

eppext
Internet-Draft
Intended status: Standards Track
Expires: December 2, 2016

H.W. Ribbers
M.W. Groeneweg
SIDN
R. Gieben

A.L.J Verschuren

May 31, 2016

Key Relay Mapping for the Extensible Provisioning Protocol
draft-ietf-eppext-keyrelay-12

Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for a key relay object that relays DNSSEC key material between EPP clients using the poll queue defined in RFC5730.

This key relay mapping will help facilitate changing the DNS operator of a domain while keeping the DNSSEC chain of trust intact.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 2, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	3
1.2. Secure Transfer of DNSSEC Key Material	3
2. Object Attributes	4
2.1. DNSSEC Key Material	5
2.1.1. <keyRelayData> element	5
3. EPP Command Mapping	5
3.1. EPP Query Commands	5
3.1.1. EPP <check> Command	6
3.1.2. EPP <info> Command	6
3.1.3. EPP <transfer> Command	9
3.2. EPP Transform Commands	9
3.2.1. EPP <create> Command	9
3.2.2. EPP <delete> Command	11
3.2.3. EPP <renew> Command	11
3.2.4. EPP <transfer> Command	12
3.2.5. EPP <update> Command	12
4. Formal Syntax	12
5. IANA Considerations	13
5.1. XML Namespace	13
5.2. XML Schema	13
5.3. EPP Extension Registry	14
6. Security Considerations	14
7. Acknowledgements	15
8. References	15
8.1. Normative References	15
8.2. Informative References	15
Appendix A. Changelog	16
A.1. draft-gieben-epp-keyrelay-00	16
A.2. draft-gieben-epp-keyrelay-01	16
A.3. draft-gieben-epp-keyrelay-02	16
A.4. draft-gieben-epp-keyrelay-03	16
A.5. draft-ietf-eppext-keyrelay-00	17
A.6. draft-ietf-eppext-keyrelay-01	17
A.7. draft-ietf-eppext-keyrelay-02	17
A.8. draft-ietf-eppext-keyrelay-03	17
A.9. draft-ietf-eppext-keyrelay-04	17
A.10. draft-ietf-eppext-keyrelay-05	18
A.11. draft-ietf-eppext-keyrelay-06	18
A.12. draft-ietf-eppext-keyrelay-07	18

A.13. draft-ietf-eppext-keyrelay-08	18
A.14. draft-ietf-eppext-keyrelay-09	18
A.15. draft-ietf-eppext-keyrelay-10	18
A.16. draft-ietf-eppext-keyrelay-11	18
A.17. draft-ietf-regext-keyrelay-00	18
Authors' Addresses	18

1. Introduction

There are certain transactions initiated by a DNS-operator that require an authenticated exchange of information between DNS-operators. Often, there is no direct channel between these parties or it is non-scalable and insecure.

One such transaction is the exchange of DNSSEC key material when changing the DNS operator for DNSSEC signed zones. We suggest that DNS-operators use the administrative EPP channel to bootstrap the delegation by relaying DNSSEC key material for the zone.

In this document we define an EPP extension to sent DNSSEC key material between EPP clients. This allows DNS operators to bootstrap automatically, reliable and securely the transfer of a domain name while keeping the DNSSEC chain of trust intact.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client, and "S:" represents lines returned by a protocol server. Indentation and white space in examples is provided only to illustrate element relationships and is not a mandatory feature of this protocol.

1.2. Secure Transfer of DNSSEC Key Material

Exchanging DNSSEC key material in preparation of a domain name transfer is one of the phases in the lifecycle of a domain name [I-D.koch-dnsop-dnssec-operator-change].

DNS-operators need to exchange DNSSEC key material before the registration data can be changed to keep the DNSSEC chain of trust intact. This exchange is normally initiated through the gaining registrar.

The gaining and losing DNS operators could talk directly to each other (the ~ arrow in Figure 1) to exchange the DNSKEY, but often there is no trusted path between the two. As both can securely interact with the registry over the administrative channel through the registrar, the registry can act as a relay for the key material exchange.

The registry is merely used as a relay channel. Therefore it is up to the losing DNS-operator to complete the intended transaction. The registry SHOULD have certain policies in place that require the losing DNS operator to cooperate with this transaction, however this is beyond this document. This document focuses on the EPP protocol syntax.

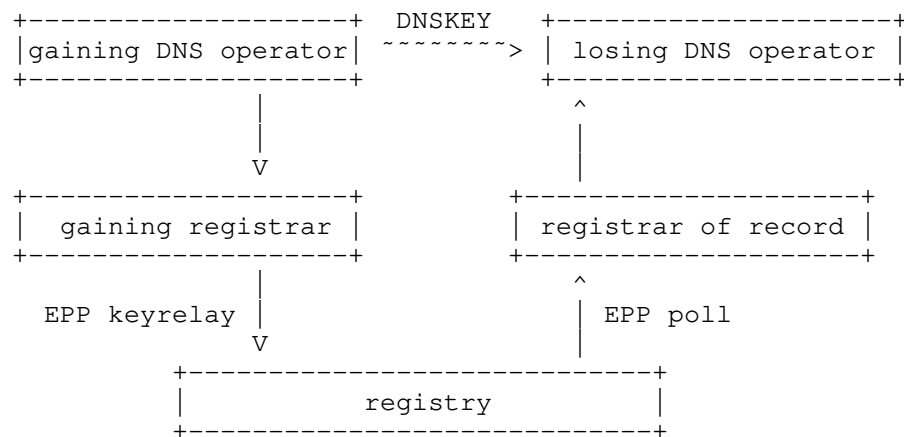


Figure 1: Transfer of DNSSEC key material.

There is no distinction in the EPP protocol between Registrars and DNS-operators, there is only mention of an EPP client and EPP server. Therefore the term EPP client will be used for the interaction with the EPP server for relaying DNSSEC key material.

2. Object Attributes

2.1. DNSSEC Key Material

The DNSSEC key material is represented in EPP by a <keyRelayData> element.

2.1.1. <keyRelayData> element

The <keyRelayData> contains the following elements:

- o One REQUIRED <keyData> element that contains the DNSSEC key material as described in [RFC5910], Section 4
- o An OPTIONAL <expiry> element that describes the expected lifetime of the relayed key(s) in the zone. When the <expiry> element is provided the losing DNS operator SHOULD remove the inserted key material from the zone after the expire time. This may be because the transaction that needed the insertion should either be completed or abandoned by that time. If a client receives a key relay object that has been sent previously it MUST update the expire time of the key material. This enables the clients to update the lifetime of the key material when a transfer is delayed.

The <expiry> element MUST contain exactly one of the following child elements:

* <absolute>: The DNSSEC key material is valid from the current date and time until it expires on the specified date and time. If a date in the past is provided this MUST be interpreted as a revocation of a previously sent key relay object.

* <relative>: The DNSSEC key material is valid from the current date and time until the end of the specified duration. If a period of zero is provided this MUST be interpreted as a revocation of a previously sent key relay object.

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mapping described here is specifically for use in this key relay mapping.

3.1. EPP Query Commands

EPP provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve

detailed information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

Check semantics do not apply to key relay objects, so there is no mapping defined for the EPP <check> command and the EPP <check> response.

3.1.2. EPP <info> Command

Info command semantics do not apply to the key relay objects, so there is no mapping defined for the EPP <info> Command.

The EPP <info> response for key relay objects is used in the EPP poll response, as described in [RFC5730]. The key relay object created with the <create> command, described in Section 3.2.1 is inserted into the receiving client's poll queue. The receiving client will receive the key relay object using the EPP <poll> command, as described in [RFC5730].

When a <poll> command has been processed successfully for a key relay poll message, the EPP <resData> element MUST contain a child <keyrelay:infData> element that is identified by the keyrelay namespace. The <keyrelay:infData> element contains the following child elements:

- o A REQUIRED <name> element containing the domain name for which the DNSSEC key material is relayed.
- o A REQUIRED <authInfo> element that contains authorization information associated with the domain object ([RFC5731], Section 3.2.1).
- o One or more REQUIRED <keyRelayData> elements containing data to be relayed, as defined in Section 2.1. A server MAY apply a server policy that specifies the number of <keyRelayData> elements that can be incorporated. When a server policy is violated, a server MUST respond with an EPP result code 2308 "Data management policy violation".
- o An OPTIONAL <crDate> element that contains the date and time of the submitted <create> command.
- o An OPTIONAL <reID> element that contains the identifier of the client that requested the key relay.

- o An OPTIONAL <acID> element that contains the identifier of the client that SHOULD act upon the key relay.

Example <poll> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:  xmlns:keyrelay="urn:ietf:params:xml:ns:keyrelay-1.0"
S:  xmlns:s="urn:ietf:params:xml:ns:secDNS-1.1"
S:  xmlns:d="urn:ietf:params:xml:ns:domain-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>1999-04-04T22:01:00.0Z</qDate>
S:      <msg>Keyrelay action completed successfully.</msg>
S:    </msgQ>
S:    <resData>
S:      <keyrelay:infData>
S:        <keyrelay:name>example.org</keyrelay:name>
S:        <keyrelay:authInfo>
S:          <d:pw>JnSdBAZSxxzJ</d:pw>
S:        </keyrelay:authInfo>
S:        <keyrelay:keyRelayData>
S:          <keyrelay:keyData>
S:            <s:flags>256</s:flags>
S:            <s:protocol>3</s:protocol>
S:            <s:alg>8</s:alg>
S:            <s:pubKey>cmlraXN0aGVhZGZlZXN0</s:pubKey>
S:          </keyrelay:keyData>
S:          <keyrelay:expiry>
S:            <keyrelay:relative>P1M13D</keyrelay:relative>
S:          </keyrelay:expiry>
S:        </keyrelay:keyRelayData>
S:        <keyrelay:crDate>
S:          1999-04-04T22:01:00.0Z
S:        </keyrelay:crDate>
S:        <keyrelay:reID>
S:          ClientX
S:        </keyrelay:reID>
S:        <keyrelay:acID>
S:          ClientY
S:        </keyrelay:acID>
S:      </keyrelay:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```


3.1.3. EPP <transfer> Command

Transfer semantics do not apply to key relay objects, so there is no mapping defined for the EPP <transfer> command.

3.2. EPP Transform Commands

EPP provides five commands to transform objects: <create> to create an instance of an object, <delete> to delete an instance of an object, <renew> to extend the validity period of an object, <transfer> to manage object sponsorship changes, and <update> to change information associated with an object.

3.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create a key relay object that includes the domain name and DNSSEC key material to be relayed. When the <create> command is validated, the server MUST insert an EPP <poll> message, using the key relay info response (See Section 3.1.2), in the receiving client's poll queue that belongs to the registrar on record of the provided domain name.

In addition to the standard EPP command elements, the <create> command MUST contain a <keyrelay:create> element that is identified by the keyrelay namespace. The <keyrelay:create> element contains the following child elements:

- o A REQUIRED <keyrelay:name> element containing the domain name for which the DNSSEC key material is relayed.
- o A REQUIRED <authInfo> element that contains authorization information associated with the domain object ([RFC5731], Section 3.2.1).
- o One or more REQUIRED <keyrelay:keyRelayData> element containing data to be relayed, as defined in Section 2.1

Example <create> commands:

Note that in the provided example the second <keyrelay:keyRelayData> element has a period of zero and thus represents the revocation of a previously sent key relay object (see Section 2.1.1).

```

C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
C:  xmlns:keyrelay="urn:ietf:params:xml:ns:keyrelay-1.0"
C:  xmlns:s="urn:ietf:params:xml:ns:secDNS-1.1"
C:  xmlns:d="urn:ietf:params:xml:ns:domain-1.0">
C:  <command>
C:    <create>
C:      <keyrelay:create>
C:        <keyrelay:name>example.org</keyrelay:name>
C:        <keyrelay:authInfo>
C:          <d:pw>JnSdBAZSxxzJ</d:pw>
C:        </keyrelay:authInfo>
C:        <keyrelay:keyRelayData>
C:          <keyrelay:keyData>
C:            <s:flags>256</s:flags>
C:            <s:protocol>3</s:protocol>
C:            <s:alg>8</s:alg>
C:            <s:pubKey>cmlraXN0aGVlZXN0</s:pubKey>
C:          </keyrelay:keyData>
C:          <keyrelay:expiry>
C:            <keyrelay:relative>P1M13D</keyrelay:relative>
C:          </keyrelay:expiry>
C:        </keyrelay:keyRelayData>
C:        <keyrelay:keyRelayData>
C:          <keyrelay:keyData>
C:            <s:flags>256</s:flags>
C:            <s:protocol>3</s:protocol>
C:            <s:alg>8</s:alg>
C:            <s:pubKey>bWFyY2lzdGhlYmVzdA==</s:pubKey>
C:          </keyrelay:keyData>
C:          <keyrelay:expiry>
C:            <keyrelay:relative>P0D</keyrelay:relative>
C:          </keyrelay:expiry>
C:        </keyrelay:keyRelayData>
C:      </keyrelay:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>

```

When a server has successfully processed the <create> command it MUST respond with a standard EPP response. See [RFC5730], Section 2.6.

Example <create> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

When a server cannot process the <create> command due to the server policy it MUST return an EPP 2308 error message. This might be the case when the server knows that the receiving client does not support keyrelay transactions. See [RFC5730], Section 2.6.

Example <create> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="2308">
S:      <msg>Data management policy violation</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.2.2. EPP <delete> Command

Delete semantics do not apply to key relay objects, so there is no mapping defined for the EPP <delete> command and the EPP <delete> response.

3.2.3. EPP <renew> Command

Renew semantics do not apply to key relay objects, so there is no mapping defined for the EPP <renew> command and the EPP <renew> response.

3.2.4. EPP <transfer> Command

Transfer semantics do not apply to key relay objects, so there is no mapping defined for the EPP <transfer> command and the EPP <transfer> response.

3.2.5. EPP <update> Command

Update semantics do not apply to key relay objects, so there is no mapping defined for the EPP <update> command and the EPP <update> response.

4. Formal Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:keyrelay-1.0"
  xmlns:keyrelay="urn:ietf:params:xml:ns:keyrelay-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1"
  xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0 protocol
      extension schema for relaying DNSSEC key material.
    </documentation>
  </annotation>

  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0" />
  <import namespace="urn:ietf:params:xml:ns:secDNS-1.1" />
  <import namespace="urn:ietf:params:xml:ns:domain-1.0" />

  <element name="keyRelayData" type="keyrelay:keyRelayDataType" />
  <element name="infData" type="keyrelay:infDataType" />
  <element name="create" type="keyrelay:createType" />

  <complexType name="createType">
    <sequence>
      <element name="name" type="eppcom:labelType" />
      <element name="authInfo" type="domain:authInfoType" />
      <element name="keyRelayData" type="keyrelay:keyRelayDataType"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="infDataType">
```

```
<sequence>
  <element name="name" type="eppcom:labelType" />
  <element name="authInfo" type="domain:authInfoType" />
  <element name="keyRelayData" type="keyrelay:keyRelayDataType"
    maxOccurs="unbounded"/>
  <element name="crDate" type="dateTime"/>
  <element name="reID" type="eppcom:clIDType" />
  <element name="acID" type="eppcom:clIDType" />
</sequence>
</complexType>

<complexType name="keyRelayDataType">
  <sequence>
    <element name="keyData" type="secDNS:keyDataType" />
    <element name="expiry" type="keyrelay:keyRelayExpiryType"
      minOccurs="0" />
  </sequence>
</complexType>

