Network Working Group A. Sullivan
Internet-Draft Afilias
Expires: January 2, 2006 July 2005

Some experiences from implementing the Extensible Provisioning Protocol
draft-sullivan-epp-experience-00

Status of this Memo

Copyright Notice

Abstract

This memo presents some experiences of a registry operator using the Extensible Provisioning Protocol (EPP). Some limitations of the protocol definition and associated documents are identified, and some alterations are suggested.

Table of Contents

1. Introduction

The Extensible Provisioning Protocol (EPP) was first published as [RFC3730] in March 2004, along with an associated Domain Name Mapping ([RFC3731]), Host Mapping ([RFC3732]), and Contact Mapping ([RFC3733]). In September 2004, an additional Mapping, for the Domain Registry Grace Period, was published as [RFC3915].

While EPP is a generic object-provisioning protocol for repositories, its initial application (and the impetus for its development) was in provisioning domain names for registration to DNS zones. The concurrent publication of Domain Name and Host Mappings is consistent with that use. Because of that history, the object status and command definitions in [RFC3731] include limitations that express the community understanding of best domain name registration practices at the time of publication.

Subsequent experience has revealed a number of areas where the protocol could use improvement in order to offer the generic provisioning services it intends to offer.

1. The creation of the "Redemption Grace Period" (supported under EPP by [RFC3915] requires that certain provisions of [RFC3731] be violated in order to offer the new service.

2. Experience with implementation suggests that status values may be insufficiently granular.

3. Experience with the poll queue mechanism suggests that a more sophisticated mechanism might be useful, or that the requirements for implementation should be relaxed.

4. Experience with <update> commands suggests either that status values are too coarsely-grained, or that another set of commands entirely needs to be defined.

Each of these limitations is discussed in more detail below. Each one would be serious in its own right; but the overall effect of the combined limitations is such that the protocol is not as generic as it aims to be. To the extent that EPP does not provide the flexibility that registries need to accommodate their business objectives and procedures, registries will adopt other protocols. If that happens, the original goal of a common registry protocol will not be met.

2. Experiences

2.1 Experiences at Afilias

Afilias began experimenting with EPP on the basis of the Internet Drafts from the provreg working group as early as 2001. Afilias provided a working EPP implementation for use by registrar-customers by 28 September 2004. These experiences have led to the conclusion that there are some unfortunate inherent limitations to the protocols as written; these are outlined below.

1. While adding extensions to the functionality defined in the protocol and mapping RFCs is relatively simple, extending those documents in ways that were not foreseen when they were written has proven to be either difficult or impossible. A good example of this is the conflict between [RFC3731] and [RFC3915] (see Section 3.1). Afilias's own attempt at implementing ICANN's Redemption Grace Period, prior to the creation of [RFC3915], also required a violation of [RFC3731]: allowing a domain to be "undeleted" after a <domain:delete> command has been successfully processed is simply in violation of the specification.

2. The message queue and <poll> command are too limited in the functionality they offer to be very useful to many users, so the message queue content all has to be duplicated using some other notification mechanism. For example, in Afilias's experience, many clients continue to rely on emailed notices to alert them of transfer requests, even though the message queue contains the information. That is partly because email is easily filtered, and can therefore cause certain notices to be processed ahead of others. The message queue, by contrast, can cause a notice of a pending transfer to wait until the delivery and acknowledgement of messages noting the successful completion of a different transfer. (More on the matter of this limitation is discussed below; see Section 3.4.)

3. Inter-registrar host and nameserver issues are a problem. The EPP RFCs are written in such a way that many alternative policies are foreclosed, so a solution, if it ever comes up, will require changes to the protocol itself or to the mapping RFCs. It is undesirable for the protocol to dictate policy this way.

4. The <update> command does not really behave in practice in the way that other object-manipulation commands do. As a practical matter, <update> alters one of several object attributes, whereas most transform operations modify either the whole object, or a predictable and small number of the object's attributes. For example, where <domain:transfer> always directly modifies at most

the domain object's expiration date and sponsor, <domain:update> may modify attributes as diverse as the domain's name servers, contacts, authInfo, or even (in the case of [RFC3915]) pendingDelete status. Yet a prohibition on update remains a prohibition of any update to the object, even though only one or two attributes may be the target of the prohibition.

3. Mapping flexibility limitations

3.1 RFC 3731 vs RFC 3915

One of the status values that [RFC3731] defines for a domain object is pendingDelete. In many cases, repository policy requires additional activity before a <domain:delete> command can be processed. In many domain repositories, some period of time must elapse before the processing is completed. Under such conditions, the <domain:delete> receives a response of 1001 (Command completed successfully; action pending: see [RFC3730]), and the off-line activity is invoked. This activity may simply be the passage of time.

Transform commands against an object with pendingDelete status set are required to be rejected, per [RFC3731]: "With one exception, transform commands MUST be rejected when a pendingCreate, pendingDelete, pendingRenew, pendingTransfer, or pendingUpdate status is set."

