d'un tout nouveau TLD à l'enregistrement. Un autre exemple est une libéralisation de l'enregistrement, passant par exemple de vérifications a priori strictes à un modèle plus ouvert. Dans les deux cas, on peut voir des conflits se faire jour, par exemple entre le titulaire le plus rapide à enregistrer un nom, et un détenteur de propriété intellectuelle qui voudrait reprendre le nom. Les périodes spéciales sont donc définies par des privilèges particuliers pour certains utilisateurs, permettant par exemple aux titulaires d'une marque déposée d'avoir un avantage pour le nom de domaine correspondant à cette marque. La période spéciale est qualifiée de « phase de lancement » ("*launch phase*"). Les extensions à EPP décrites dans ce nouveau RFC permettent de mettre en œuvre ces privilèges.

La classe ("*mapping*") décrivant les domaines en EPP figure dans le RFC 5731 [1]. Elle est prévue pour le fonctionnement standard du registre, sans intégrer les périodes spéciales. Par exemple, en fonctionnement standard, une fois que quelqu'un a enregistré un nom, c'est fini, personne d'autre ne peut le faire. Mais dans les phases de lancement, il arrive qu'on accepte plusieurs candidatures pour un même nom, qui sera ensuite attribué en fonction de divers critères (y compris parfois une mise aux enchères). Ou bien il peut y avoir des vérifications supplémentaires pendant une phase de lancement. Par exemple,

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5731.txt

certaines phases peuvent être réservées aux titulaires de propriété intellectuelle, et cela est vérifié via un organisme de validation, comme la TMCH (RFC 7848).

D'où ce RFC qui étend la classe `domain` du RFC 5731. La section 2 du RFC décrit les nouveaux attributs et éléments des domaines, la section 3 la façon de les utiliser dans les commandes EPP et la section 4 donne le schéma XML. Voyons d'abord les nouveaux éléments et attributs.

D'abord, comme il peut y avoir plusieurs candidatures pour un même nom, il faut un moyen de les distinguer. C'est le but de l'identificateur de candidature ("*application identifier*"). Lorsque le serveur EPP reçoit une commande `<domain:create>` pour un nom, il attribue un identificateur de candidature, qu'il renvoie au client, dans un élément `<launch:applicationID>`, tout en indiquant que le domaine est en état `pendingCreate` (RFC 5731, section 2.3) puisque le domaine n'a pas encore été créé. Au passage, `launch` dans `<launch:applicationID>` est une abréviation pour l'espace de noms XML `urn:ietf:params:xml:ns:launch-1.0`. Un processeur XML correct ne doit évidemment pas tenir compte de l'abréviation (qui peut être ce qu'on veut) mais uniquement de l'espace de noms associé. Cet espace est désormais enregistré à l'IANA `<https://www.iana.org/assignments/xml-registry/xml-registry.xml>` (cf. RFC 3688).

Autre nouveauté, comme un serveur peut utiliser plusieurs organismes de validation d'une marque déposée, il existe désormais un attribut `validatorID` qui indique l'organisme. Par défaut, c'est la TMCH (identificateur `tmch`). On pourra utiliser cet attribut lorsqu'on indiquera un identificateur de marque, par exemple lorsqu'on se sert de l'élément `<mark:mark>` du RFC 7848.

Les périodes spéciales ont souvent plusieurs phases, et notre RFC en définit plusieurs (dans une ouverture réelle, toutes ne sont pas forcément utilisées), qui seront utilisées dans l'élément `<launch:phase>` :

— Lever de soleil ("*sunrise*"), phase où les titulaires de marques (le RFC ne mentionne pas d'autres cas, comme le nom de l'entreprise ou d'une ville) peuvent seuls soumettre des candidatures, la marque étant validée par exemple via la TMCH,
— Prétentions ("*claims*"), où on peut enregistrer si on n'a pas de marque, mais on reçoit alors une notice disant qu'une marque similaire existe (elle est décrite plus en détail dans l'"*Internet-Draft*" `draft-ietf-regext-tmch-func-spec`), et on peut alors renoncer ou continuer (si on est d'humeur à affronter les avocats de la propriété intellectuelle), en annonçant, si on continue « oui, j'ai vu, j'y vais quand même »,
— Ruée ("*landrush*"), phase immédiatement après l'ouverture, quand tout le monde et son chien peuvent se précipiter pour enregistrer,
— État ouvert ("*open*"), une fois qu'on a atteint le régime de croisière.