<complexType name="keyRelayExpiryType">
  <choice>
    <element name="absolute" type="dateTime" />
    <element name="relative" type="duration" />
  </choice>
</complexType>
</schema>
```

5. IANA Considerations

5.1. XML Namespace

This document uses URNs to describe a XML namespace conforming to the registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

URI: urn:ietf:params:xml:ns:keyrelay-1.0

Registrant Contact: See the "Author's Address" section of this document.

XML: See the "Formal Syntax" section of this document.

5.2. XML Schema

This document uses URNs to describe a XML schema conforming to the registry mechanism described in [RFC3688]. The following URI assignment is requested of IANA:

URI: urn:ietf:params:xml:schema:keyrelay-1.0

XML: See the "Formal Syntax" section of this document.

5.3. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Key Relay Mapping for the Extensible Provisioning Protocol"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, iesg@ietf.org

TLDs: Any

IPR Disclosure: <https://datatracker.ietf.org/ipr/2393/>

Status: Active

Notes: None

6. Security Considerations

A server SHOULD NOT perform any transformation on data under server management when processing a <keyrelay:create> command. The intent of this command is to put DNSSEC key material on the poll queue of another client. To make sure that this EPP extension is interoperable with the different server policies that already have implemented EPP this extension it is not classified as must not.

Any EPP client can use this mechanism to put data on the message queue of another EPP client, allowing for the potential of a denial of service attack. However this can, and should be detected by the server. A server MAY set a server policy which limits or rejects a <keyrelay:create> command if it detects the mechanism is being abused.

For the <keyrelay:keyRelayData> data a correct <domain:authInfo> element should be used as an indication that putting the key material on the receiving EPP clients poll queue is authorized by the _registrant_ of that domain name. The authorization of EPP clients

to perform DNS changes is not covered in this document as it depends on registry specific policy.

A client that uses this mechanism to send DNSSEC key material to another client could verify through DNS that the DNSSEC key material is added to the authoritative zone of the domain. This check can be used to verify that the DNSSEC key material has traveled end-to-end from the gaining DNS operator to the losing DNS operator. This check does not tell anything about the DNSSEC chain of trust and can merely be used as a verification of a succesful transfer of the DNSSEC key material.

7. Acknowledgements

We like to thank the following individuals for their valuable input, review, constructive criticism in earlier revisions or support for the concepts described in this document:

Maarten Wullink, Marco Davids, Ed Lewis, James Mitchell, David Peal, Patrik Faltstrom, Klaus Malorny, James Gould, Patrick Mevzek, Seth Goldman, Maarten Bosteels, Ulrich Wisser, Kees Monshouwer and Scott Hollenbeck.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.
- [RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, May 2010.

8.2. Informative References

[I-D.koch-dnsop-dnssec-operator-change]

Koch, P., Sanz, M., and A. Verschuren, "Changing DNS Operators for DNSSEC signed Zones", draft-koch-dnsop-dnssec-operator-change-06 (work in progress), February 2014.

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, February 2015.

Appendix A. Changelog

[This section should be removed by the RFC editor before publishing]

A.1. draft-gieben-epp-keyrelay-00

1. Initial document.

A.2. draft-gieben-epp-keyrelay-01

1. Style and grammar changes;
2. Added an expire element as per suggestion by Klaus Malorny;
3. Make the authInfo element mandatory and make the registry check it as per feedback by Klaus Malorny and James Gould.

A.3. draft-gieben-epp-keyrelay-02

1. Added element to identify the relaying EPP client as suggested by Klaus Malorny;
2. Corrected XML for missing and excess clTRID as noted by Patrick Mevzek;
3. Added clarifications for the examples based on feedback by Patrick Mevzeck;
4. Reviewed the consistency of using DNS operator versus registrar after review comments by Patrick Faltstrom and Ed Lewis.

A.4. draft-gieben-epp-keyrelay-03

1. Style and grammar changes
2. Corrected acknowledgement section
3. Corrected XML for Expire element to not be mandatory but only occur once.

A.5. draft-ietf-eppeext-keyrelay-00

1. Added feedback from Seth Goldman and put him in the acknowledgement section.
2. IDnits formatting adjustments

A.6. draft-ietf-eppeext-keyrelay-01

1. Introducing the <relay> command, and thus separating the data and the command.
2. Updated the Introduction, describing the general use of relay vs the intended use-case of relaying DNSSEC key data.
3. Restructuring the document to make it more inline with existing EPP extensions.

A.7. draft-ietf-eppeext-keyrelay-02

1. Updated the XML structure by removing the <relay> command based on WG feedback
2. Updated the wording

A.8. draft-ietf-eppeext-keyrelay-03

1. Updated the document title in the EPP Extension Registry section
2. Restored Acknowledgement section, thanks to Marco Davids
3. Incorporated feedback from Patrick Mevzek

A.9. draft-ietf-eppeext-keyrelay-04

1. Incorporated feedback from James Gould
2. Added additional text when server is aware that receiving clients do not support keyrelay transactions or DNSSEC as suggested by Kees Monshouwer.
3. Added additional text for supporting key revocation as suggested by Kees Monshouwer
4. Updated some of the wording
5. Fix the usage of multiple keys in a create message

- A.10. draft-ietf-eppext-keyrelay-05
1. Review comments after WG last call
- A.11. draft-ietf-eppext-keyrelay-06
1. Review comments by Ulrich Wisser during IESG writeup
- A.12. draft-ietf-eppext-keyrelay-07
1. fixed changelog
- A.13. draft-ietf-eppext-keyrelay-08
1. fixed issue with authinfo
 2. fixed issue with relative period in example xml
- A.14. draft-ietf-eppext-keyrelay-09
1. fixed issue with naming
- A.15. draft-ietf-eppext-keyrelay-10
1. removed 4 spaces
- A.16. draft-ietf-eppext-keyrelay-11
1. Processed editorial changes from AD review
 2. Processed comments made during IETF last call
- A.17. draft-ietf-regext-keyrelay-00
1. Processed comments made during IESG review

Authors' Addresses

Rik Ribbers
SIDN
Meander 501
Arnhem 6825 MD
NL

Email: rik.ribbers@sidn.nl
URI: <https://www.sidn.nl/>

Marc Groeneweg
SIDN
Meander 501
Arnhem 6825 MD
NL

Email: marc.groeneweg@sidn.nl
URI: <https://www.sidn.nl/>

Miek Gieben

Email: miek@miek.nl

Antoin Verschuren

Email: ietf@antoin.nl

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2016

J. Gould
VeriSign, Inc.
W. Tan
Cloud Registry
G. Brown
CentralNic Ltd
November 2, 2015

Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)
draft-ietf-epext-launchphase-07

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension mapping for the provisioning and management of domain name registrations and applications during the launch of a domain name registry.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 5, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions Used in This Document	4
2. Object Attributes	5
2.1. Application Identifier	5
2.2. Validator Identifier	5
2.3. Launch Phases	6
2.4. Status Values	6
2.4.1. State Transition	8
2.5. Poll Messaging	9
2.6. Mark Validation Models	12
2.6.1. <launch:codeMark> element	13
2.6.2. <mark:mark> element	14
2.6.3. Digital Signature	14
2.6.3.1. <smd:signedMark> element	14
2.6.3.2. <smd:encodedSignedMark> element	14
3. EPP Command Mapping	14
3.1. EPP <check> Command	15
3.1.1. Claims Check Form	15
3.1.2. Availability Check Form	18
3.1.3. Trademark Check Form	20
3.2. EPP <info> Command	23
3.3. EPP <create> Command	26
3.3.1. Sunrise Create Form	26
3.3.2. Claims Create Form	32
3.3.3. General Create Form	35
3.3.4. Mixed Create Form	36
3.3.5. Create Response	38
3.4. EPP <update> Command	39
3.5. EPP <delete> Command	40
3.6. EPP <renew> Command	41
3.7. EPP <transfer> Command	42
4. Formal Syntax	42
4.1. Launch Schema	42
5. IANA Considerations	49
5.1. XML Namespace	49
5.2. EPP Extension Registry	50
6. Implementation Status	50
6.1. Verisign EPP SDK	51
6.2. Verisign Consolidated Top Level Domain (CTLD) SRS	51
6.3. Verisign .COM / .NET SRS	52
6.4. REngin v3.7	52
6.5. RegistryEngine EPP Service	52
6.6. Neustar EPP SDK	53

6.7. gTLD Shared Registry System	53
7. Security Considerations	54
8. Acknowledgements	54
9. Normative References	54
Appendix A. Change History	55
A.1. Change from 00 to 01	55
A.2. Change from 01 to 02	55
A.3. Change from 02 to 03	56
A.4. Change from 03 to 04	56
A.5. Change from 04 to 05	56
A.6. Change from 05 to 06	57
A.7. Change from 06 to 07	57
A.8. Change from 07 to 08	57
A.9. Change from 08 to 09	57
A.10. Change from 09 to 10	58
A.11. Change from 10 to 11	59
A.12. Change from 11 to 12	59
A.13. Change from 12 to WG 00	59
A.14. Change WG 00 to WG 01	59
A.15. Change WG 01 to WG 02	59
A.16. Change WG 02 to WG 03	60
A.17. Change WG 03 to WG 04	60
A.18. Change WG 04 to WG 05	60
A.19. Change WG 05 to WG 06	60
A.20. Change WG 06 to WG 07	60
Authors' Addresses	61

1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping specifies a flexible schema that can be used to implement several common use cases related to the provisioning and management of domain name registrations and applications during the launch of a domain name registry.

It is typical for domain registries to operate in special modes during their initial launch to facilitate allocation of domain names, often according to special rules. This document uses the term "launch phase" and the shorter form "launch" to refer to such a period.

The EPP domain name mapping [RFC5731] is designed for the steady-state operation of a registry. During a launch period, the model in place may be different from what is defined in the EPP domain name mapping [RFC5731]. For example, registries often accept multiple applications for the same domain name during the "Sunrise" launch phase, referred to as a Launch Application. A Launch Registration

refers to a registration made during a launch phase when the server uses a "first-come, first-served" model. Even in a "first-come, first-served" model, additional steps and information might be required, such as trademark information. In addition, the [I-D.ietf-eppext-tmch-smd] defines a registry interface for the Trademark Claims or "claims" launch phase that includes support for presenting a Trademark Claims Notice to the Registrant. This document proposes an extension to the domain name mapping in order to provide a uniform interface for the management of Launch Applications and Launch Registrations in launch phases.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this protocol.

"launch-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:launch-1.0". The XML namespace prefix "launch" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

"signedMark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:signedMark-1.0" that is defined in [I-D.ietf-eppext-tmch-smd]. The XML namespace prefix "smd" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

"mark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:mark-1.0" that is defined in [I-D.ietf-eppext-tmch-smd]. The XML namespace prefix "mark" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Object Attributes

This extension adds additional elements to the EPP domain name mapping [RFC5731]. Only those new elements are described here.

2.1. Application Identifier

Servers MAY allow multiple applications, referred to as a Launch Application, of the same domain name during its launch phase operations. Upon receiving a valid request to create a Launch Application, the server MUST create an application object corresponding to the request, assign an application identifier for the Launch Application, set the [RFC5731] pendingCreate status, and return the application identifier to the client with the <launch:applicationID> element. In order to facilitate correlation, all subsequent launch operations on the Launch Application MUST be qualified by the previously assigned application identifier using the <launch:applicationID> element.

If the <domain:create> command processes a request synchronously without the use of an intermediate Launch Application, then an application identifier MAY not be needed.

2.2. Validator Identifier

The Validator Identifier is the unique identifier for a Trademark Validator that validates marks and has a repository of validated marks. The OPTIONAL "validatorID" attribute is used to define the Validator Identifier of the Trademark Validator. Registries MAY support more than one Third Party Trademark Validator. The Internet Corporation for Assigned Names and Numbers (ICANN) Trademark Clearinghouse (TMCH) is the default Trademark Validator and is reserved the Validator Identifier of "tmch". If the ICANN TMCH is not used or multiple Trademark Validators are used, the Validator Identifier MUST be defined using the "validatorID" attribute.

The Validator Identifier MAY be related to one or more issuer identifiers of the <mark:id> element and the <smd:id> element defined in [I-D.ietf-eppext-tmch-smd]. Both the Validator Identifier and the Issuer Identifier used MUST be unique. The list of validator identifiers and the relationship to issuer identifiers is out of scope for this document.

The Validator Identifier MAY define a non-Trademark Validator that supports a form of claims.

2.3. Launch Phases

The server MAY support multiple launch phases sequentially or simultaneously. The <launch:phase> element MUST be included by the client to define the target launch phase of the command. The server SHOULD validate the phase and MAY validate the sub-phase of the <launch:phase> element against the active phase and OPTIONAL sub-phase of the server on a create command, and return an EPP error result code of 2306 if there is a mismatch.

The following launch phase values are defined:

- sunrise The phase during which trademark holders can submit registrations or applications with trademark information that can be validated by the server.
- landrush A post-Sunrise phase when non-trademark holders are allowed to register domain names with steps taken to address a large volume of initial registrations.
- claims The Trademark Claims phase, as defined in the TMCH Functional Specification [I-D.ietf-eppext-tmch-func-spec], in which a Claims Notice must be displayed to a prospective registrant of a domain name that matches trademarks.
- open A post-launch phase that is also referred to as "steady state". Servers MAY require additional trademark protection during this phase.
- custom A custom server launch phase that is defined using the "name" attribute.

For extensibility, the <launch:phase> element includes an OPTIONAL "name" attribute that can define a sub-phase, or the full name of the phase when the <launch:phase> element has the "custom" value. For example, the "claims" launch phase could have two sub-phases that include "landrush" and "open".

Launch phases MAY overlap to support the "claims" launch phase, defined in the TMCH Functional Specification [I-D.ietf-eppext-tmch-func-spec], and to support a traditional "landrush" launch phase. The overlap of the "claims" and "landrush" launch phases SHOULD be handled by setting "claims" as the <launch:phase> value and setting "landrush" as the sub-phase with the "name" attribute. For example, the <launch:phase> element SHOULD be <launch:phase name="landrush">claims</launch:phase>.

2.4. Status Values

A Launch Application or Launch Registration object MAY have a launch status value. The <launch:status> element is used to convey the launch status pertaining to the object, beyond what is specified in

the object mapping. A Launch Application or Launch Registration MUST set the [RFC5731] "pendingCreate" status if a launch status is supported and the launch status is not one of the final statuses, including the "allocated" and "rejected" statuses.

The following status values are defined using the required "s" attribute:

pendingValidation: The initial state of a newly-created application or registration object. The application or registration requires validation, but the validation process has not yet completed.

validated: The application or registration meets relevant registry rules.

invalid: The application or registration does not validate according to registry rules. Server policies permitting, it may transition back into "pendingValidation" for revalidation, after modifications are made to ostensibly correct attributes that caused the validation failure.

pendingAllocation: The allocation of the application or registration is pending based on the results of some out-of-band process (for example, an auction).

allocated: The object corresponding to the application or registration has been provisioned. Is a possible end state of an application or registration object.

rejected: The application or registration object was not provisioned. Is a possible end state of an application or registration object.

custom: A custom status that is defined using the "name" attribute.

Each status value MAY be accompanied by a string of human-readable text that describes the rationale for the status applied to the object. The OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than the default value of "en" (English).

For extensibility the <launch:status> element includes an OPTIONAL "name" attribute that can define a sub-status or the full name of the status when the status value is "custom". The server SHOULD NOT use the "custom" status value.

Status values MAY be skipped. For example, an application or registration MAY immediately start at the "allocated" status or an application or registration MAY skip the "pendingAllocation" status. If the launch phase does not require validation of a request, an application or registration MAY immediately skip to "pendingAllocation".

2.4.1. State Transition

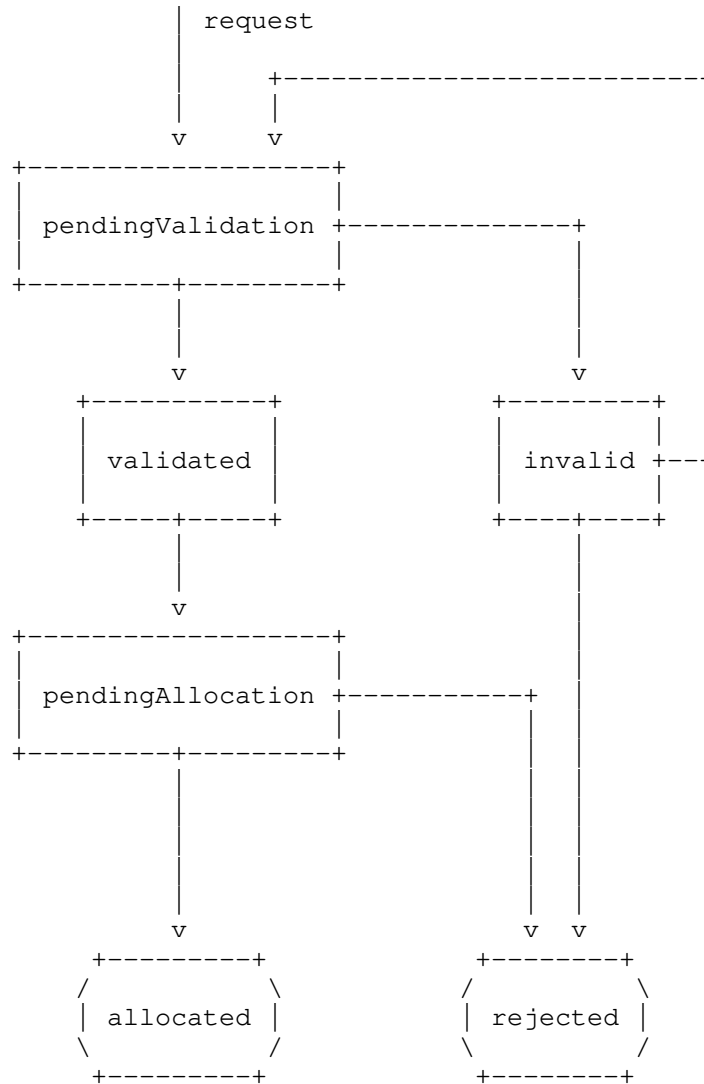


Figure 1

2.5. Poll Messaging

A Launch Application MUST and a Launch Registration MAY be handled as a domain name of [RFC5731] in "pendingCreate" status, with the launch status values defined in Section 2.4. As a Launch Application or Launch Registration transitions between the status values defined in Section 2.4, the server SHOULD insert poll messages, per [RFC5730], for the applicable intermediate statuses, including the "pendingValidation", "validated", "pendingAllocation, and "invalid" statuses, using the <domain:infData> element with the <launch:infData> extension. The <domain:infData> element MAY contain non-mandatory information, like contact and name server information. Also, further extensions that would normally be included in the response of a <domain:info> command, per [RFC5731], MAY be included. For the final statuses, including the "allocated" and "rejected" statuses, the server MUST insert a <domain:panData> poll message, per [RFC5731], with the <launch:infData> extension.

The following is an example poll message for a Launch Application that has transitioned to the "pendingAllocation" state.

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The following is an example <domain:panData> poll message for an "allocated" Launch Application.

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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 successfully allocated.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:panData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name paResult="1">domain.example</domain:name>
S:        <domain:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </domain:paTRID>
S:        <domain:paDate>2013-04-04T22:00:00.0Z</domain:paDate>
S:      </domain:panData>
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="allocated"/>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The following is an example <domain:panData> poll message for an "allocated" Launch Registration.

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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>Registration successfully allocated.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:panData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name paResult="1">domain.example</domain:name>
S:        <domain:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </domain:paTRID>
S:        <domain:paDate>2013-04-04T22:00:00.0Z</domain:paDate>
S:      </domain:panData>
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:status s="allocated"/>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

2.6. Mark Validation Models

A server MUST support at least one of the following models for validating trademark information:

code Use of a mark code by itself to validate that the mark matches the domain name. This model is supported using the <launch:codeMark> element with just the <launch:code> element.

mark The mark information is passed without any other validation element. The server will use some custom form of validation to

validate that the mark information is authentic. This model is supported using the <launch:codeMark> element with just the <mark:mark> (Section 2.6.2) element.

code with mark: A code is used along with the mark information by the server to validate the mark utilizing an external party. The code represents some form of secret that matches the mark information passed. This model is supported using the <launch:codeMark> element that contains both the <launch:code> and the <mark:mark> (Section 2.6.2) elements.

signed mark: The mark information is digitally signed as described in the Digital Signature (Section 2.6.3) section. The digital signature can be directly validated by the server using the public key of the external party that created the signed mark using its private key. This model is supported using the <smd:signedMark> (Section 2.6.3.1) and <smd:encodedSignedMark> (Section 2.6.3.2) elements.

More than one <launch:codeMark>, <smd:signedMark> (Section 2.6.3.1), or <smd:encodedSignedMark> (Section 2.6.3.2) element MAY be specified. The maximum number of marks per domain name is up to server policy.

2.6.1. <launch:codeMark> element

The <launch:codeMark> element that is used by the "code", "mark", and "code with mark" validation models, has the following child elements:

<launch:code>: OPTIONAL mark code used to validate the <mark:mark> (Section 2.6.2) information. The mark code is be a mark-specific secret that the server can verify against a third party. The OPTIONAL "validatorID" attribute is the Validator Identifier (Section 2.2) whose value indicates which Trademark Validator that the code originated from, with no default value.

<mark:mark>: OPTIONAL mark information with child elements defined in the Mark (Section 2.6.2) section.

The following is an example <launch:codeMark> element with both a <launch:code> and <mark:mark> (Section 2.6.2) element.

