Internationalized Domain Name Mapping Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-eppext-idnmap-02

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension mapping for the provisioning of Internationalized Domain Names (IDN) stored in a shared central repository. This mapping extends the EPP domain name mapping to provide additional features required to implement registrations of domain names in characters sets other than ASCII.

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 July 9, 2015.

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

This document is written in consideration with the Guidelines for Extending the Extensible Provisioning Protocol as defined in [RFC3735].

To comply with the Guidelines for the Implementation of Internationalized Domain Names [1], it is required to associate each label to be registered with a single script, as defined by the code division of the Unicode code chart. This requirement imposes a challenge for registries using the EPP protocol, since there is no such field currently in the domain name mapping to allow for this information to be exchanged.

In addition, registries intending to comply with the recommendation of section 4.1 [RFC5891] of the IDNA2008 protocol, which implies the verification of both the name in ASCII Compatible Encoding and Unicode form, will be able to do so using this extension.

This extension adds two additional data element to the EPP Domain Name mapping, to allow for association of a domain name to an IDN
table identifier, and a the domain name in Unicode Normalization Form C (NFC [2]).

2. 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 [RFC2119].

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

"idn-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:idn-1.0". The XML namespace prefix "idn" 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.

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in [RFC5730].

3.1. EPP Query Commands

This extension does not add any elements to the EPP <check>, <poll>, or <transfer> commands or responses.

3.1.1. EPP <info> Command

This extension does not add any elements to the EPP <info> command, but does include elements in the response, when the extension has been selected during a <login> command.

Example <info> command:
When the info command has been processed successfully, and the domain name is an IDN, the server must include in the <extension> section of the EPP response an <idn:data> element with the following elements:

- A <idn:table> element that contains the IDN table identifier.
- A <idn:uname> element that contains the domain name in Unicode NFC form.

Example <info> response for an authorized client:
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--espaol-zwa.example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:hostObj>ns1.example.com</domain:hostObj>
S:        <domain:hostObj>ns1.example.net</domain:hostObj>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:          <domain:hostObj>ns1.example.net</domain:hostObj>
S:        </domain:ns>
S:        <domain:c1ID>ClientX</domain:c1ID>
S:        <domain:c1ID>ClientY</domain:c1ID>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upDate>ClientX</domain:upID>
S:        <domain:upDate>1999-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <idn:data xmlns:idn="urn:ietf:params:xml:ns:idn-1.0">
S:        <idn:table>es</idn:table>
S:        <idn:uname>espa#xF1;ol.example.com</idn:uname>
S:      </idn:data>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
3.2. EPP Transform Commands

This extension does not add any elements to the EPP <delete>, <renew>, or <transfer> commands or responses.

3.2.1. EPP <create> Command

This extension defines additional elements for the EPP <create> command.

If the domain name is an IDN, the EPP command MUST contain an <extension> element, which MUST contain a child <idn:data> element with the following child elements:

- A <idn:table> element that contains the IDN table identifier as provided by the server.

- An optional <idn:uname> element that contains the domain name to be registered in Unicode NFC.

Example <create> command:
The server MUST validate the name using the procedure described in section 4.2 of [RFC5891].

If the validation of the IDN name failed because it contained a code point not available in the specified IDN table, the server MUST return an EPP error 2306.

In the specific case that the <domain:name> provided did not map to the provided <idn:uname>, the server MUST respond with an EPP error 2005.

3.3. Formal Syntax

An EPP object mapping 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.
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
   xmlns:idn="urn:ietf:params:xml:ns:idn-1.0"
   targetNamespace="urn:ietf:params:xml:ns:idn-1.0"
   elementFormDefault="qualified">
  <documentation>
    Extensible Provisioning Protocol v1.0 domain name extension
    schema for IDN Table selection.
  </documentation>
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
   schemaLocation="eppcom-1.0.xsd"/>
  <!-- Child elements found in IDN -->
  <element name="data" type="idn:idnDataType"/>
  <complexType name="idnDataType">
    <sequence>
      <element name="table" type="eppcom:minTokenType"/>
      <element name="uname" type="eppcom:labelType"
        minOccurs="0"/>
    </sequence>
  </complexType>
  <!-- End of schema. -->
</schema>

4. 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 contact namespace:

URI: urn:ietf:params:xml:ns:idn-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 contact XML schema:

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

Registrant Contact: See the "Author’s Address" section of this
document.
5. 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.