La section 2 définit aussi les états d'une candidature. Notamment :

— `pendingValidation` (validation en attente),
— `validated` (c'est bon, mais voyez plus loin),
— `invalid` (raté, vous n'avez pas de droits sur ce nom),
— `pendingAllocation` (une fois qu'on est validé, tout n'est pas fini, il peut y avoir plusieurs candidatures, avec un mécanisme de sélection, par exemple fondé sur une enchère),
— `allocated` (c'est vraiment bon),
— `rejected` (c'est fichu…)

Les changements d'état ne sont pas forcément synchrones. Parfois, il faut attendre une validation manuelle, par exemple. Dans cas, il faut notifier le client EPP, ce qui se fait avec le mécanisme des messages asynchrones ("*poll message*") du RFC 5730, section 2.9.2.3.

Comme toutes les extensions EPP, elle n'est utilisée par le client que si le serveur l'indique à l'ouverture de la session, en listant les espaces de noms XML des extensions qu'il accepte, par exemple :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
      <greeting><svID>EPP beautiful server for .example</svID>
                <svDate>2018-02-20T15:37:20.0Z</svDate>
                <svcMenu><version>1.0</version><lang>en</lang>
                <objURI>urn:ietf:params:xml:ns:domain-1.0</objURI>
                <objURI>urn:ietf:params:xml:ns:contact-1.0</objURI>
                <svcExtension>
                <extURI>urn:ietf:params:xml:ns:rgp-1.0</extURI>
                <extURI>urn:ietf:params:xml:ns:secDNS-1.1</extURI>
                <extURI>urn:ietf:params:xml:ns:launch-1.0</extURI>
                </svcExtension>
                </svcMenu>
        </greeting>
   </epp>
```

Maintenant qu'on a défini les données, la section 3 du RFC explique comment les utiliser. (Dans tous les exemples ci-dessous, `C:` identifie ce qui est envoyé par le client EPP et `S:` ce que le serveur répond.) Par exemple, la commande EPP `<check>` (RFC 5730, section 2.9.2.1) sert à vérifier si on peut enregistrer un objet (ici, un nom de domaine). Elle prend ici des éléments supplémentaires, par exemple pour tester si un nom correspond à une marque. Ici, on demande si une marque existe (notez l'extension `<launch:check>`) :

```
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <check>
C:    <domain:check
C:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>domain1.example</domain:name>
C:    </domain:check>
C:   </check>
C:   <extension>
C:    <launch:check
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:     type="trademark"/>
C:   </extension>
C:  </command>
C:</epp>
```

Et on a la réponse (oui, la marque existe dans la TMCH) :

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:     <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:     <launch:chkData
S:      xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:      <launch:cd>
S:        <launch:name exists="1">domain1.example</launch:name>
S:        <launch:claimKey validatorID="tmch">
S:        2013041500/2/6/9/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
S:        </launch:claimKey>
```

```
S:      </launch:cd>
S:     </launch:chkData>
S:    </extension>
S:  </response>
S:</epp>
```

Avec la commande EPP `<info>`, qui sert à récupérer des informations sur un nom, on voit ici qu'un nom est en attente (`pendingCreate`), et on a l'affichage de la phase actuelle du lancement, dans l'élément `<launch:phase>` :

```
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <info>
C:    <domain:info
C:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>domain.example</domain:name>
C:    </domain:info>
C:   </info>
C:   <extension>
C:    <launch:info
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:       includeMark="true">
C:      <launch:phase>sunrise</launch:phase>
C:    </launch:info>
C:   </extension>
C:  </command>
C:</epp>
```

Et le résultat, avec entre autre l'identificateur de candidature :

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:status s="pendingCreate"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
   ...
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:          <launch:applicationID>abc123</launch:applicationID>
S:          <launch:status s="pendingValidation"/>
S:          <mark:mark
S:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
S:             ...
S:         </mark:mark>
S:      </launch:infData>
```

```
S:    </extension>
S:  </response>
S:</epp>
```

C'est bien joli d'avoir des informations mais, maintenant, on voudrait créer des noms de domaine. La commande EPP `<create>` (RFC 5730, section 2.9.3.1) sert à cela. Selon la phase de lancement, il faut lui passer des extensions différentes. Pendant le lever de soleil ("*sunrise*"), il faut indiquer la marque déposée sur laquelle on s'appuie, dans `<launch:codeMark>` (il y a d'autres moyens de l'indiquer, cf. section 2.6) :