```
<launch:codeMark>
  <launch:code validatorID="sample">
    49FD46E6C4B45C55D4AC</launch:code>
    <mark:mark xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
      ...
    </mark:mark>
  </launch:codeMark>
```


2.6.2. <mark:mark> element

A <mark:mark> element describes an applicant's prior right to a given domain name that is used with the "mark", "mark with code", and the "signed mark" validation models. The <mark:mark> element is defined in [I-D.ietf-eppext-tmch-smd]. A new mark format can be supported by creating a new XML schema for the mark that has an element that substitutes for the <mark:abstractMark> element from [I-D.ietf-eppext-tmch-smd].

2.6.3. Digital Signature

Digital signatures MAY be used by the server to validate either the mark information, when using the "signed mark" validation model with the <smd:signedMark> (Section 2.6.3.1) element or the <smd:encodedSignedMark> (Section 2.6.3.2) element.

2.6.3.1. <smd:signedMark> element

The <smd:signedMark> element contains the digitally signed mark information. The <smd:signedMark> element is defined in [I-D.ietf-eppext-tmch-smd]. A new signed mark format can be supported by creating a new XML schema for the signed mark that has an element that substitutes for the <smd:abstractSignedMark> element from [I-D.ietf-eppext-tmch-smd].

2.6.3.2. <smd:encodedSignedMark> element

The <smd:encodedSignedMark> element contains an encoded form of the digitally signed <smd:signedMark> (Section 2.6.3.1) element. The <smd:encodedSignedMark> element is defined in [I-D.ietf-eppext-tmch-smd]. A new encoded signed mark format can be supported by creating a new XML schema for the encoded signed mark that has an element that substitutes for the <smd:encodedSignedMark> element from [I-D.ietf-eppext-tmch-smd].

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in the Launch Phase Extension.

This mapping is designed to be flexible, requiring only a minimum set of required elements.

While it is meant to serve several use cases, it does not prescribe any interpretation by the client or server. Such processing is

typically highly policy-dependent and therefore specific to implementations.

Operations on application objects are done via one or more of the existing EPP verbs defined in the EPP domain name mapping [RFC5731]. Registries MAY choose to support a subset of the operations.

3.1. EPP <check> Command

There are three forms of the extension to the EPP <check> command: the Claims Check Form (Section 3.1.1), the Availability Check Form (Section 3.1.2), and the Trademark Check Form (Section 3.1.3). The <launch:check> element "type" attribute defines the form, with the value of "claims" for the Claims Check Form (Section 3.1.1), with the value of "avail" for the Availability Check Form (Section 3.1.2), and with the value of "trademark" for the Trademark Check Form (Section 3.1.3). The default value of the "type" attribute is "claims". The forms supported by the server is determined by server policy. The server MUST return an EPP error result code of 2307 if it receives a check form that is not supported.

3.1.1. Claims Check Form

The Claims Check Form defines a new command called the Claims Check Command that is used to determine whether or not there are any matching trademarks, in the specified launch phase, for each domain name passed in the command, that requires the use of the "Claims Create Form" on a Domain Create Command. The availability check information defined in the EPP domain name mapping [RFC5731] MUST NOT be returned for the Claims Check Command. This form is the default form and MAY be explicitly identified by setting the <launch:check> "type" attribute to "claims".

Instead of returning whether the domain name is available, the Claims Check Command will return whether or not at least one matching trademark exists for the domain name, that requires the use of the "Claims Create Form" on a Domain Create Command. If there is at least one matching trademark that exists for the domain name, a <launch:claimKey> element is returned. The client MAY then use the value of the <launch:claimKey> element to obtain information needed to generate the Trademark Claims Notice from Trademark Validator based on the Validator Identifier (Section 2.2). The unique notice identifier of the Trademark Claims Notice MUST be passed in the <launch:noticeID> element of the extension to the Create Command (Section 3.3).

The <domain:name> elements in the EPP <check> command of EPP domain name mapping [RFC5731] define the domain names to check for matching

trademarks. The <launch:check> element contains the following child elements:

<launch:phase> Contains the value of the active launch phase of the server. The server SHOULD validate the value against the active server launch phase.

Example Claims Check command using the <check> domain command and the <launch:check> extension with the "type" explicitly set to "claims", to determine if "domain1.example", "domain2.example", and "domain3.example" require claims notices during the "claims" launch phase:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:name>domain2.example</domain:name>
C:          <domain:name>domain3.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="claims">
C:          <launch:phase>claims</launch:phase>
C:        </launch:check>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

If the <check> command has been processed successfully, the EPP <response> MUST contain an <extension> <launch:chkData> element that identifies the launch namespace. The <launch:chkData> element contains the following child elements:

<launch:phase> The phase that mirrors the <launch:phase> element included in the <launch:check>.

<launch:cd> One or more <launch:cd> elements that contain the following child elements:

<launch:name> Contains the fully qualified name of the queried domain name. This element MUST contain an "exists" attribute

whose value indicates if a matching trademark exists for the domain name that requires the use of the "Claims Create Form" on a Domain Create Command. A value of "1" (or "true") means that a matching trademark does exist and that the "Claims Create Form" is required on a Domain Create Command. A value of "0" (or "false") means that a matching trademark does not exist or that the "Claims Create Form" is NOT required on a Domain Create Command.

<launch:claimKey> Zero or more OPTIONAL claim keys that MAY be passed to a third-party trademark validator such as the Trademark Clearinghouse (TMCH) for querying the information needed to generate a Trademark Claims Notice. The <launch:claimKey> is used as the key for the query in place of the domain name to securely query the service without using a well-known value like a domain name. The OPTIONAL "validatorID" attribute is the Validator Identifier (Section 2.2) whose value indicates which Trademark Validator to query for the Claims Notice information, with the default being the ICANN TMCH. The "validatorID" attribute MAY reference a non-trademark claims clearinghouse identifier to support other forms of claims notices.

Example Claims Check response when a claims notice is not required for the domain name domain1.example, a claims notice is required for the domain name domain2.example in the "tmch", and a claims notice is required for the domain name domain3.example in the "tmch" and "custom-tmch", for the "claims" launch phase:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:phase>claims</launch:phase>
S:        <launch:cd>
S:          <launch:name exists="0">domain1.example</launch:name>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain2.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain3.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJlNr2vDsAzasdff7EasdfgjX4R0000000002
S:          </launch:claimKey>
S:        </launch:cd>
S:      </launch:chkData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.1.2. Availability Check Form

The Availability Check Form defines additional elements to extend the EPP <check> command described in the EPP domain name mapping [RFC5731]. No additional elements are defined for the EPP <check>

response. This form MUST be identified by setting the <launch:check> "type" attribute to "avail".

The EPP <check> command is used to determine if an object can be provisioned within a repository. Domain names may be made available only in unique launch phases, whilst remaining unavailable for concurrent launch phases. In addition to the elements expressed in the <domain:check>, the command is extended with the <launch:check> element that contains the following child elements:

<launch:phase> The launch phase to which domain name availability should be determined.

Example Availability Check Form command using the <check> domain command and the <launch:check> extension with the "type" set to "avail", to determine the availability of two domain names in the "idn-release" custom launch phase:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:name>domain2.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="avail">
C:        <launch:phase name="idn-release">custom</launch:phase>
C:      </launch:check>
C:    </extension>
C:  <clTRID>ABC-12345</clTRID>
C: </command>
C:</epp>
```

The Availability Check Form does not define any extension to the response of an <check> domain command. After processing the command, the server replies with a standard EPP response as defined in the EPP domain name mapping [RFC5731].

3.1.3. Trademark Check Form

The Trademark Check Form defines a new command called the Trademark Check Command that is used to determine whether or not there are any matching trademarks for each domain name passed in the command, independent of the active launch phase of the server and whether the "Claims Create Form" is required on a Domain Create Command. The availability check information defined in the EPP domain name mapping [RFC5731] MUST NOT be returned for the Claims Check Command. This form MUST be identified by setting the <launch:check> "type" attribute to "trademark".

Instead of returning whether the domain name is available, the Trademark Check Command will return whether or not at least one matching trademark exists for the domain name. If there is at least one matching trademark that exists for the domain name, a <launch:claimKey> element is returned. The client MAY then use the value of the <launch:claimKey> element to obtain Trademark Claims Notice information from Trademark Validator based on the Validator Identifier (Section 2.2).

The <domain:name> elements in the EPP <check> command of EPP domain name mapping [RFC5731] define the domain names to check for matching trademarks. The <launch:check> element does not contain any child elements with the "Trademark Check Form":

Example Trademark Check command using the <check> domain command and the <launch:check> extension with the "type" set to "trademark", to determine if "domain1.example", "domain2.example", and "domain3.example" have any matching trademarks:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:name>domain2.example</domain:name>
C:          <domain:name>domain3.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:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

If the <check> command has been processed successfully, the EPP <response> MUST contain an <extension> <launch:chkData> element that identifies the launch namespace. The <launch:chkData> element contains the following child elements:

<launch:cd> One or more <launch:cd> elements that contain the following child elements:

- <launch:name> Contains the fully qualified name of the queried domain name. This element MUST contain an "exists" attribute whose value indicates if a matching trademark exists for the domain name. A value of "1" (or "true") means that a matching trademark does exist. A value of "0" (or "false") means that a matching trademark does not exist.
- <launch:claimKey> Zero or more OPTIONAL claim keys that MAY be passed to a third-party trademark validator such as the Trademark Clearinghouse (TMCH) for querying the information needed to generate a Trademark Claims Notice. The <launch:claimKey> is used as the key for the query in place of the domain name to securely query the service without using a well-known value like a domain name. The OPTIONAL "validatorID" attribute is the Validator Identifier

(Section 2.2) whose value indicates which Trademark Validator to query for the Claims Notice information, with the default being the ICANN TMCH. The "validatorID" attribute MAY reference a non-trademark claims clearinghouse identifier to support other forms of claims notices.

Example Trademark Check response when no matching trademarks are found for the domain name domain1.example, matching trademarks are found for the domain name domain2.example in the "tmch", matching trademarks are found for domain name domain3.example in the "tmch" and "custom-tmch", for the "claims" launch phase:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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="0">domain1.example</launch:name>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain2.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain3.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJlNrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJlNr2vDsAzasdff7EasdfgjX4R0000000002
S:          </launch:claimKey>
S:        </launch:cd>
S:      </launch:chkData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.2. EPP <info> Command

This extension defines additional elements to extend the EPP <info> command and response to be used in conjunction with the EPP domain name mapping [RFC5731].

The EPP <info> command is used to retrieve information for a launch phase registration or application. The Application Identifier (Section 2.1) returned in the <launch:creData> element of the create response (Section 3.3) is used for retrieving information for a Launch Application. A <launch:info> element is sent along with the regular <info> domain command. The <launch:info> element includes an OPTIONAL "includeMark" boolean attribute, with a default value of "false", to indicate whether or not to include the mark in the response. The <launch:info> element contains the following child elements:

<launch:phase> The phase during which the application or registration was submitted or is associated with. Server policy defines the phases that are supported.
<launch:applicationID> OPTIONAL application identifier of the Launch Application.

Example <info> domain command with the <launch:info> extension to retrieve information for the sunrise application for domain.example and application identifier "abc123":

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:applicationID>abc123</launch:applicationID>
C:            </launch:info>
C:          </extension>
C:        <clTRID>ABC-12345</clTRID>
C:      </command>
C:</epp>
```

Example <info> domain command with the <launch:info> extension to retrieve information for the sunrise registration for domain.example:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:          <launch:phase>sunrise</launch:phase>
C:        </launch:info>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

If the query was successful, the server replies with a <launch:infData> element along with the regular EPP <resData>. The <launch:infData> contains the following child elements:

<launch:phase> The phase during which the application was submitted, or is associated with, that matches the associated <info> command <launch:phase>.

<launch:applicationID> OPTIONAL Application Identifier of the Launch Application.

<launch:status> OPTIONAL status of the Launch Application using one of the supported status values (Section 2.4).

<mark:mark> Zero or more <mark:mark> (Section 2.6.2) elements.

Example <info> domain response using the <launch:infData> extension with the mark information:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="pendingCreate"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
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:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.3. EPP <create> Command

There are four forms of the extension to the EPP <create> command that include the Sunrise Create Form (Section 3.3.1), the Claims Create Form (Section 3.3.2), the General Create Form (Section 3.3.3), and the Mixed Create Form (Section 3.3.4). The form is dependent on the supported launch phases (Section 2.3) as defined below.

sunrise The EPP <create> command with the "sunrise" launch phase is used to submit a registration with trademark information that can be verified by the server with the <domain:name> value. The Sunrise Create Form (Section 3.3.1) is used for the "sunrise" launch phase.

landrush The EPP <create> command with the "landrush" launch phase MAY use the General Create Form (Section 3.3.3) to explicitly specify the phase and optionally define the expected type of object to create.

claims The EPP <create> command with the "claims" launch phase is used to pass the information associated with the presentation and acceptance of the Claims Notice. The Claims Create Form (Section 3.3.2) is used and the General Create Form (Section 3.3.3) MAY be used for the "claims" launch phase.

open The EPP <create> command with the "open" launch phase is undefined but the form supported is up to server policy. Use of the Claims Create Form (Section 3.3.2) MAY be used to pass the information associated with the presentation and acceptance of the Claims Notice if required for the domain name.

custom The EPP <create> command with the "custom" launch phase is undefined but the form supported is up to server policy.

3.3.1. Sunrise Create Form

The Sunrise Create Form of the extension to the EPP domain name mapping [RFC5731] includes the verifiable trademark information that the server uses to match against the domain name to authorize the domain create. A server MUST support one of four models in Claim Validation Models (Section 2.6) to verify the trademark information passed by the client.

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element has an OPTIONAL "type" attribute that defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created. The <launch:create> element contains the following child elements:

<launch:phase> The identifier for the launch phase.

<launch:codeMark> or <smd:signedMark> or <smd:encodedSignedMark>

<launch:codeMark> Zero or more <launch:codeMark> elements. The <launch:codeMark> child elements are defined in the <launch:codeMark> element (Section 2.6.1) section.

<smd:signedMark> Zero or more <smd:signedMark> elements. The <smd:signedMark> child elements are defined in the <smd:signedMark> element (Section 2.6.3.1) section.

<smd:encodedSignedMark> Zero or more <smd:encodedSignedMark> elements. The <smd:encodedSignedMark> child elements are defined in the <smd:encodedSignedMark> element (Section 2.6.3.2) section.

The following is an example <create> domain command using the <launch:create> extension, following the "code" validation model, with multiple sunrise codes:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
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:codeMark>
C:          <launch:code>49FD46E6C4B45C55D4AD</launch:code>
C:        </launch:codeMark>
C:        <launch:codeMark>
C:          <launch:code validatorID="sample2">
C:            49FD46E6C4B45C55D4AE</launch:code>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The following is an example <create> domain command using the <launch:create> extension, following the "mark" validation model, with the mark information:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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>domainone.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
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:          <mark:mark
C:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:              ...
C:          </mark:mark>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```


The following is an example <create> domain command using the <launch:create> extension, following the "code with mark" validation model, with a code and mark information:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
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="sample">
C:            49FD46E6C4B45C55D4AC</launch:code>
C:          <mark:mark
C:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:              ...
C:            </mark:mark>
C:          </launch:codeMark>
C:        </launch:create>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The following is an example <create> domain command using the <launch:create> extension, following the "signed mark" validation model, with the signed mark information for a sunrise application:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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>domainone.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        type="application">
C:        <launch:phase>sunrise</launch:phase>
C:        <smd:signedMark id="signedMark"
C:          xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:          ...
C:        </smd:signedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The following is an example <create> domain command using the <launch:create> extension, following the "signed mark" validation model, with the base64 encoded signed mark information:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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>domainone.example</domain:name>
C:          <domain:registrant>jd1234</domain:registrant>
C:          <domain:contact type="admin">sh8013</domain:contact>
C:          <domain:contact type="tech">sh8013</domain:contact>
C:          <domain:authInfo>
C:            <domain:pw>2fooBAR</domain:pw>
C:          </domain:authInfo>
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:          <smd:encodedSignedMark>
C:            xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:              ...
C:            </smd:encodedSignedMark>
C:          </launch:create>
C:        </extension>
C:      <clTRID>ABC-12345</clTRID>
C:    </command>
C:</epp>
```

3.3.2. Claims Create Form

The Claims Create Form of the extension to the EPP domain name mapping [RFC5731] includes the information related to the registrant's acceptance of the Claims Notice.

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element has an OPTIONAL "type" attribute that defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created. The <launch:create> element contains the following child elements:

<launch:phase> Contains the value of the active launch phase of the server. The server SHOULD validate the value against the active server launch phase.

<launch:notice> One or more <launch:notice> elements that contain the following child elements:

<launch:noticeID> Unique notice identifier for the Claims Notice. The <launch:noticeID> element has an OPTIONAL "validatorID" attribute is the Validator Identifier (Section 2.2) whose value indicates which Trademark Validator is the source of the claims notice, with the default being the ICANN TMCH.

<launch:notAfter> Expiry of the claims notice.

<launch:acceptedDate> Contains the date and time that the claims notice was accepted.

The following is an example <create> domain command using the <launch:create> extension with the <launch:notice> information for the "tmch" and the "custom-tmch" validators, for the "claims" launch phase:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:registrar>jdl234</domain:registrar>
C:          <domain:contact type="admin">sh8013</domain:contact>
C:          <domain:contact type="tech">sh8013</domain:contact>
C:          <domain:authInfo>
C:            <domain:pw>2fooBAR</domain:pw>
C:          </domain:authInfo>
C:        </domain:create>
C:      </create>
C:    <extension>
C:      <launch:create>
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:          <launch:phase>claims</launch:phase>
C:          <launch:notice>
C:            <launch:noticeID validatorID="tmch">
C:              370d0b7c9223372036854775807</launch:noticeID>
C:            <launch:notAfter>2014-06-19T10:00:00.0Z
C:            </launch:notAfter>
C:            <launch:acceptedDate>2014-06-19T09:00:00.0Z
C:            </launch:acceptedDate>
C:          </launch:notice>
C:          <launch:notice>
C:            <launch:noticeID validatorID="custom-tmch">
C:              470d0b7c9223654313275808</launch:noticeID>
C:            <launch:notAfter>2014-06-19T10:00:00.0Z
C:            </launch:notAfter>
C:            <launch:acceptedDate>2014-06-19T09:00:30.0Z
C:            </launch:acceptedDate>
C:          </launch:notice>
C:        </launch:create>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.3.3. General Create Form

The General Create Form of the extension to the EPP domain name mapping [RFC5731] includes the launch phase and optionally the object type to create. The OPTIONAL "type" attribute defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created.

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element contains the following child elements:

<launch:phase> Contains the value of the active launch phase of the server. The server SHOULD validate the value against the active server launch phase.

The following is an example <create> domain command using the <launch:create> extension for a "landrush" launch phase application:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        type="application">
C:        <launch:phase>landrush</launch:phase>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.3.4. Mixed Create Form

The Mixed Create Form supports a mix of the create forms, where for example the Sunrise Create Form (Section 3.3.1) and the Claims Create Form (Section 3.3.2) MAY be supported in a single command by including both the verified trademark information and the information related to the registrant's acceptance of the Claims Notice. The server MAY support the Mixed Create Form. The "custom" launch phase SHOULD be used when using the Mixed Create Form.