6. References

6.1. Normative References


6.2. Informational References


6.3. URIs


Authors' Addresses

Francisco Obispo
Uniregistry Corp.
3-110 Governors Square
Grand Cayman, Grand Cayman KY1-1108
KY

Phone: +194990334499
Email: fobispo@uniregistry.com
URI: http://www.uniregistry.com/
Luis Enrique Munoz  
Uniregistry Corp.  
3-110 Governors Square  
Grand Cayman, Grand Cayman  KY1-1108  
KY  

Phone: +19499034226  
Email: fobispo@uniregistry.com  
URI: http://www.uniregistry.com/
Abstract

This document describes a generic Extensible Provisioning Protocol (EPP) extension for the purpose of relaying data between registrars.

Furthermore, this document describes a specific implementation for relaying DNSSEC key material between DNS operators (by means of their respective registrars), to facilitate the change of DNS operator, while keeping the DNSSEC chain of trust intact.

This I-D introduces a new generic command <relay> and an element <relayData>. For the specific implementation of relaying DNSSEC key material it introduces an extension of the <relayData> with a <keyRelayData> element.

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 July 12, 2015.
1. Introduction

There are certain transactions in the lifecycle of a domain name, that require interaction between registrars but need registration data from the registry. Since all registrars involved have a secure channel to the registry for maintaining the delegation, the registry can act as relay for such data to transfer securely and authoritative between the registrars involved.

Currently these transactions aren’t supported in the Extensible Provisioning Protocol (EPP) [RFC5730]. One example of such a transaction is the exchange of DNSSEC key material to keep the DNSSEC chain of trust intact in case of a change of DNS-operator.

In this document we will define:

- A protocol extension that implements the relaying of data between registrars through the existing authenticated EPP channel. This protocol extension introduces a new EPP command called <relay> with an element <relayData>.

- An extension to the <relayData> element called <keyRelayData> that can be used for the relaying DNSSEC key material using the <relay> command.

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. Relaying Data

The <relay> command uses the existing authenticated EPP channel between the registrar and the registry. Registrars can use this secure channel for relaying data to other registrars. The registry serves as an intermediary between two registrars (see Figure 1).
The `<relay>` command uploads data from a registrar X to the registry. The uploaded data is then pushed onto the message queue of registrar Y by the registry based on the information within the `<relayData>` element of the `<relay>` command and the registration data maintained by the registry.

The data to be relayed MUST relate to registration data of the registry. The `<relay>` command is not intended to relay data that has no relationship to registration data. We have e-mail for that.

If for some reason the registry cannot process the `<relay>` command, an EPP error response MUST be returned. If the registry does process the `<relay>` command it MUST put all elements of `<relayData>` on to the message queue of registrar Y.

1.2.1. Rationale For a New Command

This new `<relay>` command can be best described as a "transient command" as it only facilitates communication of data between two registrars without changing the registration data at the registry. No existing EPP command can be (re)used for this function. This extension of EPP is in accordance to [RFC3735].

1.2.2. Extending `<relayData>` per use case

One MUST extend the `<relayData>` element per use case to define the data to be relayed. In the extension, one MUST make provisions for the registry how to determine the receiving registrar of the `<relay>` command.

1.3. 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 (through the gaining registrar) DNSSEC key material before the registration data can be changed.

```
+--------------------+  DNSKEY   +---------------------+
| gaining DNS operator| ˜˜˜˜˜˜˜˜> | losing DNS operator |
+--------------------+           +---------------------+
|                   ^|
V                   |
+--------------------+         +---------------------+
| gaining registrar |         | registrar of record |
+--------------------+         +---------------------+
| EPP relay          |         | EPP poll             |
V                   |
+-----------------------------+
|           registry          |
+-----------------------------+
```

Figure 2: Transfer of DNSSEC key material.