In effect, then, when a <domain:delete> command is issued against a domain, there are two possibilities for activity in the repository: either the domain object is immediately deleted from the repository (with response 1000), or the domain object has the pendingDelete status set, and the client software receives response code 1001.

In order to support the ICANN redemption grace period, [RFC3915] extends [RFC3731] to provide the ability to cause the domain to be restored to the repository. The difficulty is that such an <update> command (which is a transform command) is required to fail, because of the pendingDelete status on the object. In addition, [RFC3915] is not allowed to extend [RFC3731] in a way inconsistent with the existing [RFC3731], because that is not part of the extension mechanism defined by [RFC3730]: "Protocol commands and responses MAY be extended by an <extension> element that contains additional elements whose syntax and semantics are not explicitly defined by EPP or an EPP object mapping." Therefore, either [RFC3731] must change, or [RFC3915] is unimplementable.

The above would be nothing more than a quibble, except that this sort of trouble will occur in any case where extensions are desired, and the extension intends to alter the behaviour defined by the parent mapping. Following are some more examples of possible use-cases foreclosed by the existing documents.

3.2 Alternative policies for handling nameservers

It might be desirable to foster some sort of off-line negotiation or processing of <host:delete> commands when a host is associated with a domain as its nameserver. No one to our knowledge has yet proposed a nameserver-management regime that solves all the resolution and lame delegation problems in DNS; but the relationship between hosts and domains, as described in [RFC3731] and [RFC3732], more or less enshrines lame delegation as a fact of life, and provides no allowance for policies that require that <delete> commands cause additional processing to ensure correct DNS entries.

3.3 Granularity of the <update> command

It is frequently the case that one desires to limit some subset of <update> commands. EPP <update> commands allow client software to make changes to only particular elements of the manipulated object; but the serverUpdateProhibited and clientUpdateProhibited status values block any update from being applied to the object with that status. This hampers the opportunities for control of an object. For instance, a registry might reasonably wish to implement a policy under which name servers MAY be changed on a domain object but no other changes are permitted. The serverUpdateProhibited status would not allow such a change, but no other status is available to enforce the desired behaviour.

In Afilias's experience, for instance, it has been occasionally desirable to prohibit changes to registrant data associated with a domain during a period in which non-technical considerations were being satisfied. In such cases, Afilias had to choose between, on the one hand, violating the integrity of the serverUpdateProhibited status value on the domain object in order to allow client software to manage nameserver associations; and prohibiting nameserver management for domains that had unresolved non-technical issues, on the other.

Creating support for more granular controls of <update> breaks the symmetry between prohibited status values and commands. But since the EPP <update> command is so much more versatile than the other commands, a different approach is needed to manage it.

One might argue that a server that needs more granular control should not use serverUpdateProhibited, and should instead define the more granular prohibitions in an extension. This is an appealing idea, but has the disadvantage that every repository operator may come up with a different extension. These divergent extensions will not cleave to the original goal of EPP: the desire was for client software to be able to work more or less unchanged from one repository to another. That said, it is easy to appreciate that breaking the direct correlation between a command and its prohibition would be at least counterintuitive, and at worst could cause trouble with extending the base documents in the same way as discussed in Section 3.1.

3.4 Message queue priority

It is sometimes desirable to offer service messages in a priority-based order. The description of the EPP <poll> command in [RFC3730] makes the server message queue a FIFO queue. Some service messages, however, are of greater urgency than others, or may be of greater interest to clients.

There may be reasons to prefer the simple (FIFO) nature of the message queue design in some cases. Unfortunately, [RFC3730] also requires that service messages MUST be generated "for all clients affected by an action on an object," where that action either is performed on an object not in direct response to a client request, or where the actions of one client may indirectly affect a second client. This requirement tends to flood the message queue with a large number of messages that are not at all urgent for clients, and further reduces the value of the message queue to the clients. For instance, Afilias's experience indicates that clients are much more sensitive to transactions that entail monetary loss than those that do not. The requirements of the message queue, however, may mean that a large number of notices of domain objects completing their pendingDelete period (and therefore disappearing from the registry) be acknowledged by the client, and dequeued, before the client can find out that a large number of automatic domain renewals have been processed (resulting in financial charges to the client). As a result, Afilias has found itself in the position of having to choose between conformance with the letter of the protocol, and satisfying client demands.

The additional provision that "Servers MAY implement other mechanisms to dequeue and deliver messages" is, moreover, just confusing. Perhaps it makes the message queue requirement OPTIONAL, but in a round-about manner. If that is the case, it would be preferable to make the <poll> command OPTIONAL, or make generation of service messages OPTIONAL in some cases.

4. Suggested alterations

In principle, since EPP is extensible, each of the troublesome cases (as well as any not enumerated here) could be rectified by an extension. In the cases above, the extensions mechanism unfortunately does not permit the desired extensions, because the extension would be in contravention of the base document. This restriction on extensions seems to be sensible, because it is paradoxical to violate a base document in order to extend it.