The following is an example <create> domain command using the <launch:create> extension, with using a mix of the Sunrise Create Form (Section 3.3.1) and the Claims Create Form (Section 3.3.2) by including both a mark and a notice:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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>domainone.example</domain:name>
C:        <domain:registrant>jdl234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        type="application">
C:        <launch:phase name="non-tmch-sunrise">custom</launch:phase>
C:        <launch:codeMark>
C:          <mark:mark
C:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:            ...
C:          </mark:mark>
C:        </launch:codeMark>
C:        <launch:notice>
C:          <launch:noticeID validatorID="tmch">
C:            49FD46E6C4B45C55D4AC
C:          </launch:noticeID>
C:          <launch:notAfter>2012-06-19T10:00:10.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2012-06-19T09:01:30.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```


3.3.5. Create Response

If the create was successful, the server MAY reply with the <launch:creData> element along with the regular EPP <resData> to indicate the server generated Application Identifier (Section 2.1), when multiple applications of a given domain name are supported; otherwise no extension is included with the regular EPP <resData>. The <launch:creData> element contains the following child elements:

<launch:phase> The phase of the application that mirrors the <launch:phase> element included in the <launch:create>.
<launch:applicationID> The application identifier of the application.

An example response when multiple overlapping applications are supported by the server:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:creData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.4. EPP <update> Command

This extension defines additional elements to extend the EPP <update> command to be used in conjunction with the domain name mapping.

A client MUST NOT pass the extension on an EPP <update> command to a server that does not support launch applications. A server that does not support launch applications during its launch phase MUST return an EPP error result code of 2102 when receiving an EPP <update> command with the extension.

Registry policies permitting, clients may update an application object by submitting an EPP <update> command along with a <launch:update> element to indicate the application object to be updated. The <launch:update> element contains the following child elements:

<launch:phase> The phase during which the application was submitted or is associated with.
<launch:applicationID> The application identifier for which the client wishes to update.

The following is an example <update> domain command with the <launch:update> extension to add and remove a name server of a sunrise application with the application identifier "abc123":

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update>
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>domain.example</domain:name>
C:          <domain:add>
C:            <domain:ns>
C:              <domain:hostObj>ns2.domain.example</domain:hostObj>
C:            </domain:ns>
C:          </domain:add>
C:          <domain:rem>
C:            <domain:ns>
C:              <domain:hostObj>ns1.domain.example</domain:hostObj>
C:            </domain:ns>
C:          </domain:rem>
C:        </domain:update>
C:      </update>
C:      <extension>
C:        <launch:update>
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:update>
C:        </extension>
C:      <clTRID>ABC-12345</clTRID>
C:    </command>
C:</epp>
```

This extension does not define any extension to the response of an <update> domain command. After processing the command, the server replies with a standard EPP response as defined in the EPP domain name mapping [RFC5731].

3.5. EPP <delete> Command

This extension defines additional elements to extend the EPP <delete> command to be used in conjunction with the domain name mapping.

A client MUST NOT pass the extension on an EPP <delete> command to a server that does not support launch applications. A server that does not support launch applications during its launch phase MUST return

an EPP error result code of 2102 when receiving an EPP <delete> command with the extension.

Registry policies permitting, clients MAY withdraw an application by submitting an EPP <delete> command along with a <launch:delete> element to indicate the application object to be deleted. The <launch:delete> element contains the following child elements:

<launch:phase> The phase during which the application was submitted or is associated with.
<launch:applicationID> The application identifier for which the client wishes to delete.

The following is an example <delete> domain command with the <launch:delete> extension:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

This extension does not define any extension to the response of a <delete> domain command. After processing the command, the server replies with a standard EPP response as defined in the EPP domain name mapping [RFC5731].

3.6. EPP <renew> Command

This extension does not define any extension to the EPP <renew> command or response described in the EPP domain name mapping [RFC5731].

3.7. EPP <transfer> Command

This extension does not define any extension to the EPP <transfer> command or response described in the EPP domain name mapping [RFC5731].

4. Formal Syntax

One schema is presented here that is the EPP Launch Phase Mapping schema.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

4.1. Launch Schema

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- o Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- o Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- o Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:ietf:params:xml:ns:launch-1.0"
  xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"
  xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <!--
  Import common element types.
  -->
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>
  <import namespace="urn:ietf:params:xml:ns:mark-1.0"/>
  <import namespace="urn:ietf:params:xml:ns:signedMark-1.0"/>

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      domain name extension schema
      for the launch phase processing.
    </documentation>
  </annotation>

  <!--
  Child elements found in EPP commands.
  -->
  <element name="check" type="launch:checkType"/>
  <element name="info" type="launch:infoType"/>
  <element name="create" type="launch:createType"/>
  <element name="update" type="launch:idContainerType"/>
  <element name="delete" type="launch:idContainerType"/>

  <!--
  Common container of id (identifier) element
  -->
  <complexType name="idContainerType">
    <sequence>
      <element name="phase"
        type="launch:phaseType"/>
      <element name="applicationID"
        type="launch:applicationIDType"/>
    </sequence>
  </complexType>

  <!--
```

```
Definition for application identifier
-->
<simpleType name="applicationIDType">
  <restriction base="token"/>
</simpleType>

<!--
Definition for launch phase. Name is an optional attribute
used to extend the phase type. For example, when
using the phase type value of &qt;custom&gt;, the name
can be used to specify the custom phase.
-->
<complexType name="phaseType">
  <simpleContent>
    <extension base="launch:phaseTypeValue">
      <attribute name="name" type="token"/>
    </extension>
  </simpleContent>
</complexType>

<!--
Enumeration of for launch phase values.
-->
<simpleType name="phaseTypeValue">
  <restriction base="token">
    <enumeration value="sunrise"/>
    <enumeration value="landrush"/>
    <enumeration value="claims"/>
    <enumeration value="open"/>
    <enumeration value="custom"/>
  </restriction>
</simpleType>

<!--
Definition for the sunrise code
-->
<simpleType name="codeValue">
  <restriction base="token">
    <minLength value="1"/>
  </restriction>
</simpleType>

<complexType name="codeType">
  <simpleContent>
    <extension base="launch:codeValue">
      <attribute name="validatorID"
        type="launch:validatorIDType" use="optional"/>
    </extension>
  </simpleContent>
</complexType>
```

```
        </extension>
      </simpleContent>
    </complexType>

    <!--
    Definition for the notice identifier
    -->
    <simpleType name="noticeIDValue">
      <restriction base="token">
        <minLength value="1"/>
      </restriction>
    </simpleType>

    <complexType name="noticeIDType">
      <simpleContent>
        <extension base="launch:noticeIDValue">
          <attribute name="validatorID"
            type="launch:validatorIDType" use="optional"/>
        </extension>
      </simpleContent>
    </complexType>

    <!--
    Definition for the validator identifier
    -->
    <simpleType name="validatorIDType">
      <restriction base="token">
        <minLength value="1"/>
      </restriction>
    </simpleType>

    <!--
    Possible status values for sunrise application
    -->
    <simpleType name="statusValueType">
      <restriction base="token">
        <enumeration value="pendingValidation"/>
        <enumeration value="validated"/>
        <enumeration value="invalid"/>
        <enumeration value="pendingAllocation"/>
        <enumeration value="allocated"/>
        <enumeration value="rejected"/>
        <enumeration value="custom"/>
      </restriction>
    </simpleType>

    <!--
    Status type definition
```



```
-->
<complexType name="statusType">
  <simpleContent>
    <extension base="normalizedString">
      <attribute name="s" type="launch:statusValueType"
        use="required"/>
      <attribute name="lang" type="language"
        default="en"/>
      <attribute name="name" type="token"/>
    </extension>
  </simpleContent>
</complexType>

<!--
codeMark Type that contains an optional code
with mark information.
-->
<complexType name="codeMarkType">
  <sequence>
    <element name="code" type="launch:codeType"
      minOccurs="0"/>
    <element ref="mark:abstractMark"
      minOccurs="0"/>
  </sequence>
</complexType>

<!--
Child elements for the create command
-->
<complexType name="createType">
  <sequence>
    <element name="phase" type="launch:phaseType"/>
    <choice minOccurs="0">
      <element name="codeMark" type="launch:codeMarkType"
        maxOccurs="unbounded"/>
      <element ref="smd:abstractSignedMark"
        maxOccurs="unbounded"/>
      <element ref="smd:encodedSignedMark"
        maxOccurs="unbounded"/>
    </choice>
    <element name="notice"
      type="launch:createNoticeType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="type" type="launch:objectType"/>
</complexType>

<!--
```

```
Type of launch object
-->
<simpleType name="objectType">
  <restriction base="token">
    <enumeration value="application"/>
    <enumeration value="registration"/>
  </restriction>
</simpleType>

<!--
Child elements of the create notice element.
-->
<complexType name="createNoticeType">
  <sequence>
    <element name="noticeID" type="launch:noticeIDType"/>
    <element name="notAfter" type="dateTime"/>
    <element name="acceptedDate" type="dateTime"/>
  </sequence>
</complexType>

<!--
Child elements of check (Claims Check Command).
-->
<complexType name="checkType">
  <sequence>
    <element name="phase" type="launch:phaseType"
      minOccurs="0"/>
  </sequence>
  <attribute name="type" type="launch:checkFormType"
    default="claims"/>
</complexType>

<!--
Type of check form
(claims check or availability check)
-->
<simpleType name="checkFormType">
  <restriction base="token">
    <enumeration value="claims"/>
    <enumeration value="avail"/>
    <enumeration value="trademark"/>
  </restriction>
</simpleType>
```

```
<!--
Child elements of info command.
-->
<complexType name="infoType">
  <sequence>
    <element name="phase" type="launch:phaseType"/>
    <element name="applicationID"
      type="launch:applicationIDType"
      minOccurs="0"/>
  </sequence>
  <attribute name="includeMark" type="boolean"
    default="false"/>
</complexType>

<!--
Child response elements.
-->
<element name="chkData" type="launch:chkDataType"/>
<element name="creData" type="launch:idContainerType"/>
<element name="infData" type="launch:infDataType"/>

<!--
<check> response elements.
-->
<complexType name="chkDataType">
  <sequence>
    <element name="phase" type="launch:phaseType"
      minOccurs="0"/>
    <element name="cd" type="launch:cdType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="cdType">
  <sequence>
    <element name="name" type="launch:cdNameType"/>
    <element name="claimKey" type="launch:claimKeyType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="cdNameType">
  <simpleContent>
    <extension base="eppcom:labelType">
      <attribute name="exists" type="boolean"
        use="required"/>
    </extension>
  </simpleContent>
```

```
</complexType>

<complexType name="claimKeyType">
  <simpleContent>
    <extension base="token">
      <attribute name="validatorID"
        type="launch:validatorIDType" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<!--
<info> response elements
-->
<complexType name="infDataType">
  <sequence>
    <element name="phase" type="launch:phaseType"/>
    <element name="applicationID"
      type="launch:applicationIDType"
      minOccurs="0"/>
    <element name="status" type="launch:statusType"
      minOccurs="0"/>
    <element ref="mark:abstractMark"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

</schema>
END
```

5. IANA Considerations

5.1. XML Namespace

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688].

Registration request for the launch namespace:

URI: urn:ietf:params:xml:ns:launch-1.0
Registrant Contact: See the "Author's Address" section of this document.
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the launch XML schema:

URI: urn:ietf:params:xml:schema:launch-1.0

Registrant Contact: See the "Author's Address" section of this document.

XML: See the "Formal Syntax" section of this document.

5.2. EPP Extension Registry

The EPP extension described in this document should be registered by the IANA in the EPP Extension Registry described in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

Reference: (insert reference to RFC version of this document)

Registrant Name and Email Address: IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

6. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 6982 [RFC6982] before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982 [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982 [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable

experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

6.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-ietf-eppext-launchphase.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

URL: http://www.verisigninc.com/en_US/channel-resources/domain-registry-products/epp-sdks

6.2. Verisign Consolidated Top Level Domain (CTLD) SRS

Organization: Verisign Inc.

Name: Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS)

Description: The Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS) implements the server-side of draft-ietf-eppext-launchphase for a variety of Top Level Domains (TLD's).

Level of maturity: Production

Coverage: The "signed mark" Mark Validation Model, the Claims Check Form for the EPP <check> Command, the Sunrise and Claims Forms for the EPP <create> Command of Launch Registrations and Launch Applications. For Launch Applications the Poll Messaging, the EPP <info> Command, the EPP <update> Command, and the EPP <delete> Command is covered.

Licensing: Proprietary

Contact: jgould@verisign.com

6.3. Verisign .COM / .NET SRS

Organization: Verisign Inc.

Name: Verisign .COM / .NET Shared Registry System (SRS)

Description: The Verisign Shared Registry System (SRS) for .COM, .NET and other IDN TLD's implements the server-side of draft-ietf-eppept-launchphase.

Level of maturity: Operational Test Environment (OTE)

Coverage: The "signed mark" Mark Validation Model, the Claims Check Form for the EPP <check> Command, the Sunrise and Claims Forms for the EPP <create> Command of Launch Registrations.

Licensing: Proprietary

Contact: jgould@verisign.com

6.4. REngin v3.7

Organization: Domain Name Services (Pty) Ltd

Name: REngin v3.7

Description: Server side implementation only

Level of maturity: Production

Coverage: All features from version 12 have been implemented

Licensing: Proprietary Licensing with Maintenance Contracts

Contact: info@dnservices.co.za

URL: <https://www.registry.net.za> and soon <http://dnservices.co.za>

6.5. RegistryEngine EPP Service

Organization: CentralNic

Name: RegistryEngine EPP Service

Description: Generic high-volume EPP service for gTLDs, ccTLDs and SLDs

Level of maturity: Deployed in CentralNic's production environment as well as two other gTLD registry systems, and two ccTLD registry systems.

Coverage: Majority of elements including TMCH sunrise, landrush and TM claims as well as sunrise applications validated using codes.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: <https://www.centralnic.com>

6.6. Neustar EPP SDK

Organization: Neustar

Name: Neustar EPP SDK

Description: The Neustar EPP SDK includes client implementation of draft-ietf-eppext-launchphase in both Java and C++.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: trung.tran@neustar.biz

6.7. gTLD Shared Registry System

Organization: Stichting Internet Domeinnaamregistratie Nederland (SIDN)

Name: gTLD Shared Registry System

Description: The gTLD SRS implements the server side of the draft-ietf-eppext-launchphase.

Level of maturity: (soon) Production

Coverage: The following parts of the draft are supported:

- Signed mark validation model using Digital Signature (Section 2.6.3)
- Claims Check Form (Section 3.1.1)
- Sunrise Create Form (Section 3.3.1)

Claims Create Form (Section 3.3.2)

The parts of the document not described here are not implemented.

Licensing: Proprietary

Contact: rik.ribbers@sidn.nl

7. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730], the EPP domain name mapping [RFC5731], and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

Updates to, and deletion of an application object must be restricted to clients authorized to perform the said operation on the object.

As information contained within an application, or even the mere fact that an application exists may be confidential. Any attempt to operate on an application object by an unauthorized client MUST be rejected with an EPP 2201 (authorization error) return code. Server policy may allow <info> operation with filtered output by clients other than the sponsoring client, in which case the <domain:infData> and <launch:infData> response SHOULD be filtered to include only fields that are publicly accessible.

8. Acknowledgements

The authors wish to acknowledge the efforts of the leading participants of the Community TMCH Model that led to many of the changes to this document, which include Chris Wright, Jeff Neuman, Jeff Eckhaus, and Will Shorter.

Special suggestions that have been incorporated into this document were provided by Jothan Frakes, Keith Gaughan, Seth Goldman, Michael Holloway, Jan Jansen, Rubens Kuhl, Ben Levac, Gustavo Lozano, Klaus Malorny, Alexander Mayrhofer, Patrick Mevzek, James Mitchell, Francisco Obispo, Mike O'Connell, Bernhard Reutner-Fischer, Trung Tran, Ulrich Wisser and Sharon Wodjenski.

9. Normative References

- [I-D.ietf-eppext-tmch-func-spec]
Lozano, G., "TMCH functional specifications", draft-ietf-eppext-tmch-func-spec-00 (work in progress), October 2015.

- [I-D.ietf-eppext-tmch-smd]
Lozano, G., "Mark and Signed Mark Objects Mapping", draft-ietf-eppext-tmch-smd-03 (work in progress), September 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<http://www.rfc-editor.org/info/rfc5731>>.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.
- [RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<http://www.rfc-editor.org/info/rfc7451>>.

Appendix A. Change History

A.1. Change from 00 to 01

1. Changed to use camel case for the XML elements.
2. Replaced "cancelled" status to "rejected" status.
3. Added the child elements of the <claim> element.
4. Removed the XML schema and replaced with "[TBD]".

A.2. Change from 01 to 02

1. Added support for both the ICANN and ARI/Neustar TMCH models.
2. Changed the namespace URI and prefix to use "launch" instead of "launchphase".
3. Added definition of multiple claim validation models.

4. Added the <launch:signedClaim> and <launch:signedNotice> elements.
5. Added support for Claims Info Command

A.3. Change from 02 to 03

1. Removed XSI namespace per Keith Gaughan's suggestion on the provreg list.
2. Added extensibility to the launch:status element and added the pendingAuction status per Trung Tran's feedback on the provreg list.
3. Added support for the Claims Check Command, updated the location and contents of the signedNotice, and replaced most references of Claim to Mark based on the work being done on the ARI/Neustar launch model.

A.4. Change from 03 to 04

1. Removed references to the ICANN model.
2. Removed support for the Claims Info Command.
3. Removed use of the signedClaim.
4. Revised the method for referring to the signedClaim from the XML Signature using the IDREF URI.
5. Split the launch-1.0.xsd into three XML schemas including launch-1.0.xsd, signeMark-1.0.xsd, and mark-1.0.xsd.
6. Split the "claims" launch phase to the "claims1" and "claims2" launch phases.
7. Added support for the encodedSignedMark with base64 encoded signedMark.
8. Changed the elements in the createNoticeType to include the noticeID, timestamp, and the source elements.
9. Added the class and effectiveDate elements to mark.

A.5. Change from 04 to 05

1. Removed reference to <smd:zone> in the <smd:signedMark> example.
2. Incorporated feedback from Bernhard Reutner-Fischer on the provreg mail list.
3. Added missing launch XML prefix to applicationIDType reference in the idContainerType of the Launch Schema.
4. Added missing description of the <mark:pc> element in the <mark:addr> element.
5. Updated note on replication of the EPP contact mapping elements in the Mark Contact section.

A.6. Change from 05 to 06

1. Removed the definition of the mark-1.0 and signedMark-1.0 and replaced with reference to draft-lozano-smd, that contains the definition for the mark, signed marked, and encoded signed mark.
2. Split the <launch:timestamp> into <launch:generatedDate> and <launch:acceptedDate> based on feedback from Trung Tran.
3. Added the "includeMark" optional attribute to the <launch:info> element to enable the client to request whether or not to include the mark in the info response.
4. Fixed state diagram to remove redundant transition from "invalid" to "rejected"; thanks Klaus Malorny.