As the `<relay>` command uses a secure channel, it can be used as a method for exchanging this DNSSEC key material securely (see Figure 2).

The gaining and losing DNS operators could talk directly to each other (the `˜` arrow) 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.

This I-D contains an extension of the `<relayData>` element for this use case.

2. Object Attributes

2.1. DNSSEC Key Material

To transfer DNSSEC key material with the `<relay>` command the generic `<relayData>` is extended with a `<keyRelayData>` element that contains the data for relaying the key material. See Section 1.2.2.

This `<keyRelayData>` element REQUIRES a minimum of three child elements:

- A `<name>` element which contains the domain name for which we upload the key. The registry MUST relay the `<keyRelayData>` to the registrar of record of the provided domain name.
One or more <keyData> elements that contains the DNSSEC key material as described in [RFC5910], Section 4.2.

An <authInfo> element that contains an authorization token ([RFC5731], Section 3.2.1). This indicates that the registrar has authorization from the registrant to change the zone data, and a possible future transfer is authorized. The registry MAY check if the <authInfo> data is correct and if it does, it MUST return an EPP error response if the authorization token is not correct.

And an OPTIONAL <expiry> child element.

An <expiry> element that describes the expected lifetime of the relayed key(s) in the zone. The losing DNS operator can use this as an indication when to safely remove the inserted key material from the zone. This may be because the transaction that needed the insertion is either completed or has been abandoned if not completed before this expire time. The <expiry> element MUST contain one of the following child elements:

* <absolute/>: The policy is valid from the current date and time until it expires on the specified date and time.

* <relative/>: The policy is valid from the current date and time until the end of the specified duration.

3. EPP Command Mapping

3.1. EPP Transient Commands

3.1.1. EPP <relay> Command

The EPP <relay> command is a generic EPP command used for relaying data between registrars. It contains the data to be relayed and the client transaction identifier. It has been designed to be extensible for usage in other use-cases.

The <relay> command REQUIRES the following child elements:

* One or more <relayData> elements containing data to be relayed.

* An OPTIONAL <clTRID> (client transaction identifier) element that MAY be used to uniquely identify the command to the registrar. See [RFC5730], Section 2.5.

Example <relay> command:
3.2. EPP Query Commands

This EPP extension does not change any command other than the EPP <poll> command response.

3.2.1. EPP <poll> command

This extension adds elements to the response to a <poll> command with the "op" attribute set to "req". Specifically, a <panData> element is added to the <resData> section of the service message, containing the following elements:

- A REQUIRED <relayData> element that contains the relayed data.
- A REQUIRED <paDate> element that contains the date and time of the submitted <relay> command.
- A REQUIRED <reID> element that contains the identifier of the registrar that requested the data relay.

- A REQUIRED <acID> element that contains the identifier of the registrar that SHOULD act upon the data 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:secDNS="urn:ietf:params:xml:ns:secDNS-1.1"
S:     xmlns:domain="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>Relay action completed successfully.</msg>
S:         </msgQ>
S:         <resData>
S:             <r:panData xmlns:r="urn:ietf:params:xml:ns:relay-1.0">
S:                 <r:relayData>
S:                     <k:keyRelayData xmlns:k="urn:ietf:params:xml:ns:keyrelay-1.0">
S:                         <k:name>example.org</k:name>
S:                         <k:keyData>
S:                             <secDNS:flags>256</secDNS:flags>
S:                             <secDNS:protocol>3</secDNS:protocol>
S:                             <secDNS:alg>8</secDNS:alg>
S:                             <secDNS:pubKey>cmlraXN0aGViZXN0</secDNS:pubKey>
S:                         </k:keyData>
S:                         <k:authInfo>
S:                             <domain:pw>JnSdBAZSxxzJ</domain:pw>
S:                         </k:authInfo>
S:                         <k:expiry>
S:                             <k:relative>P1M13D</k:relative>
S:                         </k:expiry>
S:                     </k:keyRelayData>
S:                 </r:relayData>
S:                 <r:paDate>1999-04-04T22:01:00.0Z</r:paDate>
S:                 <r:reID>ClientX</r:reID>
S:                 <r:acID>ClientY</r:acID>
S:             </r:panData>
S:         </resData>
S:         <trID>
S:             <clTRID>BCD-23456</clTRID>
S:             <svTRID>65432-WXY</svTRID>
S:         </trID>
S:     </response>
S: </epp>
4. Formal Syntax