It appears, however, that some improvements can be made by making the protocol less strict, on the grounds that many cases currently defined by the protocol or mapping documents are in fact cases of server policy.

4.1 Status values and prohibitions

4.1.1 Suggestions

In general, status values should be aligned with exactly one effect on a repository object.

For instance, the prohibition against deleting hosts associated with any other object should be lifted, and the matter relegated to repository policy. If the repository wishes to restrict deletion of a host object when the host is associated with any other object, the repository may use the status serverDeleteProhibited. This approach is generalisable: there is no reason that pending* status values need prohibit further action, although in most cases it will be desirable for the server to set prohibitive status values on objects with pending action. Nevertheless, that a domain object is pendingTransfer is no reason to prohibit, at the protocol level, updating its name server associations. By removing the restriction on what may happen when a domain has its pendingDelete status set, the conflict between [RFC3731] and [RFC3915] is resolved.

For similar reasons, it is desirable to align the list of prohibited status values to correspond with the actions that may be taken against a repository object. For instance, serverUpdateProhibited and clientUpdateProhibited status values on a domain object prevent a very wide array of activities. If we break apart the serverUpdateProhibited and clientUpdateProhibited status values in [RFC3731] such that each resulting status prohibits exactly one thing, we get the following:

clientUpdateNSProhibited,serverUpdateNSProhibited:

Requests to update the object nameserver associations MUST be rejected.

clientUpdateContactProhibited, serverUpdateContactProhibited:

Requests to update the object contact associations MUST be rejected.

clientUpdateStatusProhibited, serverUpdateStatusProhibited:

Requests to update the object status values MUST be rejected, except for requests to remove this status.

clientUpdateRegistrantProhibited, serverUpdateRegistrantProhibited:

Requests to change the object registrant MUST be rejected.

clientUpdateAuthInfoProhibited, serverUpdateAuthInfoProhibited:

Requests to change the object authInfo MUST be rejected.

This approach also has the happy consequence that extending (in this case) the domain mapping with additional commands suggests a straightforward addition to the status values as well. In this way, the mapping remains easy to understand.

4.1.2 Objections and alternatives

One might object to narrowing the applicability of each status value as being cumbersome and complicated. On such a view, it is preferable to grant that some status values entail that certain other actions are foreclosed, for the sake of the simplicity of implementation. But while this is true in many cases, it need not be true in every case. Moreover, this foreclosure is really a matter of server policy and not of protocol, and so should be left up to the implementor, even if it makes the protocol marginally more difficult to implement.

One might object to the addition of attribute-specific updateProhibited values, on the grounds of consistency: the other prohibitions are all command-based. Commands aside from update, however, generally do only one thing: a domain is either deleted or not, according to whether the command completes or fails. Only <update> is special, in that it updates parts of an object and leaves the rest untransformed. That said, the proposed modification is admittedly an ugly one. Additional evidence of need is probably warranted before making a change here.

4.2 Polling

4.2.1 A priority indication and response

Finally, in order to improve the utility of the EPP <poll> command, it is desirable to add one OPTIONAL "priority" attribute to the <poll> command. In case the attribute is not present, the value of the "priority" attribute is 0. If the message queue is not empty, a successful response to a <poll> command MUST return the first message from the message queue having an identical priority. If the value of the "priority" attribute is the special value 1000, the command MUST return the first message from the message queue having the highest priority.

Example modified <poll> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
C:     epp-1.0.xsd">
C:  <command>
C:    <poll op="req"
C:          priority="1"/>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

A <poll> acknowledgement response notes the number of messages remaining in the queue and the ID of the next message available for retrieval for each priority:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
S:     epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <msgQ priority="0" count="4" id="12346"/>
S:    <msgQ priority="1" count="1" id="8910"/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

4.2.2 Other polling approaches

One might object to the complication of the poll queue mechanism. But there is nothing that requires implementers to use a priority other than "0", so a single FIFO queue is always still available to implementers if that is preferable.

5. Internationalization Considerations

This memo introduces no internationalization considerations beyond those in [RFC3730].

6. IANA Considerations

This memo introduces no IANA considerations beyond those in [RFC3730].

7. Security Considerations

This memo introduces no security considerations beyond those in [RFC3730].

8. Acknowledgements

The author wishes to thank Scott Hollenbeck and John Klensin for some very helpful remarks on predecessors to this document. Howard Eland, Janusz Sienkiewicz, and Michael Young each provided remarks about Afilias's experience in implementing EPP. Any mangling of their clear thoughts is the responsibility of the author.

9. References

[RFC3730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", RFC 3730, March 2004.

[RFC3731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", RFC 3731, March 2004.

[RFC3732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", RFC 3732, March 2004.

[RFC3733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", RFC 3733, March 2004.

[RFC3734] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Transport Over TCP", RFC 3734, March 2004.

[RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, September 2004.

Author's Address

Andrew J. Sullivan
Afilias
204-4141 Yonge Street
Toronto, ON M2P 2A8
CA

Phone: +1 416 646 3304
Email: andrew@ca.afilias.info

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.