A.7. Change from 06 to 07

1. Proof-read grammar and spelling.
2. Changed "pendingAuction" status to "pendingAllocation", changed "pending" to "pendingValidation" status, per proposal from Trung Tran and seconded by Rubens Kuhl.
3. Added text related to the use of RFC 5731 pendingCreate to the Application Identifier section.
4. Added the Poll Messaging section to define the use of poll messaging for intermediate state transitions and pending action poll messaging for final state transitions.

A.8. Change from 07 to 08

1. Added support for use of the launch statuses and poll messaging for Launch Registrations based on feedback from Sharon Wodjenski and Trung Tran.
2. Incorporated changes based on updates or clarifications in draft-lozano-tmch-func-spec-01, which include:
 1. Removed the unused <launch:generatedDate> element.
 2. Removed the <launch:source> element.
 3. Added the <launch:notAfter> element based on the required <tmNotice:notAfter> element.

A.9. Change from 08 to 09

1. Made <choice> element optional in <launch:create> to allow passing just the <launch:phase> in <launch:create> per request from Ben Levac.
2. Added optional "type" attribute in <launch:create> to enable the client to explicitly define the desired type of object (application or registration) to create to all forms of the create extension.

3. Added text that the server SHOULD validate the <launch:phase> element in the Launch Phases section.
4. Add the "General Create Form" to the create command extension to support the request from Ben Levac.
5. Updated the text for the Poll Messaging section based on feedback from Klaus Malorny.
6. Replaced the "claims1" and "claims2" phases with the "claims" phase based on discussion on the provreg list.
7. Added support for a mixed create model (Sunrise Create Model and Claims Create Model), where a trademark (encoded signed mark, etc.) and notice can be passed, based on a request from James Mitchell.
8. Added text for the handling of the overlapping "claims" and "landrush" launch phases.
9. Added support for two check forms (claims check form and availability check form) based on a request from James Mitchell. The availability check form was based on the text in draft-rbp-application-epp-mapping.

A.10. Change from 09 to 10

1. Changed noticeIDType from base64Binary to token to be compatible with draft-lozano-tmch-func-spec-05.
2. Changed codeType from base64Binary to token to be more generic.
3. Updated based on feedback from Alexander Mayrhofer, which include:
 1. Changed "extension to the domain name extension" to "extension to the domain name mapping".
 2. Changed use of 2004 return code to 2306 return code when phase passed mismatches active phase and sub-phase.
 3. Changed description of "allocated" and "rejected" statuses.
 4. Moved sentence on a synchronous <domain:create> command without the use of an intermediate application, then an Application Identifier MAY not be needed to the Application Identifier section.
 5. Restructured the Mark Validation Models section to include the "<launch:codeMark> element" sub-section, the "<mark:mark> element" sub-section, and the Digital Signature sub-section.
 6. Changed "Registries may" to "Registries MAY".
 7. Changed "extensed" to "extended" in "Availability Check Form" section.
 8. Broke the mix of create forms in the "EPP <create> Command" section to a fourth "Mixed Create Form" with its own sub-section.
 9. Removed "displayed or" from "displayed or accepted" in the <launch:acceptedDate> description.

10. Replaced "given domain name is supported" with "given domain name are supported" in the "Create Response" section.
 11. Changed the reference of 2303 (object does not exist) in the "Security Considerations" section to 2201 (authorization error).
 12. Added arrow from "invalid" status to "pendingValidation" status and "pendingAllocation" status to "rejected" status in the State Transition Diagram.
 4. Added the "C:" and "S:" example prefixes and related text in the "Conventions Used in This Document" section.
- A.11. Change from 10 to 11
1. Moved the claims check response <launch:chkData> element under the <extension> element instead of the <resData> element based on the request from Francisco Obispo.
- A.12. Change from 11 to 12
1. Added support for multiple validator identifiers for claims notices and marks based on a request and text provided by Mike O'Connell.
 2. Removed domain:exDate element from example in section 3.3.5 based on a request from Seth Goldman on the provreg list.
 3. Added clarifying text for clients not passing the launch extension on update and delete commands to servers that do not support launch applications based on a request from Sharon Wodjenski on the provreg list.
- A.13. Change from 12 to WG 00
1. Changed to eppext working group draft by changing draft-tan-epp-launchphase to draft-ietf-eppext-launchphase and by changing references of draft-lozano-tmch-smd to draft-ietf-eppext-tmch-smd.
- A.14. Change WG 00 to WG 01
1. Removed text associated with support for the combining of status values based on feedback from Patrick Mevzek on the provreg mailing list, discussion on the eppext mailing list, and discussion at the eppext IETF meeting on March 6, 2014.
- A.15. Change WG 01 to WG 02
1. Changed the <launch:claim> element to be zero or more elements and the <launch:notice> element to be one or more elements in the

Claims Create Form. These changes were needed to be able to support more than one concurrent claims services.

A.16. Change WG 02 to WG 03

1. Added the "Implementation Status" section based on an action item from the eppext IETF-91 meeting.
2. Moved Section 7 "IANA Considerations" and Section 9 "Security Considerations" before Section 5 "Acknowledgements". Moved "Change Log" Section to end.
3. Updated the text for the Claims Check Form and the Claims Create Form to support checking for the need of the claims notice and passing the claims notice outside of the "claims" phase.
4. Added the new Trademark Check Form to support determining whether or not a trademark exists that matches the domain name independent of whether a claims notice is required on create. This was based on a request from Trung Tran and a discussion on the eppext mailing list.

A.17. Change WG 03 to WG 04

1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.

A.18. Change WG 04 to WG 05

1. Added a missing comma to the description of the <launch:phase> element, based on feedback from Keith Gaughan on the eppext mailing list.
2. Added the SIDN implementation status information.
3. Fixed a few indentation issues in the samples.

A.19. Change WG 05 to WG 06

1. Removed duplicate "TMCH Functional Specification" URIs based on feedback from Scott Hollenbeck on the eppext mailing list.
2. Changed references of example?.tld to domain?.example to be consistent with RFC 6761 based on feedback from Scott Hollenbeck on the eppext mailing list.
3. A template was added to section 5 to register the XML schema in addition to the namespace based on feedback from Scott Hollenbeck on the eppext mailing list.

A.20. Change WG 06 to WG 07

1. Changed reference of lozano-tmch-func-spec to ietf-eppext-tmch-func-spec.

Authors' Addresses

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US

Email: jgould@verisign.com
URI: <http://www.verisigninc.com>

Wil Tan
Cloud Registry
Suite 32 Seabridge House
377 Kent St
Sydney, NSW 2000
AU

Phone: +61 414 710899
Email: wil@cloudregistry.net
URI: <http://www.cloudregistry.net>

Gavin Brown
CentralNic Ltd
35-39 Mooregate
London, England EC2R 6AR
GB

Phone: +44 20 33 88 0600
Email: gavin.brown@centralnic.com
URI: <https://www.centralnic.com>

Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 6, 2015

S. Hollenbeck
Verisign Labs
December 3, 2014

Extension Registry for the Extensible Provisioning Protocol
draft-ietf-epext-reg-10

Abstract

The Extensible Provisioning Protocol (EPP) includes features to add functionality by extending the protocol. It does not, however, describe how those extensions are managed. This document describes a procedure for the registration and management of extensions to EPP and it specifies a format for an IANA registry to record those extensions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 6, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Extension Specification and Registration Procedure	3
2.1. Extension Specification	3
2.1.1. Designated Expert Evaluation Criteria	3
2.2. Registration Procedure	4
2.2.1. Required Information	4
2.2.2. Registration Form	5
2.2.3. Registration Processing	8
2.2.4. Updating Registry Entries	8
3. IANA Considerations	8
4. Security Considerations	11
5. Acknowledgements	11
6. References	11
6.1. Normative References	11
6.2. Informative References	12
Appendix A. Change Log	12
Author's Address	12

1. Introduction

Domain name registries implement a variety of operational and business models. The differences in these models made it impossible to develop a "one size fits all" provisioning protocol, so the Extensible Provisioning Protocol (EPP, [RFC5730]) was designed to focus on a minimal set of common functionality with built-in extension capabilities that allow new features to be specified on an "as needed" basis. Guidelines for extending EPP are documented in Informational RFC 3735 [RFC3735].

RFCs 3735 and 5730 do not describe how extension development can be managed and coordinated. This has led to a situation in which server operators can develop different extensions to address similar needs, such as the provisioning of Value Added Tax (VAT) information. Clients then need to support multiple extensions that serve similar purposes, and interoperability suffers.

An IANA registry can be used to help manage and coordinate the development of protocol extensions. This document describes an IANA registry that can be used to coordinate the development of EPP extensions.

2. Extension Specification and Registration Procedure

This section describes the format of an IANA registry and the procedures used to populate and manage registry entries.

2.1. Extension Specification

This registry uses the "Specification Required" policy described in RFC 5226 [RFC5226]. An English language version of the extension specification will be referenced from the registry, though non-English versions of the specification can also be provided. Note that Section 2.1 of RFC 3735 [RFC3735] provides specific guidelines for documenting EPP extensions.

Note that the "Specification Required" policy implies review by a Designated Expert. Section 3 of RFC 5226 describes the role of Designated Experts and the function they perform.

2.1.1. Designated Expert Evaluation Criteria

A high-level description of the role of the Designated Expert is described in Section 3.2 RFC 5226. Specific guidelines for the appointment of Designated Experts and evaluation of EPP extensions are provided here.

The IESG should appoint a small pool of individuals (perhaps 3 - 5) to serve as designated experts as described in Section 3.2 of RFC 5226. The pool should have a single administrative chair who is appointed by the IESG. The designated experts should use the existing eppext mailing list (eppext@ietf.org) for public discussion of registration requests. This implies that the mailing list should remain open after the work of the EPPEXT working group has concluded.

Extensions should be evaluated for architectural soundness using the guidelines described in RFC 3735 [RFC3735], including the Security Considerations section of that document. Expert evaluation should explicitly include consideration of the privacy consequences of proposed extensions, and, at a minimum, ensure that any privacy considerations are fully documented in the relevant specification(s).

The results of the evaluation should be shared via email with the registrant and the eppext mailing list. Issues discovered during the evaluation can be corrected by the registrant and those corrections can be submitted to the designated experts until the designated experts explicitly decide to accept or reject the registration request. The designated experts must make an explicit decision and that decision must be shared via email with the registrant and the eppext mailing list. If the specification for an extension is an

IETF Standards Track document, no review is required by the Designated Expert.

Designated experts should be permissive in their evaluation of requests to register extensions that have been implemented and deployed by at least one registry/registrar pair. This implies that it may indeed be possible to register multiple extensions that provide the same functionality. Requests to register extensions that have not been deployed should be evaluated with a goal of reducing functional duplication. A potential registrant who submits a request to register a new, un-deployed extension that includes similar functionality to an existing, registered extension should be made aware of the existing extension. The registrant should be asked to reconsider their request given the existence of a similar extension. Should they decline to do so perceived similarity should not be a sufficient reason for rejection as long as all other requirements are met.

2.2. Registration Procedure

The registry contains information describing each registered extension. Registry entries are created and managed by sending forms to IANA that describe the extension and the operation to be performed on the registry entry.

2.2.1. Required Information

Name of Extension: A case-insensitive, ASCII text string that contains the name of the extension specification. Non-ASCII representations of the extension name can be included in the "Notes" described below.

Document Status: The document status ("Informational", "Standards Track", etc.) of the specification document. For documents that are not RFCs, this will always be "Informational".

Reference: A reference to the specification of this extension. This could be an RFC number or some other pointer to the document defining the extension.

Registrant Name and Email Address: The name and email address of the person that is responsible for managing the registry entry. If the registration is of an IETF Standards Track document, this can simply be listed as "IESG, <iesg@ietf.org>".

TLDs: A text string containing the top-level domain name (or domain names), including the preceding ".", for which the extension has been specified (e.g., ".org"). If there are multiple TLDs, they are given

as a list of domain names separated by commas, (e.g. ".com, .net"). Internationalized Domain Name (IDN) TLDs should be specified in A-label [RFC5890] format. If the extension is not associated with a specific top-level domain, the case-insensitive text string "Any" can be used to indicate that.

IPR Disclosure: A pointer to any Intellectual Property Rights (IPR) disclosure document(s) related to this extension, or "None" if there are no such disclosures. This can be an IPR disclosure filed with the IETF in accordance with RFC 3979 [RFC3979] as updated by RFC 4879 [RFC4879] if the extension is part of an IETF Contribution, or can be other IPR disclosure documents identifying the claimed intellectual property rights and terms of use for extensions that are not part of an IETF Contribution.

Status: Either "Active" or "Inactive". The "Active" status is used for extensions that are currently implemented and in use. The "Inactive" status is used for extensions that are not implemented or are otherwise not being used.

Notes: Either "None" or other text that describes optional notes to be included with the registered extension. If the Status value is "Inactive" text should be included to describe how and when this state was reached.

2.2.2. Registration Form

The required information must be formatted consistently using the following form. Form field names and values may appear on the same line:

-----BEGIN FORM-----

Name of Extension:

<text string> (quotes are optional)

Document Status: <document status>

Reference: <RFC number, URL, etc.>

Registrant Name and Email Address:

<registrant name>, <email address>

TLDs: "Any" | <one or more TLD text strings separated by commas>

IPR Disclosure: "None" | <URL>

Status: "Active" | "Inactive"

Notes: "None" | <optional text>
-----END FORM-----

Example form with RFC specification:

-----BEGIN FORM-----

Name of Extension:

"An Extension RFC for the Extensible Provisioning Protocol (EPP)"

Document Status:

Standards Track

Reference:

<http://tools.ietf.org/html/rfcXXXX>

Registrant Name and Email Address:

IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

-----END FORM-----

Example form with non-RFC specification:

-----BEGIN FORM-----

Name of Extension:

"An Example Extension for the .example Top-Level Domain"

Document Status:

Informational

Reference:

<http://www.example.com/html/example-epp-ext.txt>

Registrant Name and Email Address:

John Doe, jdoe@example.com

TLDs: .example

IPR Disclosure:

<http://www.example.com/ipr/example-epp-ext-ipr.html>

Status: Active

Notes: None

-----END FORM-----

2.2.3. Registration Processing

Registrants should send each registration form to IANA with a single record for incorporation into the registry. Send the form via email to <iana@iana.org>, and include a subject line indicating whether the enclosed form represents an insertion of a new record (indicated by the word "INSERT" in the subject line) or a replacement of an existing record (indicated by the word "MODIFY" in the subject line). At no time can a record be deleted from the registry. On receipt of the registration request, IANA will initiate review by the designated expert(s), who will evaluate the request using the criteria in Section 2.1.1, in consultation with the eppext mailing list.

2.2.4. Updating Registry Entries

When submitting changes to existing registry entries, include text in the "Notes" field of the registration form describing the change. Under normal circumstances, registry entries are only be updated by the registrant. If the registrant becomes unavailable or otherwise unresponsive, the designated expert can submit a registration form to IANA to update the registrant information. Entries can change state from "Active" to "Inactive" and back again as long as state change requests conform to the processing requirements identified in this document. In addition to entries that become "Inactive" due to a lack of implementation, entries for which a specification becomes consistently unavailable over time should be marked "Inactive" by the designated expert until such time as the specification again becomes reliably available.

3. IANA Considerations

IANA is requested to create a new protocol registry to manage EPP extensions. This registry should appear under its own heading on IANA's protocol listings, using the same title as the name of the registry. The information to be registered and the procedures to be followed in populating the registry are described in Section 2.

Name of registry: Extensions for the Extensible Provisioning Protocol

Section at <http://www.iana.org/protocols>:

Registry Title: Extensions for the Extensible Provisioning Protocol

Registry Name: Extensions for the Extensible Provisioning Protocol

Registration Procedure: Specification Required

Reference: this draft

Required information: See Section 2.2.1.

Review process: "Specification Required" as described in RFC 5226 [RFC5226].

Size, format, and syntax of registry entries: See Section 2.2.1.

Initial assignments and reservations:

```
-----BEGIN FORM-----
Name of Extension:
"Domain Registry Grace Period Mapping for the
Extensible Provisioning Protocol (EPP)"

Document Status:
Standards Track

Reference:
http://tools.ietf.org/html/rfc3915

Registrant Name and Email Address:
IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None
-----END FORM-----
```

```
-----BEGIN FORM-----
Name of Extension:
"E.164 Number Mapping for the
Extensible Provisioning Protocol (EPP)"

Document Status:
Standards Track

Reference:
http://tools.ietf.org/html/rfc4114

Registrant Name and Email Address:
IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None
-----END FORM-----

-----BEGIN FORM-----
Name of Extension:
"ENUM Validation Information Mapping for the
Extensible Provisioning Protocol"

Document Status:
Standards Track

Reference:
http://tools.ietf.org/html/rfc5076

Registrant Name and Email Address:
IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None
-----END FORM-----
```

-----BEGIN FORM-----
Name of Extension:
"Domain Name System (DNS) Security Extensions Mapping for the
Extensible Provisioning Protocol (EPP)"

Document Status:
Standards Track

Reference:
<http://tools.ietf.org/html/rfc5910>

Registrant Name and Email Address:
IESG, <iesg@ietf.org>

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None
-----END FORM-----

In addition, the form used to populate and manage the registry is to be added to the table of Protocol Registration Forms maintained by IANA.

4. Security Considerations

This document introduces no new security considerations to EPP. However, extensions should be evaluated according to the Security Considerations of RFC 3735 [RFC3735].

5. Acknowledgements

The information described in the registry is based on a suggestion posted to the provreg mailing list by Jay Daley in August 2013.

6. References

6.1. Normative References

- [RFC3979] Bradner, S., "Intellectual Property Rights in IETF Technology", BCP 79, RFC 3979, March 2005.
- [RFC4879] Narten, T., "Clarification of the Third Party Disclosure Procedure in RFC 3979", BCP 79, RFC 4879, April 2007.

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

6.2. Informative References

- [RFC3735] Hollenbeck, S., "Guidelines for Extending the Extensible Provisioning Protocol (EPP)", RFC 3735, March 2004.

Appendix A. Change Log

- Initial -00: First working group version.
- 01: Added initial registry entries to Section 3.
 - 02: Spelling corrections. Added Section 2.1.1. Added "Notes" field to the registration template.
 - 03: Added reference to Section 2.1 of RFC 3735 in Section 2.1.
 - 04: Added "Status" field to the registration template. Fixed typo in Section 2.1.1. Reformatted examples and initial registry entries.
 - 05: Added text to clarify how existing registry entries can and can't be edited.
 - 06: Modified text in Section 2.1.1 to make it clear that it is possible to register functionally similar extensions.
 - 07: Address WG last call comments.
 - 08: Address AD review comments.
 - 09: Address IETF last call and IESG review comments.
 - 10: Re-address IETF last call and IESG review comments. Added text to note that the extension name must be represented using ASCII characters.

Author's Address

Scott Hollenbeck
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
US

Email: shollenbeck@verisign.com
URI: <http://www.verisignlabs.com/>

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2016

G. Lozano
ICANN
March 9, 2016

Mark and Signed Mark Objects Mapping
draft-ietf-eppext-tmch-smd-06

Abstract

Domain Name Registries (DNRs) may operate in special modes for certain periods of time enabling trademark holders to protect their rights during the introduction of a Top Level Domain (TLD).