4.1. Formal Syntax <relay> command and POLL response

<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:relay-1.0"
 xmlns:r="urn:ietf:params:xml:ns:relay-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">
  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0 protocol extension schema for relaying data.
    </documentation>
  </annotation>

  <import namespace="urn:ietf:params:xml:ns:epp-1.0"
    schemaLocation="epp-1.0.xsd"/>
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
    schemaLocation="eppcom-1.0.xsd"/>

  <element name="relay" type="r:relayDataType"/>
  <element name="panData" type="r:relayPanDataType"/>

  <complexType name="relayDataType">
    <sequence>
      <element name="relayData" type="epp:extAnyType"/>
      <element name="clTRID" type="epp:trIDStringType"
        minOccurs="0"/>
    </sequence>
  </complexType>

  <complexType name="relayPanDataType">
    <sequence>
      <element name="relayData" type="epp:extAnyType"/>
      <element name="paDate" type="dateTime"/>
      <element name="reID" type="eppcom:clIDType"/>
      <element name="acID" type="eppcom:clIDType"/>
    </sequence>
  </complexType>

</schema>
4.2. Formal Syntax <keyRelayData> data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:ietf:params:xml:ns:keyrelay-1.0"
  xmlns:k="urn:ietf:params:xml:ns:keyrelay-1.0"
  xmlns:epp="urn:ietf:params:xml:ns:epp-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 data.
    </documentation>
  </annotation>
  <import namespace="urn:ietf:params:xml:ns:epp-1.0"
    schemaLocation="epp-1.0.xsd" />
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"
    schemaLocation="eppcom-1.0.xsd" />
  <import namespace="urn:ietf:params:xml:ns:secDNS-1.1"
    schemaLocation="secdns-1.1.xsd" />
  <import namespace="urn:ietf:params:xml:ns:domain-1.0"
    schemaLocation="domain-1.0.xsd" />

  <element name="keyRelayData" type="k:keyRelayDataType" />

  <complexType name="keyRelayDataType">
    <sequence>
      <element name="name" type="eppcom:labelType" />
      <element name="keyData" type="secDNS:keyDataType"
        minOccurs="1"
        maxOccurs="unbounded" />
      <element name="authInfo" type="domain:authInfoType" />
      <element name="expiry" type="k: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

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

Four URI assignments must be completed by the IANA.

Registration request for the extension namespaces:

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

Registrant Contact: IESG

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

Registration request for the extension XML schemas:

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

Registrant Contact: IESG

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

6. Security Considerations

A registry MUST NOT perform any transformation on registration data under registry management when processing a <relay> command.

Any registrar can use this mechanism to put data on the message queue of another registrar, allowing for the potential of a denial of service attack. However this can, and SHOULD be detected by the registry. A registry MAY set a server policy which limits or rejects <relay> messages if it detects the mechanism is being abused.

For the <keyRelayData> data a correct <authInfo> element SHOULD be used as an indication that putting the key material on the registrar’s message queue is authorized by the _registrant_ of that domain name. This draft does not specify how this <authInfo> is provided to the registrar. This depends on how the DNS operator is authorised to perform DNS changes on behalf of the registrant through the registrar on record. This authorisation is not covered in this I-D.
7. References

7.1. Normative References


7.2. Informative References


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-eppext-keyrelay-00

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

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

1. Introducing the <relay> command, and thus seperating 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 exisiting EPP extensions.
Authors’ Addresses

Miek Gieben
Google
Email: miek@google.com

Marc Groeneweg
SIDN Labs
Meander 501
Arnhem 6825 MD
NL
Email: marc.groeneweg@sidn.nl
URI: https://www.sidn.nl/

Rik Ribbers
SIDN Labs
Meander 501
Arnhem 6825 MD
NL
Email: rik.ribbers@sidn.nl
URI: https://www.sidn.nl/

Antoin Verschuren
Email: ietf@antoin.nl
Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)
draft-ietf-eppext-launchphase-04

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 August 31, 2015.