```
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
   ...
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <launch:code validatorID="sample1">
C:            49FD46E6C4B45C55D4AC</launch:code>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:  </command>
C:</epp>
```

On reçoit une réponse qui dit que le domaine n'est pas encore créé, mais on a un identificateur de candidature (un numéro de ticket, quoi) en `<launch:applicationID>`. Notez le code de retour 1001 (j'ai compris mais je ne vais pas le faire tout de suite) et non pas 1000, comme ce serait le cas en régime de croisière :

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <domain:creData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:       <domain:name>domain.example</domain:name>
S:       <domain:crDate>2010-08-10T15:38:26.623854Z</domain:crDate>
S:      </domain:creData>
S:    </resData>
S:    <extension>
S:      <launch:creData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>2393-9323-E08C-03B1
```

```
S:        </launch:applicationID>
S:      </launch:creData>
S:    </extension>
S:  </response>
S:</epp>
```

De même, des extensions permettent de créer un domaine pendant la phase où il faut indiquer qu'on a vu les prétentions qu'avait un titulaire de marque sur ce nom. Le RFC décrit aussi l'extension à utiliser dans la phase de ruée ("*landrush*"), mais j'avoue n'avoir pas compris son usage (puisque, pendant la ruée, les règles habituelles s'appliquent).

On peut également retirer une candidature, avec la commande EPP `<delete>` qui, en mode standard, sert à supprimer un domaine. Il faut alors indiquer l'identifiant de la candidature qu'on retire :

```
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <delete>
C:    <domain:delete
C:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>domain.example</domain:name>
C:    </domain:delete>
C:   </delete>
C:   <extension>
C:    <launch:delete
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:      <launch:phase>sunrise</launch:phase>
C:      <launch:applicationID>abc123</launch:applicationID>
C:    </launch:delete>
C:   </extension>
C:  </command>
C:</epp>
```

Et les messages non sollicités ("*poll*"), envoyés de manière asynchrone par le serveur ? Voici un exemple, où le serveur indique que la candidature a été jugée valide (le mécanisme par lequel on passe d'un état à un autre dépend de la politique du serveur) :

```
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>2013-04-04T22:01:00.0Z</qDate>
S:      <msg>Application pendingAllocation.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        ...
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <launch:infData
```

```
S:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>abc123</launch:applicationID>
S:        <launch:status s="pendingAllocation"/>
S:      </launch:infData>
S:    </extension>
S:  </response>
S:</epp>
```

Voilà, vous savez l'essentiel, si vous voulez tous les détails, il faudra lire la section 3 complète, ainsi que la section 4, qui contient le schéma XML des extensions pour les phases de lancement. Comme toutes les extensions à EPP, celle de ce RFC est désormais dans le registre des extensions EPP <https://www.iana.org/assignments/epp-extensions/epp-extensions.xml>, décrit dans le RFC 7451.

Notez que ce RFC ne fournit pas de moyen pour indiquer au client EPP quelle est la politique d'enregistrement pendant la période spéciale. Cela doit être fait par un mécanisme externe (page Web du registre, par exemple).

Quelles sont les mises en œuvre de ce RFC ? L'extension pour les phases de lancement est ancienne (première description en 2011) et de nombreux registres offrent désormais cette possibilité. C'est d'autant plus vrai que l'ICANN impose aux registres de ses nouveaux TLD de gérer les phases de lancement avec cette extension. Ainsi :

- Le kit de développement de clients EPP <http://www.verisigninc.com/en_US/channel-resources/domain-registry-products/epp-sdks> de Verisign a cette extension, sous une licence libre.
- Logiquement, le système d'enregistrement de Verisign, utilisé notamment pour `.com` et `.net` (mais également pour bien d'autres TLD) gère cette extension (code non libre et non public, cette fois).
- Le serveur REngin <http://dns.net.za/rengin/> (non libre), utilisé pour `.za` a aussi cette extension.
- Le serveur de CentralNIC, pareil.
- Le client EPP distribué par Neustar est sous licence libre et sait gérer les phases de lancement.
- Le serveur (non libre) de SIDN (qui gère le domaine national `.nl`) fait partie de ceux qui ont mis en œuvre ce RFC.
- Et côté client, il y a le logiciel libre Net : :DRI <https://github.com/thedarkwinter/Net-DRI> (extension ajoutée dans une scission nommée `tdw`, pas dans le logiciel originel de Patrick Mevzek), cf. `LaunchPhase.pm`.