One of those special modes of operation is the Sunrise Period. The Sunrise Period allows trademark holders an advance opportunity to register domain names corresponding to their trademarks before names are generally available to the public.

This document describes the format of a mark and a digitally signed mark used by trademark holders for registering domain names during the sunrise phase of generic Top Level Domains (gTLDs). Three types of mark objects are defined in this specification: registered trademarks, court-validated marks, and marks protected by statute or treaty.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Object Description	4
2.1. Holder and Contacts objects	4
2.2. Mark	6
2.3. Signed Mark	9
2.4. Encoded Signed Mark	13
3. Formal Syntax	13
3.1. Signed Mark Schema	13
3.2. Mark Schema	15
4. Implementation Status	21
4.1. Verisign EPP SDK	21
4.2. Verisign Consolidated Top Level Domain (CTLD) SRS	22
4.3. Verisign .COM / .NET SRS	22
4.4. REngin v3.7	22
4.5. Uniregistry Corp. Shared Registry System (uSRS)	23
5. Acknowledgements	23
6. IANA Considerations	23
7. Security Considerations	24
8. References	25
8.1. Normative References	25
8.2. Informative References	26
Author's Address	26

1. Introduction

Domain Name Registries (DNRs) may operate in special modes for certain periods of time enabling trademark holders to protect their rights during the introduction of a Top Level Domain (TLD).

One of those special modes of operation is the Sunrise Period. The Sunrise Period allows trademark holders an advance opportunity to register domain names corresponding to their trademarks before names are generally available to the public.

This specification was defined as part of the development of the ICANN Trademark Clearinghouse (TMCH). The ICANN TMCH is a global repository for trademark data used by DNRs, registrars and trademark holders during the registration process of domain names.

This document describes a mapping of the common elements found in trademark data. A digitally signed mark format is defined in order to support digital signatures on the mark. Finally a mapping for encoding the signed mark document is defined.

Three types of mark objects are defined in this specification: registered trademarks, court-validated marks, and marks protected by statute or treaty.

This specification is intended to be used in the gTLD space, but nothing precludes the use of this format by other entities.

The detailed policy regarding the public key infrastructure (PKI), authorized validators, and other requirements must be defined based on the local policy of the entities using this specification. In the case of gTLDs, the detailed policy regarding the use of this specification is defined in the Rights Protection Mechanism Requirements document (see [ICANN-TMCH]), and the PKI is defined in [I-D.ietf-eppext-tmch-func-spec]. Implementations will need to implement such a PKI (or an equivalent) in order for the signatures defined in this document to have any useful semantics.

The objects specified in this document can be referenced by application protocols like the Extensible Provisioning Protocol (EPP), defined in [RFC5730].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

XML (EXtensible Markup Language) is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented in order to develop a conforming implementation.

"signedMark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:signedMark-1.0". The XML namespace prefix "smd" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

"mark-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:mark-1.0". The XML namespace prefix "mark" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

2. Object Description

This section defines the Mark and Signed Mark objects. Empty complex element types and abstract elements are defined to support additional Mark and Signed Mark definitions using XML schema substitution groups. Support for replacement through the XML schema substitution groups is included in the description of the objects.

This section defines some elements as OPTIONAL. If an element is not defined as OPTIONAL, then it MUST be included in the object.

The following elements are defined as telephone numbers: <mark:voice>, <mark:fax> and <smd:voice>. The representation of telephone numbers in this specification is derived from structures defined in [ITU.E164.2005]. Telephone numbers described in this mapping are character strings that MUST begin with a plus sign ("+", ASCII value 0x002B), followed by a country code defined in [ITU.E164.2005], followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number. An optional "x" attribute is provided to note telephone extension information.

The following elements are defined as email addresses: <mark:email> and <smd:email>. Email address syntax is defined in [RFC5322].

2.1. Holder and Contacts objects

Marks are linked to Holder objects and optionally linked to Contact objects. This section defines the <mark:holder> and <mark:contact> objects.

- o The child elements of <mark:holder> include:

- * A <mark:name> element that contains the name of the individual holder of the mark. At least one of <mark:name> and <mark:org>

- MUST be specified, and `<mark:name>` is OPTIONAL if `<mark:org>` is specified.
- * A `<mark:org>` element that contains the name of the organization holder of the mark. At least one of `<mark:name>` and `<mark:org>` MUST be specified, and `<mark:org>` is OPTIONAL if `<mark:name>` is specified.
 - * A `<mark:addr>` element that contains the address information of the holder of a mark. A `<mark:addr>` contains the following child elements:
 - + One, two or three OPTIONAL `<mark:street>` elements that contains the holder's street address.
 - + A `<mark:city>` element that contains the holder's city.
 - + An OPTIONAL `<mark:sp>` element that contains the holder's state or province.
 - + An OPTIONAL `<mark:pc>` element that contains the holder's postal code.
 - + A `<mark:cc>` element that contains the holder's country code. This a two-character code from [ISO3166-2].
 - * An OPTIONAL `<mark:voice>` element that contains the holder's voice telephone number.
 - * An OPTIONAL `<mark:fax>` element that contains the holder's facsimile telephone number.
 - * An OPTIONAL `<mark:email>` element that contains the email address of the holder.
- o The child elements of `<mark:contact>` include:
- * A `<mark:name>` element that contains name of the responsible person.
 - * An OPTIONAL `<mark:org>` element that contains the name of the organization of the contact.
 - * A `<mark:addr>` element that contains the address information of the contact. A `<mark:addr>` contains the following child elements:

- + One, two or three OPTIONAL `<mark:street>` elements that contains the contact's street address.
- + A `<mark:city>` element that contains the contact's city.
- + An OPTIONAL `<mark:sp>` element that contains the contact's state or province.
- + An OPTIONAL `<mark:pc>` element that contains the contact's postal code.
- + A `<mark:cc>` element that contains the contact's country code. This a two-character code from [ISO3166-2].
- * A `<mark:voice>` element that contains the contact's voice telephone number.
- * An OPTIONAL `<mark:fax>` element that contains the contact's facsimile telephone number.
- * A `<mark:email>` element that contains the contact's email address.

2.2. Mark

A `<mark:mark>` element that describes an applicant's prior right to a given domain name.

A `<mark:mark>` element substitutes for the `<mark:abstractMark>` abstract element to define a concrete definition of a mark. The `<mark:abstractMark>` element can be replaced by other mark definitions using the XML schema substitution groups feature.

The child elements of the `<mark:mark>` element include:

One or more `<mark:trademark>`, `<mark:treatyOrStatute>` and `<mark:court>` elements that contains the detailed information of marks.

- o A `<mark:trademark>` element that contains the following child elements:
 - * A `<mark:id>` that uniquely identifies a mark in relation to a repository of marks potentially maintained by more than one issuer. A `<mark:id>` value is a concatenation of the local identifier, followed by a hyphen ("-", ASCII value 0x002D), followed by the issuer identifier.
 - * A `<mark:markName>` element that contains the mark text string.

- * One or more <mark:holder> elements that contains the information of the holder of the mark. An "entitlement" attribute is used to identify the entitlement of the holder, possible values are: owner, assignee and licensee.
- * Zero or more OPTIONAL <mark:contact> elements that contains the information of the representative of the mark registration. A "type" attribute is used to identify the type of contact, possible values are: owner, agent or thirdparty.
- * A <mark:jurisdiction> element that contains the two-character code of the jurisdiction where the trademark was registered. This is a two-character code from [WIPO.ST3].
- * Zero or more OPTIONAL <mark:class> elements that contain the WIPO Nice Classification class numbers of the mark as defined in the WIPO Nice Classification [WIPO-NICE-CLASSES].
- * Zero or more OPTIONAL <mark:label> elements that contain the A-label form (as defined in [RFC5890]) of the label that correspond to the <mark:markName>.
- * A <mark:goodsAndServices> element that contains the full description of the goods and services mentioned in the mark registration document.
- * An OPTIONAL <mark:apId> element that contains the trademark application ID registered in the trademark office.
- * An OPTIONAL <mark:apDate> element that contains the date the trademark was applied for.
- * A <mark:regNum> element that contains the trademark registration number registered in the trademark office.
- * A <mark:regDate> element that contains the date the trademark was registered.
- * An OPTIONAL <mark:exDate> element that contains the expiration date of the trademark.
- o A <mark:treatyOrStatute> element that contains the following child elements:
 - * A <mark:id>, see definition in the <mark:trademark> section above.

- * A <mark:markName>, see definition in the <mark:trademark> section above.
- * One or more <mark:holder>, see definition in the <mark:trademark> section above.
- * Zero or more OPTIONAL <mark:contact>, see definition in the <mark:trademark> section above.
- * One or more <mark:protection> elements that contain the countries and region of the country where the mark is protected. The <mark:protection> element contains the following child elements:
 - + A <mark:cc> element that contains the two-character code of the country in which the mark is protected. This is a two-character code from [ISO3166-2].
 - + An OPTIONAL <mark:region> element that contains the name of a city, state, province or other geographic region of <mark:country> in which the mark is protected.
 - + Zero or more OPTIONAL <mark:ruling> elements that contains the two-character code of the national territory in which the statute or treaty is applicable. This is a two-character code from [ISO3166-2].
 - + Zero or more OPTIONAL <mark:label>, see definition in the <mark:trademark> section above.
- * A <mark:goodsAndServices>, see definition in the <mark:trademark> section above.
- * A <mark:refNum> element that contains the serial number of the mark.
- * A <mark:proDate> element that contains the date of protection of the mark.
- * A <mark:title> element that contains the title of the treaty or statute.
- * A <mark:execDate> element that contains the execution date of the treaty or statute.
- o A <mark:court> element that contains the following child elements:

- * A <mark:id>, see definition in the <mark:trademark> section above.
- * A <mark:markName>, see definition in the <mark:trademark> section above.
- * One or more <mark:holder>, see definition in the <mark:trademark> section above.
- * Zero or more OPTIONAL <mark:contact>, see definition in the <mark:trademark> section above.
- * Zero or more OPTIONAL <mark:label>, see definition in the <mark:trademark> section above.
- * A <mark:goodsAndServices>, see definition in the <mark:trademark> section above.
- * A <mark:refNum> element that contains the reference number of the court's opinion.
- * A <mark:proDate> element that contains the date of protection of the mark.
- * A <mark:cc> element that contains the two-character code of the country where the court is located. This a two-character code from [ISO3166-2].
- * Zero or more OPTIONAL <mark:region> elements that contains the name of a city, state, province or other geographic region of <mark:cc> in which the mark is protected. In case <mark:region> is specified a default-deny approach MUST be assumed regarding the regions of a country.
- * A <mark:courtName> element that contains the name of the court.

2.3. Signed Mark

The <smd:signedMark> is a digitally signed XML document using XML Signature [XMLDSIG]. The <smd:signedMark> XML document (SMD) includes a required "id" attribute of type XSD ID for use with an IDREF URI from the Signature element. The SMD might be transmitted as part of an already XML based protocol, therefore exclusive XML canonicalization as defined in [XMLC14N] MUST be used.

A <smd:signedMark> element substitutes for the <smd:abstractSignedMark> abstract element to define a concrete definition of a signed mark. The <smd:abstractSignedMark> element

can be replaced by other signed mark definitions using the XML schema substitution groups feature.

The child elements of the <smd:signedMark> element include:

- o The <smd:id> that uniquely identifies an SMD in relation to a repository of SMDs potentially maintained by more than one issuer. The <smd:id> value is a concatenation of the local identifier, followed by a hyphen ("-", ASCII value 0x002D), followed by the issuer identifier.
- o A <smd:issuerInfo> element that contains the information of the issuer of the mark registration. A "issuerID" attribute is used to specify the issuer identifier. The child elements include:
 - * A <smd:org> element that contains the organization name of the issuer.
 - * A <smd:email> element that contains the issuer customer support email address.
 - * An OPTIONAL <smd:url> element that contains the HTTP or HTTPS URL of the issuer's site.
 - * An OPTIONAL <smd:voice> element that contains the issuer's voice telephone number.
- o A <smd:notBefore> element that contains the creation date and time of the SMD.
- o A <smd:notAfter> element that contains the expiration date and time of the SMD.
- o A <mark:mark> element that contains the mark information as defined in the Mark (Section 2.2) section.

The following is an example of an SMD:

```
<?xml version="1.0" encoding="UTF-8"?>
<smd:signedMark xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
id="smd1">
  <smd:id>0000001751376056503931-65535</smd:id>
  <smd:issuerInfo issuerID="65535">
    <smd:org>ICANN TMCH TESTING TMV</smd:org>
    <smd:email>notavailable@example.com</smd:email>
    <smd:url>https://www.example.com</smd:url>
    <smd:voice>+32.000000</smd:voice>
  </smd:issuerInfo>
```

```
<smd:notBefore>2013-08-09T13:55:03.931Z</smd:notBefore>
<smd:notAfter>2017-07-23T22:00:00.000Z</smd:notAfter>
<mark:mark xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
  <mark:trademark>
    <mark:id>00052013734689731373468973-65535</mark:id>
    <mark:markName>Test & Validate</mark:markName>
    <mark:holder entitlement="owner">
      <mark:org>Ag corporation</mark:org>
      <mark:addr>
        <mark:street>1305 Bright Avenue</mark:street>
        <mark:city>Arcadia</mark:city>
        <mark:sp>CA</mark:sp>
        <mark:pc>90028</mark:pc>
        <mark:cc>US</mark:cc>
      </mark:addr>
    </mark:holder>
    <mark:contact type="agent">
      <mark:name>Tony Holland</mark:name>
      <mark:org>Ag corporation</mark:org>
      <mark:addr>
        <mark:street>1305 Bright Avenue</mark:street>
        <mark:city>Arcadia</mark:city>
        <mark:sp>CA</mark:sp>
        <mark:pc>90028</mark:pc>
        <mark:cc>US</mark:cc>
      </mark:addr>
      <mark:voice>+1.2025562302</mark:voice>
      <mark:fax>+1.2025562301</mark:fax>
      <mark:email>info@agcorporation.com</mark:email>
    </mark:contact>
    <mark:jurisdiction>US</mark:jurisdiction>
    <mark:class>15</mark:class>
    <mark:label>testandvalidate</mark:label>
    <mark:label>test---validate</mark:label>
    <mark:label>testand-validate</mark:label>
    <mark:label>test-et-validate</mark:label>
    <mark:label>test-validate</mark:label>
    <mark:label>test--validate</mark:label>
    <mark:label>test-etvalidate</mark:label>
    <mark:label>testetvalidate</mark:label>
    <mark:label>testvalidate</mark:label>
    <mark:label>testet-validate</mark:label>
    <mark:goodsAndServices>guitar</mark:goodsAndServices>
    <mark:regNum>1234</mark:regNum>
    <mark:regDate>2012-12-31T23:00:00.000Z</mark:regDate>
  </mark:trademark>
</mark:mark>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
```

```
<SignedInfo>
  <CanonicalizationMethod
    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
  <SignatureMethod
    Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
  <Reference URI="#smd1">
    <Transforms>
      <Transform
        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
      </Transforms>
    <DigestMethod
      Algorithm="http://www.w3.org/2001/04/xmenc#sha256" />
    <DigestValue>wgyW3nZPoEfpptlhRILKnOQnbdU6ArM7ShrAfHgDFg=</DigestValue>
  </Reference>
</SignedInfo>
<SignatureValue>
jMu4PfyQGijBF0GWSEPFJCjmywCEqR2h4LD+ge6XQ+JnmKFFCuCZS/3SLKAX0L1w
QDFO2e0Y69k2G7/LGE37X3vOflobFM1oGwja8+GMVraoto5xAd4/AF7eHukgAymD
o9toxo2h0yV4A4PmXzsU6S86XtCcUE+S/WM72nyn47zoUCzzPKHZBRyeWehVFQ+
jYRMIAMzM57HHQA+6eaXefRvtPETgU04aVIVSugc4OUAZZwbYcZrC6w0aQqqqAZi
30aPOBYbAvHMSmWSS+hFkbshomJfHxb97TD2grlYnRQIzqXk7WbHWy2SYdA+sI/Z
ipJsXNa6osTUw1CzA7jfwA==
</SignatureValue>
<KeyInfo>
  <X509Data>
    <X509Certificate>
MIIESTCCAzGgAwIBAgIBAJANBgkqhkiG9w0BAQsFADBiMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJQ0FO
TiBUTUNIMRswGQYDVQQDEXJJQ0FOTiBUTUNIIFRFU1QgQ0EwHhcNMjMwMjA4MDAw
MDAwWWhcNMTgMjA3MjM1OTU5WjBsmQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0Ex
FDASBgNVBACTC0xvcyBBbmdlbGVzMRcwFQYDVQQKEw5WYWxpZGF0b3IgcVE1DSDEh
MB8GA1UEAxMYVmFsaWRhdG9yIFRnQ0ggVEVTVCBDRVJUMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAo/cwvXhbVY10RDWWvoveZpETVZVVcMCovUVNg/sw
WinuMgEWgVQFrz0xA04pEhXCFVv4evbUpekJ5buqU1gmQyOsCKQ1hOHTdPjvkC5u
pDqa5lFlk0TMaMkIQjs7aUKCmA4RG4tTTGK/EjR1ix8/D0gHYVRldy1YPrMP+ou7
5bOVnIos+HifRatrIv4qEqwLL4FTZAUpaCa2BmgXfy2CSRQbxD5Or1gcSa3vurh5
sPMCnxqaXmIXmQipS+DuEBqMM8tldaN7RYojUEKRGVsNk5i9y2/7sjn1zyyUPf7v
L4GgDYqhJYWV61DnXgx/Jd6CWxvsndF6scscQzUTel+hywIDAQABo4H/MIH8MAwG
A1UdEwEB/wQCMAAwHQYDVIR0OBByEFPZEcIQcD/Bj2IFz/LEruo2ADJviMIGMBgNV
HSMEgYQwgYGAFO0/7kEh3FuEKS+Q/kYHaD/W6wihoWakZDBiMQswCQYDVQQGEwJV
UzELMAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJ
Q0FOTiBUTUNIMRswGQYDVQQDEXJJQ0FOTiBUTUNIIFRFU1QgQ0GCAQEwDgYDVIR0P
AQH/BAQDAgeAMC4GA1UdHwQnMCUwI6AhoB+GHWh0dHA6Ly9jcmwuaWNhbm4ub3Jn
L3RtY2guY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQ2qSy7ui+43cebKUKWPrzz9y/
IkrMeJGKjo40n+9uekaw3DJ5EqiOf/qZ4pjBD++oR6BJCb6NQuQKwnoAz5lE4Ssu
y5+i93oT3HfyVc4gNMioHm1PS1917DBKrbwbzAea/0jKWVzrvMv7TBfjxD3AQo1R
bU5dBr6IjbdLFlnO5x0G0mrG7x5OUPuurihyiURpFDpwh8KAHlwMcCpXGXFrtGKk
wydgyVYAty7otkl/z3bZkCVT34gPvF70sR6+QxUy8u0LzF5A/beYaZpxSYG31amL
```



```
AdXitTWfipaIGea9lEGFM0L9+Bg7XzNn4nVLXokyEB3bgS4scG6QznX23FGk
  </X509Certificate>
  </X509Data>
  </KeyInfo>
  </Signature>
</smd:signedMark>
```

NOTE: The example shown above includes white-spaces for indentation purposes. It is RECOMMENDED that SMDs do not include white-spaces between the XML elements, in order to mitigate risks of invalidating the digital signature when transferring of SMDs between applications takes place.