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
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
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 [1], 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 [2], 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>example.tld</domain:name>
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">example.tld</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">example.tld</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:   </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
The <launch:check> element contains the following child elements:

<launcher: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 "example1.tld", "example2.tld", and "example3.tld" require claims notices during the "claims" launch phase:

```xml
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 xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
  C:        <domain:name>example1.tld</domain:name>
  C:        <domain:name>example2.tld</domain:name>
  C:        <domain:name>example3.tld</domain:name>
  C:      </domain:check>
  C:    </check>
  C:    <extension>
  C:      <launch:check xmlns:launch="urn:ietf:params:xml:ns:launch-1.0" 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 example1.tld, a claims notice is required for the domain name example2.tld in the "tmch", and a claims notice is required for the domain name example3.tld 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:    <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">example1.tld</launch:name>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">example2.tld</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">example3.tld</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJ1NrDO92vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJ1Nr2vDsAzasdf7EasdfgjX4R0000000002
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>example1.tld</domain:name>
C:      <domain:name>example2.tld</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 "example1.tld", "example2.tld", and "example3.tld" 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>example1.tld</domain:name>
C:      <domain:name>example2.tld</domain:name>
C:      <domain:name>example3.tld</domain:name>
C:    </domain:check>
C:   </check>
C:   <extension>
C:    <launch:check
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:     type="trademark"/>
C:   </extension>
C:  </command>
C:</epp>

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 example1.tld, matching trademarks are found for the domain name example2.tld in the "tmch", matching trademarks are found for domain name example3.tld in the "tmch" and "custom-tmch", for the "claims" launch phase:

```xml
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:name exists="0">example1.tld</launch:name>
S:     <launch:name exists="1">example2.tld</launch:name>
S:       <launch:claimKey validatorID="tmch">
S:         2013041500/2/6/rJ1NrDO92vDsZqgjX4R0000000001
S:       </launch:claimKey>
S:       <launch:claimKey validatorID="custom-tmch">
S:         20140423200/1/2/3/rJ1Nr2vDsAzasdf7EasdgjX4R0000000002
S:       </launch:claimKey>
S:     </launch:cd>
S:     <launch:cd>
S:       <launch:name exists="1">example3.tld</launch:name>
S:       <launch:claimKey validatorID="tmch">
S:         2013041500/2/6/rJ1NrDO92vDsZqgjX4R0000000001
S:       </launch:claimKey>
S:       <launch:claimKey validatorID="custom-tmch">
S:         20140423200/1/2/3/rJ1Nr2vDsAzasdf7EasdgjX4R0000000002
S:       </launch:claimKey>
S:     </launch:cd>
S:   </launch:chkData>
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 example.tld 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 xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.tld</domain:name>
C:       </domain:info>
C:     </info>
C:     <extension>
C:       <launch:info xmlns:launch="urn:ietf:params:xml:ns:launch-1.0" 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 example.tld:

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 xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>example.tld</domain:name>
C:    </domain:info>
C:   </info>
C:   <extension>
C:    <launch:info 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:

```xml
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>example.tld</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:   </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>example.tld</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:  </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:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>exampleone.tld</domain:name>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>

      <extension>
        <launch:create xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
          <launch:phase>sunrise</launch:phase>
          <launch:codeMark>
            <mark:mark xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
            ...
          </mark:mark>
        </launch:create>
      </extension>

    </create>
  </command>
</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:

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

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

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>exampleone.tld</domain:name>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
    <extension>
      <launch:create
       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
        <launch:phase>sunrise</launch:phase>
        <smd:encodedSignedMark
         xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
         ...
        </smd:encodedSignedMark>
      </launch:create>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</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>example.tld</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: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</launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:00.0Z</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</launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:30.0Z</launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
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:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.tld</domain:name>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
    <extension>
      <launch:create xmlns:launch="urn:ietf:params:xml:ns:launch-1.0" type="application">
        <launch:phase>landrush</launch:phase>
      </launch:create>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</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>exampleone.tld</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 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</launch:notAfter>
C:          <launch:acceptedDate>2012-06-19T09:01:30.0Z</launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
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:

```xml
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>example.tld</domain:name>
S:        <domain:crDate>2010-08-10T15:38:26.623854Z</domain:crDate>
S:      </domain:creData>
S:    </resData>
S:    <extension>
S:      <launch:creData
S:         xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>2393-9323-E08C-03B1
S:        <launch:applicationID>
S:      </launch:creData>
S:    </extension>
S:    </response>
S:</epp>
```
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>example.tld</domain:name>
C:        <domain:add>
C:          <domain:ns>
C:            <domain:hostObj>ns2.example.tld</domain:hostObj>
C:          </domain:ns>
C:        </domain:add>
C:        <domain:rem>
C:          <domain:ns>
C:            <domain:hostObj>ns1.example.tld</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:

```xml
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>example.tld</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:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- 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.
<?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>

</schema>
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 &lt;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>
<!-- Definition for the notice identifier -->
<complexType name="noticeIDType">
    <simpleContent>
        <extension base="launch:noticeIDValue">
            <attribute name="validatorID" type="launch:validatorIDType" use="optional"/>
        </extension>
    </simpleContent>
</complexType>

<!-- Definition for the validator identifier -->
<complexType name="validatorIDType">
    <simpleContent>
        <extension base="token">
            <attribute name="name" type="token" use="optional"/>
        </extension>
    </simpleContent>
</complexType>

<!-- Possible status values for sunrise application -->
<complexType name="statusValueType">
    <simpleContent>
        <extension base="token">
            <attribute name="name" type="token" use="optional"/>
        </extension>
    </simpleContent>
</complexType>

<!-- 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>

<complexType name="codeMarkType">
  <sequence>
    <element name="code" type="launch:codeType" minOccurs="0"/>
    <element ref="mark:abstractMark" minOccurs="0"/>
  </sequence>
</complexType>

<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>

<!--
codeMark Type that contains an optional code with mark information.
-->
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)
-->
<complexType name="checkFormType">
  <restriction base="token">
    <enumeration value="claims"/>
    <enumeration value="avail"/>
    <enumeration value="trademark"/>
  </restriction>
</complexType>
<!--
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>
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]. The following URI assignment is requested of IANA:

URI: urn:ietf:params:xml:ns: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

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-eppext-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

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. References

9.1. Normative References

[I-D.ietf-eppext-tmch-smd]


9.2. URIs


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.
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.

Authors’ Addresses
Mark and Signed Mark Objects Mapping
draft-ietf-eppext-tmch-smd-01

Abstract

This document describes the format of a mark and a digitally signed mark, referred to as a signed mark and the Signed Mark Data (SMD) file as defined by the ICANN Trademark Clearinghouse.

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 August 30, 2015.

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. Terminology ........................................... 3
2. Object Description ........................................... 3
  2.1. Holder and Contacts objects ............................ 3
  2.2. Mark .................................................. 5
  2.3. Signed Mark ........................................... 9
  2.4. Encoded Signed Mark ................................... 12
  2.5. Appendix A. base64 encoded signedMark ................. 13
3. Formal Syntax ................................................ 15
  3.1. Signed Mark Schema .................................... 15
  3.2. Mark Schema ........................................... 18
4. Implementation Status ....................................... 23
  4.1. Verisign EPP SDK ...................................... 24
  4.2. Verisign Consolidated Top Level Domain (CTLD) SRS ..... 24
  4.3. Verisign .COM / .NET SRS ................................ 25
  4.4. REngin v3.7 ........................................... 25
  4.5. Uniregistry Corp. Shared Registry System (uSRS) ...... 26
5. Acknowledgements ............................................. 26
6. Change History ............................................... 26
7. IANA Considerations ......................................... 28
8. Security Considerations ..................................... 28
9. Normative References ......................................... 28
Author’s Address ................................................. 29
1. Introduction