2.4. Encoded Signed Mark

The `<smd:encodedSignedMark>` element contains an encoded form of an SMD (described in Section 2.3), with the encoding defined by the "encoding" attribute with the default "encoding" value of "base64" [RFC4648].

The following is an example of a `<smd:encodedSignedMark>` element that uses the default "base64" for encoding a `<smd:signedMark>` element.

```
<smd:encodedSignedMark
  xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNtZDpzaWduZWRNYXJ
rIHhtbG5zOnNtZD0idXJuOmllldGY6cGFyYW1zOnhtbDpuczpzaWduZWRNYXJrLTEuMCIGA
W... (base64 data elided for brevity) ...
PC9zbWQ6c2lnbmVKTWFyaz4=
</smd:encodedSignedMark>
```

3. Formal Syntax

Two schemas are presented here. The first schema is the schema for the signed mark. The second schema is the schema for the mark.

The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

3.1. Signed Mark Schema

Copyright (c) 2016 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

BEGIN

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:ietf:params:xml:ns:signedMark-1.0"
  xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0"
  xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"
  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <annotation>
    <documentation>
      Schema for representing a Signed Trademark.
    </documentation>
  </annotation>

  <import namespace="urn:ietf:params:xml:ns:mark-1.0" />
  <import namespace="http://www.w3.org/2000/09/xmldsig#" />

  <!--
  Abstract signed mark for replacement via substitution.
  -->
  <element name="abstractSignedMark" type="smd:abstractSignedMarkType"
    abstract="true"/>

  <!--
  Empty type for use in extending for a signed mark
  -->
  <complexType name="abstractSignedMarkType"/>

  <element name="signedMark" type="smd:signedMarkType"
    substitutionGroup="smd:abstractSignedMark"/>

  <element name="encodedSignedMark" type="smd:encodedSignedMarkType"/>

  <complexType name="signedMarkType">
    <complexContent>
      <extension base="smd:abstractSignedMarkType">
        <sequence>
          <element name="id" type="mark:idType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

```

        <element name="issuerInfo" type="smd:issuerInfoType"/>
        <element name="notBefore" type="dateTime"/>
        <element name="notAfter" type="dateTime"/>
        <element ref="mark:abstractMark"/>
        <element ref="dsig:Signature"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
</extension>
</complexContent>
</complexType>

<complexType name="issuerInfoType">
    <sequence>
        <element name="org" type="token"/>
        <element name="email" type="mark:minTokenType"/>
        <element name="url" type="token" minOccurs="0"/>
        <element name="voice" type="mark:el64Type" minOccurs="0"/>
    </sequence>
    <attribute name="issuerID" type="token" use="required"/>
</complexType>

<complexType name="encodedSignedMarkType">
    <simpleContent>
        <extension base="token">
            <attribute name="encoding" type="token" default="base64"/>
        </extension>
    </simpleContent>
</complexType>
</schema>
END

```

3.2. Mark Schema

Copyright (c) 2016 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

```

BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:ietf:params:xml:ns:mark-1.0"
    xmlns:mark="urn:ietf:params:xml:ns:mark-1.0"

```

```
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<annotation>
  <documentation>
    Schema for representing a Trademark, also referred to
    as Mark.
  </documentation>
</annotation>

<!--
Abstract mark for replacement via substitution.
-->
<element name="abstractMark" type="mark:abstractMarkType"
  abstract="true"/>

<!--
<mark:mark> element definition
-->
<element name="mark" type="mark:markType"
  substitutionGroup="mark:abstractMark"/>

<!--
Empty type for use in extending for a mark
-->
<complexType name="abstractMarkType"/>

<!--
<mark:mark> child elements
-->
<complexType name="markType">
  <complexContent>
    <extension base="mark:abstractMarkType">
      <sequence>
        <element name="trademark" type="mark:trademarkType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="treatyOrStatute"
          type="mark:treatyOrStatuteType" minOccurs="0"
          maxOccurs="unbounded"/>
        <element name="court" type="mark:courtType" minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="holderType">
  <sequence>
```

```
<element name="name" type="token" minOccurs="0"/>
<element name="org" type="token" minOccurs="0"/>
<element name="addr" type="mark:addrType"/>
<element name="voice" type="mark:e164Type" minOccurs="0"/>
<element name="fax" type="mark:e164Type" minOccurs="0"/>
<element name="email" type="mark:minTokenType" minOccurs="0"/>
</sequence>
<attribute name="entitlement" type="mark:entitlementType"/>
</complexType>

<complexType name="contactType">
  <sequence>
    <element name="name" type="token"/>
    <element name="org" type="token" minOccurs="0"/>
    <element name="addr" type="mark:addrType"/>
    <element name="voice" type="mark:e164Type"/>
    <element name="fax" type="mark:e164Type" minOccurs="0"/>
    <element name="email" type="mark:minTokenType"/>
  </sequence>
  <attribute name="type" type="mark:contactTypeType"/>
</complexType>

<complexType name="trademarkType">
  <sequence>
    <element name="id" type="mark:idType"/>
    <element name="markName" type="token"/>
    <element name="holder" type="mark:holderType"
      maxOccurs="unbounded" />
    <element name="contact" type="mark:contactType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="jurisdiction" type="mark:ccType"/>
    <element name="class" type="integer" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="label" type="mark:labelType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="goodsAndServices" type="token" />
    <element name="apId" type="token" minOccurs="0"/>
    <element name="apDate" type="dateTime" minOccurs="0"/>
    <element name="regNum" type="token"/>
    <element name="regDate" type="dateTime"/>
    <element name="exDate" type="dateTime" minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="treatyOrStatuteType">
  <sequence>
    <element name="id" type="mark:idType"/>
    <element name="markName" type="token"/>
  </sequence>
</complexType>
```

```

    <element name="holder" type="mark:holderType"
      maxOccurs="unbounded" />
    <element name="contact" type="mark:contactType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="protection" type="mark:protectionType"
      maxOccurs="unbounded"/>
    <element name="label" type="mark:labelType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="goodsAndServices" type="token" />
    <element name="refNum" type="token"/>
    <element name="proDate" type="dateTime"/>
    <element name="title" type="token"/>
    <element name="execDate" type="dateTime"/>
  </sequence>
</complexType>

<complexType name="courtType">
  <sequence>
    <element name="id" type="mark:idType"/>
    <element name="markName" type="token"/>
    <element name="holder" type="mark:holderType"
      maxOccurs="unbounded" />
    <element name="contact" type="mark:contactType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="label" type="mark:labelType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="goodsAndServices" type="token" />
    <element name="refNum" type="token"/>
    <element name="proDate" type="dateTime"/>
    <element name="cc" type="mark:ccType"/>
    <element name="region" type="token" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="courtName" type="token"/>
  </sequence>
</complexType>

<!--
Address (<mark:addr>) child elements
-->
<complexType name="addrType">
  <sequence>
    <element name="street" type="token" minOccurs="1" maxOccurs="3"/>
    <element name="city" type="token"/>
    <element name="sp" type="token" minOccurs="0"/>
    <element name="pc" type="mark:pcType" minOccurs="0"/>
    <element name="cc" type="mark:ccType"/>
  </sequence>
</complexType>

```

```
<!--
<mark:protection> child elements
-->
<complexType name="protectionType">
  <sequence>
    <element name="cc" type="mark:ccType"/>
    <element name="region" type="token" minOccurs="0"/>
    <element name="ruling" type="mark:ccType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!--
Postal code definition
-->
<simpleType name="pcType">
  <restriction base="token">
    <maxLength value="16"/>
  </restriction>
</simpleType>

<!--
Country code definition
-->
<simpleType name="ccType">
  <restriction base="token">
    <length value="2"/>
  </restriction>
</simpleType>

<!--
Phone number with extension definition
-->
<complexType name="e164Type">
  <simpleContent>
    <extension base="mark:e164StringType">
      <attribute name="x" type="token"/>
    </extension>
  </simpleContent>
</complexType>

<!--
Phone number with extension definition
-->
<simpleType name="e164StringType">
  <restriction base="token">
    <pattern value="(\+[0-9]{1,3}\.[0-9]{1,14})?"/>
    <maxLength value="17"/>
  </restriction>
</simpleType>
```

```
</restriction>
</simpleType>

<!--
Id type definition
-->
<simpleType name="idType">
  <restriction base="token">
    <pattern value="\d+-\d+"/>
  </restriction>
</simpleType>

<!--
DNS label type definition
-->
<simpleType name="labelType">
  <restriction base="token">
    <minLength value="1"/>
    <maxLength value="63"/>
    <pattern value="[a-zA-Z0-9]([a-zA-Z0-9\-*[a-zA-Z0-9])?">
  </restriction>
</simpleType>

<!--
Type used for email addresses
-->
<simpleType name="minTokenType">
  <restriction base="token">
    <minLength value="1"/>
  </restriction>
</simpleType>

<simpleType name="entitlementType">
  <restriction base="token">
    <enumeration value="owner"/>
    <enumeration value="assignee"/>
    <enumeration value="licensee"/>
  </restriction>
</simpleType>

<simpleType name="contactTypeType">
  <restriction base="token">
    <enumeration value="owner"/>
    <enumeration value="agent"/>
    <enumeration value="thirdparty"/>
  </restriction>
</simpleType>
</schema>
```


END

4. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 6982 [RFC6982] before publication.

This section records the status of known implementations of the format defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982 [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982 [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

4.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-ietf-
eppext-tmch-smd.

Level of maturity: Production

Coverage: All aspects of the draft-ietf-eppext-tmch-smd are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

URL: http://www.verisigninc.com/en_US/channel-resources/domain-registry-products/epp-sdks

4.2. Verisign Consolidated Top Level Domain (CTLD) SRS

Organization: Verisign Inc.

Name: Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS)

Description: The Verisign Consolidated Top Level Domain (CTLD) Shared Registry System (SRS) implements the server-side of draft-ietf-eppept-tmch-smd for a variety of Top Level Domains (TLD's).

Level of maturity: Production

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppept-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark. Implements the encoding of the Mark in supporting the response of draft-ietf-eppept-launchphase.

Licensing: Proprietary

Contact: jgould@verisign.com

4.3. Verisign .COM / .NET SRS

Organization: Verisign Inc.

Name: Verisign .COM / .NET Shared Registry System (SRS)

Description: The Verisign Shared Registry System (SRS) for .COM, .NET and other IDN TLD's implements the server-side of draft-ietf-eppept-tmch-smd.

Level of maturity: Operational Test Environment (OTE)

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppept-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark.

Licensing: Proprietary

Contact: jgould@verisign.com

4.4. REngin v3.7

Organisation: Domain Name Services (Pty) Ltd

Name: REngin v3.7

Description: Server side implementation only

Level of maturity: Production

Coverage: All aspects of draft-ietf-epext-tmch-smd have been implemented

Licensing: Proprietary Licensing with Maintenance Contracts

Contact: info@dnservices.co.za

URL: <http://domain-name.services>

4.5. Uniregistry Corp. Shared Registry System (uSRS)

Organization: Uniregistry Corp.

Name: Uniregistry Corp. Shared Registry System (uSRS)

Description: Uniregistry's Shared Registry System implements the server-side of draft-ietf-epext-tmch-smd for its TLD registry.

Level of maturity: Production

Coverage: Implements parsing and validation of all aspects of draft-ietf-epext-tmch-smd including the Signed Mark, the Encoded Signed Mark, and the contained Mark. Implements the encoding of the Mark in supporting the response of draft-ietf-epext-launchphase.

Licensing: Proprietary

Contact: fobispo@uniregistry.link

5. Acknowledgements

Special thanks to Chris Wright for creating the first prototype of a SMD; James Gould, Wil Tan and Gavin Brown for creating the mark and SMD definitions in their EPP draft launch extension on which this draft is based. Portions of the security section were shamefully copied from RFC5105. The author would like to acknowledge the following individuals for their contributions to this document: Scott Hollenbeck and Jan Jansen.

6. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. Two URI assignments have been registered by the IANA.

Registration request for the signed mark namespace:

URI: urn:ietf:params:xml:ns:signedMark-1.0

Registrant Contact: IESG

XML: None. Namespace URIs do not represent an XML specification.

Registration request for the signed mark schema:

URI: urn:ietf:params:xml:schema:signedMark-1.0

Registrant Contact: IESG

XML: See the "Formal Syntax" section of this document.

Registration request for the mark namespace:

URI: urn:ietf:params:xml:ns:mark-1.0

Registrant Contact: IESG

XML: None. Namespace URIs do not represent an XML specification.

Registration request for the mark schema:

URI: urn:ietf:params:xml:schema:mark-1.0

Registrant Contact: IESG

XML: See the "Formal Syntax" section of this document.

7. Security Considerations

The security of a Signed Mark object depends on the security of the underlying XML DSIG algorithms. As such, all the security considerations from [XMLDSIG] apply here as well.

The digital signature algorithm used in Signed Mark objects SHOULD be RSA-SHA256 [RFC4051]. The size of the RSA key SHOULD be at least 2048 bits. A valid reason for choosing something else would be if RSA-SHA256 would be deemed to not provide sufficient security.

In the case of the ICANN Trademark Clearinghouse (TMCH), Signed Mark objects use the algorithms for digesting and signing recommended in this document.

Signed Marks are used primarily for sunrise domain name registrations in gTLDs, but other third parties might be using them. A party using Signed Marks should verify that the digital signature is valid based on local policy. In the case of gTLDs, the RPM Requirements document [ICANN-TMCH] defines such policy, and the PKI is defined in [I-D.ietf-eppext-tmch-func-spec]. Implementations will need to implement such a PKI (or an equivalent) in order for the signatures defined in this document to have any useful semantics.

8. References

8.1. Normative References

[ICANN-TMCH]

ICANN, "ICANN Trademark Clearinghouse, Rights Protection Mechanism Requirements", 2013,
<<http://newgtlds.icann.org/en/about/trademark-clearinghouse/rpm-requirements-30sep13-en.pdf>>.

[ISO3166-2]

ISO, "International Standard for country codes and codes for their subdivisions", 2006,
<http://www.iso.org/iso/home/standards/country_codes.htm>.

[ITU.E164.2005]

International Telecommunication Union, "The international public telecommunication numbering plan", 2010,
<<https://www.itu.int/rec/T-REC-E.164-201011-I/en>>.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.

[RFC3688]

Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
DOI 10.17487/RFC3688, January 2004,
<<http://www.rfc-editor.org/info/rfc3688>>.

[RFC4051]

Eastlake 3rd, D., "Additional XML Security Uniform Resource Identifiers (URIs)", RFC 4051,
DOI 10.17487/RFC4051, April 2005,
<<http://www.rfc-editor.org/info/rfc4051>>.

[RFC4648]

Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,
<<http://www.rfc-editor.org/info/rfc4648>>.

- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<http://www.rfc-editor.org/info/rfc5322>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<http://www.rfc-editor.org/info/rfc5890>>.
- [WIPO-NICE-CLASSES] WIPO, "WIPO Nice Classification", 2015, <<http://www.wipo.int/classifications/nice/en>>.
- [WIPO.ST3] WIPO, "Recommended standard on two-letter codes for the representation of states, other entities and intergovernmental organizations", March 2007, <<http://www.wipo.int/standards/en/pdf/03-03-01.pdf>>.
- [XMLC14N] W3C Recommendation, "Exclusive XML Canonicalization Version 1.0", 2002, <<http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718>>.
- [XMLDSIG] W3C Recommendation, "XML Signature Syntax and Processing (Second Edition)", 2013, <<http://www.w3.org/TR/xmlsig-core1>>.

8.2. Informative References

- [I-D.ietf-eppext-tmch-func-spec] Lozano, G., "TMCH functional specifications", draft-ietf-eppext-tmch-func-spec-00 (work in progress), October 2015.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.

Author's Address

Gustavo Lozano
ICANN
12025 Waterfront Drive, Suite 300
Los Angeles 90292
US

Phone: +1.3103015800
Email: gustavo.lozano@icann.org

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 15, 2016

N. Kong
J. Yao, Ed.
X. Li
CNNIC
J. Xie
CONAC
W. Tan
Cloud Registry
October 13, 2015

Extensible Provisioning Protocol (EPP) Domain Name Mapping Extension for
Bundling Registration
draft-kong-epext-bundling-registration-02

Abstract

This document describes an extension of Extensible Provisioning Protocol (EPP) domain name mapping for the provisioning and management of bundling registration of domain names. Specified in XML, this mapping extends the EPP domain name mapping to provide additional features required for the provisioning of bundled domain names.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 15, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Definitions	4
4. Overview	4
5. Requirement for Bundling Registration of Names	5
6. Object Attributes	6
6.1. RDN	6
6.2. BDN	6
7. EPP Command Mapping	6
7.1. EPP Query Commands	7
7.1.1. EPP <check> Command	7
7.1.2. EPP <info> Command	8
7.1.3. EPP <transfer> Query Command	9
7.2. EPP Transform Commands	9
7.2.1. EPP <create> Command	10
7.2.2. EPP <delete> Command	12
7.2.3. EPP <renew> Command	13
7.2.4. EPP <transfer> Command	14
7.2.5. EPP <update> Command	14
8. Formal Syntax	14
9. Internationalization Considerations	16
10. IANA Considerations	16
11. Security Considerations	17
12. Implementation Status	17
13. Acknowledgements	17
14. Change History	18

14.1.	draft-kong-epp-bundle-mapping: Version 00	18
14.2.	draft-kong-epp-bundle-mapping: Version 01	18
14.3.	draft-kong-epp-bundle-mapping: Version 02	18
15.	References	18
15.1.	Normative References	18
15.2.	Informative References	19
	Authors' Addresses	20

1. Introduction

Bundled domain names are those who share the same TLD but whose second level labels are variants, or those who has identical second level labels for which certain parameters are shared in different TLDs. For example, Public Interest Registry, request to implement technical bundling of second level domains for .NGO and .ONG. So we have two kinds of bundled domain names. First one is in the form of "V-label.TLD" in which the second level labels (V-label) are variants sharing the same TLD; Second one is in the form of "LABEL.V-tld" in which the second level labels(LABEL) are same with the different TLDs (V-tld);

For the name variants, some registries adopt the policy that variant IDNs which are identified as equivalent are allocated or delegated to the same registrant. For example, the specified registration policy of Chinese Domain Name (CDN) is that a registrant can apply an original CDN in any forms: Simplified Chinese (SC) form, Traditional Chinese (TC) form, or other variant forms, then the corresponding variant CDN in SC form and that in TC form will also be delegated to the same registrant. All variant names in the same TLD contain same attributes.