This document describes the format of a mark and a digitally signed mark, referred to as a signed mark and the Signed Mark Data (SMD) file as defined by the ICANN Trademark Clearinghouse. This document provides a framework that 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 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 objects associated with marks and signed marks. Empty complex element types and abstract elements are defined to support additional mark and signed mark definition using XSD substitution groups. Support for replacement through the XSD substitution groups is included in the descriptions of the objects.

2.1. Holder and Contacts objects

Marks are linked to Holder objects and optionally linked to Contacts objects. This section defines the <mark:holder> and <mark:contact> objects.
The child elements of <mark:holder> include:

- An OPTIONAL <mark:name> element that contains the name of the holder. A <mark:name> MUST be specified in case <mark:org> is not specified.
- An OPTIONAL <mark:org> element that contains the name of the organization holder of the mark. A <mark:org> MUST be specified in case <mark:name> is not 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 organization’s street address.
  - A <mark:city> element that contains the organization’s city.
  - An OPTIONAL <mark:sp> element that contains the organization’s state or province.
  - An OPTIONAL <mark:pc> element that contains the organization’s postal code.
  - A <mark:cc> element that contains the organization’s country code. This a two-character code from [ISO3166-2].
- An OPTIONAL <mark:voice> element that contains the organization’s voice telephone number.
- An OPTIONAL <mark:fax> element that contains the organization’s facsimile telephone number.
- An OPTIONAL <mark:email> element that contains the email address of the holder.

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> element that contains an identifier of the mark. The identifier MUST be globally unique in relation to the repository of marks. 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 Nice Classification class numbers of the mark as defined in the Nice List of Classes [1].

* Zero or more OPTIONAL `<mark:label>` elements that contain the A-label form 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>` element that contains an identifier of the mark. The identifier MUST be globally unique in relation to the
repository of marks. 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.

* 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 countries of the ruling. This is a two-character code from [ISO3166-2].

* Zero or more OPTIONAL <mark:label> elements that contain the A-label form 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.

* A <mark:refNum> element that contains the number of the mark of the treaty or statute.

* 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>` element that contains an identifier of the mark. The identifier MUST be globally unique in relation to the repository of marks. 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.

    * Zero or more OPTIONAL `<mark:label>` elements that contain the A-label form 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.

    * 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 the fragment of XML that is digitally signed using XML Signature [2]. The `<smd:signedMark>` includes a required "id" attribute of type XSD ID for use with an IDREF URI from the Signature element. The certificate of the issuer MAY be issued by a Certificate Authority (CA) that can be chained with the issuer’s certificate by the validating client.

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:

- The `<smd:id>` value is a concatenation of the local identifier, followed by a hyphen ("-", ASCII value 0x002D), followed by the issuer identifier.

- 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 URL of the issuer’s site.
  * An OPTIONAL `<smd:voice>` element that contains the issuer’s voice telephone number.

- A `<smd:notBefore>` element that contains the creation date and time of the signed mark.

- A `<smd:notAfter>` element that contains the expiration date and time of the signed mark.
A <mark:mark> element that contains the mark information as defined in the Mark (Section 2.2) section.


The following is an example <smd:signedMark> using the XML Signature [2] to sign all of the elements of <smd:signedMark> element.

```xml
<?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">
    ICANN TMCH TESTING TMV</smd:org>
    <smd:email>notavailable@example.com</smd:email>
    <smd:url>http://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 &amp; Validate</mark:markName>
      <mark:holder entitlement="owner">
        Ag corporation
        <mark:addr>
          1305 Bright Avenue
          Arcadia
          CA
          90028
          US
        </mark:addr>
      </mark:holder>
      <mark:contact type="agent">
        Tony Holland
        Ag corporation
        <mark:addr>
          1305 Bright Avenue
          Arcadia
          CA
          90028
          US
        </mark:addr>
        +1.2025562302
      </mark:contact>
    </mark:trademark>
  </mark:mark>
</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.

NOTE: Exclusive XML canonicalization SHOULD be used when generating the SMD. SHA256/RSA-SHA256 SHOULD be used for digesting and signing. The size of the RSA key SHOULD be at least 2048 bits.

2.4. Encoded Signed Mark

The <smd:encodedSignedMark> element contains an encoded form of the digitally signed <smd:signedMark> element, described in Section 2.3, with the encoding defined by the "encoding" attribute with the default "encoding" value of "base64". The "base64" encoded text of the <smd:encodedSignedMark> element MUST conform to [RFC2045]. A full example of a <smd:encodedSignedMark> element is presented in Appendix A.

</x509Certificate>
</x509Data>
</KeyInfo>
</Signature>
</smd:signedMark>

Lozano
Expires August 30, 2015
The following is an example of a `<smd:encodedSignedMark>` element that uses the default "base64" for encoding a `<smd:signedMark>` element.

```xml
<smd:encodedSignedMark
    xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">

PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZyJhbd extravQgPHN2Z1RyYWN0aW9uPnRyYW5zZmluPWZhbGlkYXRl

```

PoLo8lNtZdp5R3BzRlk4KICA8bWFyaIz4=
```
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) 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:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- Neither the name of the IETF nor the names of its 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.
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: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"
  schemaLocation="mark-1.0.xsd" />
<import namespace="http://www.w3.org/2000/09/xmldsig#"
  schemaLocation="xmldsig-core-schema.xsd"/>

<!--
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

```xml
<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"/>
                <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:e164Type" 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>
```

END
3.2. Mark 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:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- 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: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>

Lozano                   Expires August 30, 2015               [Page 18]
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="mark:token" minOccurs="0" />
    <element name="org" type="mark: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="mark: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="mark:dateTime" minOccurs="0" />
    <element name="regNum" type="mark:token" />
    <element name="regDate" type="mark:dateTime" />
    <element name="exDate" type="mark:dateTime" minOccurs="0" />
  </sequence>
</complexType>

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

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

Level of maturity: Operational Test Environment (OTE)

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-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-eppext-tmch-smd have been implemented
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-eppext-tmch-smd for its TLD registry.

Level of maturity: Production

Coverage: Implements parsing and validation of all aspects of draft-ietf-eppext-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-eppext-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.

6. Change History

Version draft-ietf-eppext-tmch-smd-00 to version draft-ietf-eppext-tmch-smd-01

Implementation Status section added.

Added type to the encoding element.

Version draft-lozano-tmch-smd-03 to version draft-ietf-eppext-tmch-smd-00
Internet-Draft resubmitted.

Version 02 to version 03

<smd:signedMark> example is now aligned with ICANN test SMDs.
<smd:encodedSignedMark> example is replaced with a public ICANN test SMD.
Several recommendations where added.

Version 01 to version 02

Change apID and regNum of trademark validated mark to token.
Change refNum of treatyOrStatute validated mark to token.
Change refNum of court validated mark to token.

Version 00 to version 01

Add missing email element to holderType.
Change ruling from an attribute to an element.

Version preview-01 to version 00

signedMarkType now ref mark:abstractMark.
Security section completed.

Version preview-00 to preview-01

Full example of an encodedSignedMark added.
signedMark example now fully validates with XSD.
Fixed labelType to allow two-character labels.
Missing mark:protectionType added in the XSD.
Issuer email is now required.
7. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. Three 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: See the "Author’s Address" section of this document.

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

Registration request for the Mark namespace:

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

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

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

8. Security Considerations

The object mapping described in this document does not provide any security services or introduce any additional considerations.

9. Normative References


[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
Internet-Draft            Mark and Signed Mark             February 2015


STD 69, RFC 5730, August 2009.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running
Code: The Implementation Status Section", RFC 6982,
July 2013.

[WIPO.ST3] WIPO, "Recommended standard on two-letter codes for the
representation of states, other entities and


Author’s Address

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

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