The basic Extensible Provisioning Protocol (EPP) domain name mapping [RFC5731] provides the domain name registration one by one. It does not specify how to register the bindled names which share the same attributes.

In order to meet above requirements of the bundled names registration, this document describes an extension of the EPP domain name mapping [RFC5731] for the provisioning and management of bundled names. This document is specified using the Extensible Markup Language (XML) 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

The EPP core protocol specification [RFC5730] provides a complete description of EPP command and response structures. A thorough understanding of the base protocol specification is necessary to understand the extension of mapping described in this document.

This document uses lots of the concepts of the IDN, so a thorough understanding of the IDNs for Application (IDNA, described in [RFC5890], [RFC5891], and [RFC5892]) and a thorough understanding of variant approach discussed in [RFC4290] are both required.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

uLabel is defined in [RFC 5890]. uLabel is expressed in this document as a number of characters with the format of U+XXXX where XXXX is a UNICODE point.

"b-dn-1.0" in this document is used as an abbreviation for urn:ietf:params:xml:ns:b-dn-1.0.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and white space in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this specification.

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented to develop a conforming implementation.

3. Definitions

The following definitions are used in this document:

- o Registered Domain Name (RDN), represents the valid domain name that users submitted for registration by the first time.
- o Bundled Domain Name (BDN), represents the bundled domain name produced according to the bundled domain name registration policy.

4. Overview

Domain registries have traditionally adopted a registration model whereby metadata relating to a domain name, such as its expiration date and sponsoring registrar, are stored as properties of the domain object. The domain object is then considered an atomic unit of registration, on which operations such as update, renewal and deletion may be performed.

Bundled names, brought about the need for multiple domain names to be registered and managed as a single package. In this model, the

registry typically accepts a domain registration request (i.e. EPP domain <create> command) containing the domain name to be registered. This domain name is referred to as the RDN in this document. As part of the processing of the registration request, the registry generates a set of bundled names that are related to the RDN, either programmatically or with the guidance of registration policies, and place them in the registration package together with the RDN.

The bundled names have the same properties, such as expiration date and sponsoring registrar, by sharing one domain object. So when users update any property of a domain object within a bundle package, that property of all other domain objects in the bundle package will be updated at the same time.

5. Requirement for Bundling Registration of Names

The bundled names whether they are in the form of "V-label.TLD" or in the form of "LABEL.V-tld" should share some parameter or attributes associated with domain names. Typically, Bundled names will share the following parameters or attributes:

- o Registrar Ownership
- o Registration and Expiry Dates
- o Registrant, Admin, Billing, and Technical Contacts
- o Name Server Association
- o Domain Status
- o Applicable grace periods (Add Grace Period, Renewal Grace Period, Auto-Renewal Grace Period, Transfer Grace Period, and Redemption Grace Period)

Because the domain names are bundled and share the same parameters or attributes, the EPP command should do some processing for these requirements:

- o When performing a domain check, either BDN or RDN can be queried for the EPP command, and will return the same response.
- o When performing a domain info, either BDN or RDN can be queried, the same response will include both BDN and RDN information with the same attributes.
- o When performing a domain Create, either BDN or RDN will be accepted. If the domain name is available, both BDN and RDN will be registered.
- o When performing a domain Delete, either BDN or RDN will be accepted. If the domain name is available, both BDN and RDN will be deleted.
- o When performing a domain renew, either BDN or RDN will be accepted. Upon a successful domain renewal, both BDN and RDN will have their expiry date extended by the requested term. Upon a successful domain renewal, both BDN and RDN will conform to the same renew grace period.

- o When performing a domain transfer, either BDN or RDN will be accepted. Upon successful completion of a domain transfer request, both BDN and RDN will enter a pendingTransfer status. Upon approval of the transfer request, both BDN and RDN will be owned and managed by the same new registrant.
- o When performing a domain update, either BDN or RDN will be accepted. Any modifications to contact associations, name server associations, domain status values and authorization information will be applied to both BDN and RDN.

6. Object Attributes

This extension defines following additional elements to the EPP domain name mapping [RFC5731]. All of these additional elements can be got from <domain:info> command.

6.1. RDN

The RDN is an ASCII name or an IDN with the A-label [RFC5890] form. In this document, its corresponding element is <b-dn:rdn>. An optional attribute "uLabel" associated with <b-dn:rdn> is used to represent the U-label [RFC5890] form. An optional boolean "activated" attribute, with a default true value, is used to indicate the presence of the label in the zone file.

For example: <b-dn:rdn uLabel="U+5B9E"U+4F8B".example> xn--fsq270a.example</b-dn:rdn>

6.2. BDN

The BDN is an ASCII name or an IDN with the A-label [RFC5890] form which is converted from the corresponding BDN. In this document, its corresponding element is <b-dn:bdn>. An optional attribute "uLabel" associated with <b-dn:bdn> is used to represent the U-label [RFC5890] form.

For example: <b-dn:bdn uLabel="U+5BE6"U+4F8B".example> xn--fsqz41a.example</b-dn:bdn>

7. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in the EPP core protocol specification [RFC5730]. The command mappings described here are specifically for use in provisioning and managing bundled names via EPP.

7.1. EPP Query Commands

EPP provides three commands to retrieve domain information: <check> to determine if a domain object can be provisioned within a repository, <info> to retrieve detailed information associated with a domain object, and <transfer> to retrieve domain-object transfer status information.

7.1.1. EPP <check> Command

This extension does not add any element to the EPP <check> command or <check> response described in the EPP domain name mapping [RFC5731]. However, when either RDN or BDN is sent for check, response SHOULD contain both RDN and BDN information, which may also give some explanation in the reason field to tell the user that the associated domain name is a produced name according to some bundle domain name policy.

Example <check> Response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:chkData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:      <domain:cd>
S:        <domain:name avail="1">
S:          xn--fsq270a.example</domain:name>
S:      </domain:cd>
S:      <domain:cd>
S:        <domain:name avail="1">
S:          xn--fsqz41a.example</domain:name>
S:      <domain:reason>This associated domain name is
S:        a produced name
S:        based on bundle name policy.</domain:reason>
S:      </domain:cd>
S:    </domain:chkData>
S:  </resData>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54322-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

7.1.2. EPP <info> Command

This extension does not add any element to the EPP <info> command described in the EPP domain mapping [RFC5731]. However, additional elements are defined for the <info> response.

When an <info> command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:infData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:infData> element contains the <b-dn:bundle> which has the following child elements:

- o An <b-dn:rdn> element that contains the RDN, along with the attributes described below.
- o An OPTIONAL <b-dn:bdn> element that contains the BDN, along with the attributes described below.

The above elements contain the following attributes:

- o An optional "uLabel" attribute represents the U-label of the element.

Example <info> Response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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>xn--fsq270a.example</domain:name>
S:        <domain:roid>58812678-domain</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>123</domain:registrant>
S:        <domain:contact type="admin">123</domain:contact>
S:        <domain:contact type="tech">123</domain:contact>
S:        <domain:ns>
S:        <domain:hostObj>ns1.example.cn
S:      </domain:hostObj>
S:    </domain:ns>
S:    <domain:clID>ClientX</domain:clID>
```

```

S:      <domain:crID>ClientY</domain:crID>
S:      <domain:crDate>2011-04-03T22:00:00.0Z
        </domain:crDate>
S:      <domain:exDate>2012-04-03T22:00:00.0Z
        </domain:exDate>
S:      <domain:authInfo>
S:      <domain:pw>2fooBAR</domain:pw>
S:      </domain:authInfo>
S:      </domain:infData>
S:      </resData>
S:      <extension>
S:      <b-dn:infData
S:      xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
S:      <b-dn:bundle>
S:      <b-dn:rdn uLabel="U+5B9E"U+4F8B".example
S:      >xn--fsq270a.example</b-dn:rdn>
S:      <b-dn:bdn uLabel="U+5BE6"U+4F8B".example
S:      >xn--fsqz41a.example</b-dn:bdn>
S:      </b-dn:bundle>
S:      </b-dn:infData>
S:      </extension>
S:      <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:      </trID>
S:      </response>
S:</epp>

```

<info> Response for the unauthorized client has not been changed, see [RFC5731] for detail.

An EPP error response MUST be returned if an <info> command cannot be processed for any reason.

7.1.3. EPP <transfer> Query Command

This extension does not add any element to the EPP <transfer> command or <transfer> response described in the EPP domain mapping [RFC5731].

7.2. EPP Transform Commands

EPP provides five commands to transform domain objects: <create> to create an instance of a domain object, <delete> to delete an instance of a domain object, <renew> to extend the validity period of a domain object, <transfer> to manage domain object sponsorship changes, and <update> to change information associated with a domain object.

When these commands have been processed successfully, the EPP `<resData>` element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP `<extension>` element SHOULD contain the `<b-dn:bundle>` which has the following child elements:

- o An `<b-dn:rdn>` element that contains the RDN, along with the attributes described below.
- o An OPTIONAL `<b-dn:bdn>` element that contains the BDN, along with the attributes described below.

The above elements contain the following attribute:

- o An optional "uLabel" attribute represents the U-label of the element.

7.2.1. EPP `<create>` Command

This extension defines additional elements to extend the EPP `<create>` command described in the EPP domain name mapping [RFC5731] for bundled names registration.

In addition to the EPP command elements described in the EPP domain mapping [RFC5731], the `<create>` command SHALL contain an `<extension>` element. The `<extension>` element SHOULD contain a child `<b-dn:create>` element that identifies the bundle namespace and the location of the bundle name schema.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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>xn--fsq270a.example</domain:name>
C:        <domain:period unit="y">2</domain:period>
C:        <domain:registrant>123</domain:registrant>
C:        <domain:contact type="admin">123</domain:contact>
C:        <domain:contact type="tech">123</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <b-dn:create
C:        xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
C:        <b-dn:rdn uLabel="U+5B9E"U+4F8B".example>
C:          xn--fsq270a.example</b-dn:rdn>
C:        </b-dn:create>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When an <create> command has been processed successfully, the EPP <creData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:creData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:creData> element contains the <b-dn:bundle> element.

Example <create> Response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:creData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--fsq270a.example</domain:name>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:creData>
S:    </resData>
S:    <extension>
S:      <b-dn:creData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="U+5B9E"U+4F8B".example
S:            >xn--fsq270a.example</b-dn:rdn>
S:          <b-dn:bdn uLabel="U+5BE6"U+4F8B".example
S:            >xn--fsqz41a.example</b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:creData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

<create> Response for the unauthorized client has not been changed, see [RFC5731] for detail.

An EPP error response MUST be returned if an <create> command cannot be processed for any reason.

7.2.2. EPP <delete> Command

This extension does not add any element to the EPP <delete> command described in the EPP domain mapping [RFC5731]. However, additional elements are defined for the <delete> response.

When a <delete> command has been processed successfully, the EPP <delData> element MUST contain child elements as described in the EPP

domain mapping [RFC5731]. In addition, the EPP <extension> element SHOULD contain a child <b-dn:delData> element that identifies the extension namespace if the domain object has data associated with this extension and based on its service policy. The <b-dn:delData> element SHOULD contain the <b-dn:bundle> element.

Example <delete> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
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:      <b-dn:delData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="U+5B9E"U+4F8B".example>xn--fsq270a.ex
ample</b-dn:rdn>
S:          <b-dn:bdn uLabel="U+5BE6"U+4F8B".example>xn--fsq41a.ex
ample</b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:delData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

An EPP error response MUST be returned if a <delete> command cannot be processed for any reason.

7.2.3. EPP <renew> Command

This extension does not add any element to the EPP <renew> command described in the EPP domain name mapping [RFC5731]. However, when either RDN or BDN is sent for renew, response SHOULD contain both RDN and BDN information. When the command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:renData> which contains <b-dn:bundle> element.

7.2.4. EPP <transfer> Command

This extension does not add any element to the EPP <transfer> command described in the EPP domain name mapping [RFC5731]. When the command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:trnData> which contains <b-dn:bundle> element.

7.2.5. EPP <update> Command

This extension does not add any element to the EPP <update> command described in the EPP domain name mapping [RFC5731]. When the command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. This EPP <extension> element SHOULD contain the <b-dn:upData> which contains <b-dn:bundle> element.

8. Formal Syntax

An EPP object name mapping extension for bundled names is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of the object mapping suitable for automated validation of EPP XML instances. The BEGIN and END tags are not part of the schema; they are used to note the beginning and ending of the schema for URI registration purposes.

BEGIN

```
<?xml version="1.0"      encoding="UTF-8"?>

  <schema targetNamespace="urn:ietf:params:xml:ns:b-dn-1.0"
    xmlns:b-dn="urn:ietf:params:xml:ns:b-dn-1.0"
    xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
    xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

    <!--
    Import common element types.
    -->
    <import namespace="urn:iana:xml:ns:eppcom-1.0"
      schemaLocation="eppcom-1.0.xsd"/>
    <import namespace="urn:iana:xml:ns:epp-1.0"
      schemaLocation="epp-1.0.xsd"/>
    <annotation>
      <documentation>
        Extensible Provisioning Protocol v1.0
        Bundle Domain Extension Schema v1.0
```

```
</documentation>
</annotation>

<!--
Child elements found in EPP      commands.
-->
<element name="create" type="b-dn:createDataType"/>

<!--
Child elements of the <b-dn:create>      command
All      elements must be present at      time of creation
-->
<complexType name="createDataType">
  <sequence>
    <element name="rdn" type="b-dn:rdnType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

<!--
Child elements of the <b-dn:update>      command
All      elements must be present at      time of creation
-->

<!--
Child elements found in EPP      commands.
-->
<element name="infData" type="b-dn:trnDataType"/>
<element name="delData" type="b-dn:trnDataType"/>
<element name="creData" type="b-dn:trnDataType"/>
<element name="renData" type="b-dn:trnDataType"/>
<element name="trnData" type="b-dn:trnDataType"/>
<element name="upData" type="b-dn:trnDataType"/>

<complexType name="trnDataType">
  <sequence>
    <element name="bundle" type="b-dn:bundleType" />
  </sequence>
</complexType>

<!--
<transfer> response      elements.
All      elements must be present at      time of poll query
-->
<complexType name="bundleType">
  <sequence>
    <element name="rdn" type="b-dn:rdnType" />
```

```
        <element name="bdn" type="b-dn:rdnType"
                  minOccurs="0"   maxOccurs="unbounded" />

    </sequence>
</complexType>

<complexType name="rdnType">
  <simpleContent>
    <extension base="eppcom:labelType">
      <attribute name="uLabel" type="eppcom:labelType"/>
    </extension>
  </simpleContent>
</complexType>

<!--
End      of schema.
-->
</schema>
```

END

9. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 and UTF-16. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED.

As an extension of the EPP domain name mapping, the elements, element content described in this document MUST inherit the internationalization conventions used to represent higher-layer domain and core protocol structures present in an XML instance that includes this extension.

10. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. IANA is requested to assign the following two URIs.

Registration request for the IDN namespace:

- o URI: urn:ietf:params:xml:ns:b-dn-1.0

- o Registrant Contact: See the "Author's Address" section of this document.
- o XML: None. Namespace URI does not represent an XML specification.

Registration request for the IDN XML schema:

- o URI: urn:ietf:params:xml:schema:b-dn-1.0
- o Registrant Contact: See the "Author's Address" section of this document.
- o XML: See the "Formal Syntax" section of this document.

11. Security Considerations

The object mapping extension described in this document does not provide any other security services or introduce any additional considerations beyond those described by [RFC5730] or those caused by the protocol layers used by EPP.

12. Implementation Status

Note to RFC Editor: Please remove this section before publication.

- o CNNIC has implemented this extension in his EPP based Chinese domain name registration system.
- o Public Interest Registry, has requested to implement technical bundling of second level domains for .NGO and .ONG. This means that by registering and purchasing a domain in the .ngo TLD, for example, the NGO registrant is also registering and purchasing the corresponding name in the .ong TLD (and vice-versa for registrations in .ong).

13. Acknowledgements

The authors especially thank the authors of [RFC5730] and [RFC5731] and the following ones of CNNIC: Weiping Yang, Chao Qi. This draft extends the draft draft-kong-epp-idn-variants-mapping to support both forms of bundled names: V-label.TLD and LABEL.V-tld.

Useful comments were made by John Klensin, Scott Hollenbeck, Patrick Mevzek and Edward Lewis.

14. Change History

RFC Editor: Please remove this section.

14.1. draft-kong-epp-bundle-mapping: Version 00

- o EPP extensiton for bundled domain name registrations.

14.2. draft-kong-epp-bundle-mapping: Version 01

- o Change the proposed category from EXP to STD.
- o Add the section of Implementation Status.
- o Refine the text, and update the examples.

14.3. draft-kong-epp-bundle-mapping: Version 02

- o Refine the texts.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<http://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<http://www.rfc-editor.org/info/rfc5731>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<http://www.rfc-editor.org/info/rfc5890>>.

- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <<http://www.rfc-editor.org/info/rfc5891>>.
- [RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <<http://www.rfc-editor.org/info/rfc5892>>.
- [W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204", February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.
- [W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.
- [W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028", October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

15.2. Informative References

- [bundle.name]
ICANN, "Registry Services Technical Evaluation Panel (RSTEP) Report on Public Interest Registry's Request to Implement Technical Bundling in .NGO and .ONG", July 2014, <<https://www.icann.org/public-comments/rstep-technical-bundling-2014-07-29-en>>.
- [Final.Integrated.Issues.Report]
ICANN, "The IDN Variant Issues Project: A Study of Issues Related to the Management of IDN Variant TLDs", February 2012, <<http://www.icann.org/en/topics/idn/idn-vip-integrated-issues-final-clean-20feb12-en.pdf>>.

[RFC4290] Klensin, J., "Suggested Practices for Registration of Internationalized Domain Names (IDN)", RFC 4290, DOI 10.17487/RFC4290, December 2005, <<http://www.rfc-editor.org/info/rfc4290>>.

Authors' Addresses

Ning Kong
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3147
Email: nkong@cnnic.cn

Jiankang Yao (editor)
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3007
Email: yaojk@cnnic.cn

Xiaodong Li
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3020
Email: xl@cnnic.cn

Jiagui Xie
CONAC
Jia 31, North Guangximen, Xibahe, Chaoyang District
Beijing, Beijing 100028
China

Phone: +86 10 10 5203 5025
Email: xieljg@conac.cn

Wil Tan
Cloud Registry
Suite 32 Seabridge House, 377 Kent St
Sydney, NSW 2000
Australia

Phone: +61 414 710899
Email: wil@cloudregistry.net