Internet Engineering Task Force J. Gould
Internet-Draft VeriSign, Inc.
Intended status: Standards Track W. Tan
Expires: May 16, 2014 Cloud Registry
G. Brown
CentralNic Ltd
November 12, 2013

Launch Phase Mapping for the Extensible Provisioning Protocol (EPP) draft-tan-epp-launchphase-12

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on May 16, 2014.

Copyright Notice

include Simplified BSD License text as described in Section 4.e of

Table of Contents

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 TMCH Functional Specification [1] 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 [draft-lozano-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 [draft-lozano-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 [draft-lozano-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.

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 [1], 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.

Certain status values MAY be combined. For example, an application or registration may be both "invalid" and "rejected". Additionally, certain statuses 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

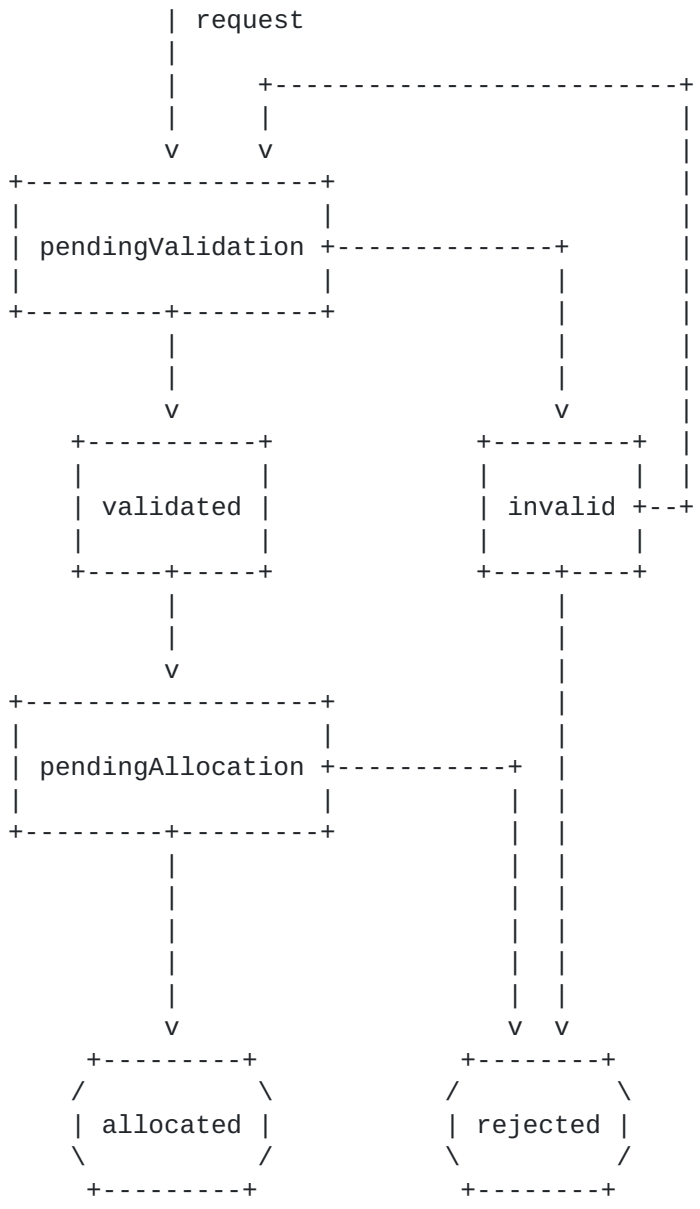

Figure 1

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:        ...
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:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

2.6. Mark Validation Models

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

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

mark The mark information is passed without any other validation element. The server will use some custom form of validation to validate that the mark information is authentic. This model is supported using the <launch:codeMark> element with just the <mark: mark> (Section 2.6.2) element.

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

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

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

2.6.1. <launch:codeMark> element

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

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

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

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

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

```
   </launch:codeMark>
```

2.6.2. <mark:mark> element

A <mark:mark> element describes an applicant's prior right to a given domain name that is used with the "mark", "mark with code", and the "signed mark" validation models. The <mark:mark> element is defined in [draft-lozano-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 [draft-lozano-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 [draft-lozano-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 [draft-lozano-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 [draft-lozano-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 [draft-lozano-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 two forms of the extension to the EPP <check> command: the Claims Check Form (Section 3.1.1) and the Availability Check Form (Section 3.1.2). The <launch:check> element "type" attribute defines the form, with the value of "claims" for the Claims Check Form (Section 3.1.1) and with the value of "avail" for the Availability Check Form (Section 3.1.2). 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. 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. If there is at least one matching trademark that exists for the domain name, a <launch:claimKey> element is returned. The client MAY then use the value of the <launch:claimKey> element to obtain information needed to generate the Trademark Claims Notice from Trademark Validator based on the Validator Identifier (Section 2.2). The unique notice identifier of the Trademark Claims Notice MUST be passed in the <launch:noticeID> element of the extension to the Create Command (Section 3.3).

The <domain:name> elements in the EPP <check> command of EPP domain name mapping [RFC5731] define the domain names to check for matching trademarks. The <launch:check> element contains the following child elements:

<launch:phase> The launch phase that SHOULD be "claims".

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" and "example2.tld" have any matching trademarks during the "claims" launch phase:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:   <check>
C:    <domain:check
C:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:      <domain:name>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="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 launch phase that SHOULD be "claims".
<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 for the claims launch phase. A value of "0" (or "false") means that a matching trademark does not exist.

<launch:claimKey> An OPTIONAL claim key 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.

Example Claims Check response when no matching trademarks are found for the domain name example1.tld and matching trademarks are found for the domain name example2.tld for the "claims" launch phase:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:     <msg>Command completed successfully</msg>
S:    </result>
S:   <extension>
S:     <launch:chkData
S:      xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:      <launch:phase>claims</launch:phase>
S:      <launch:cd>
S:        <launch:name exists="0">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: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].

3.2. 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
C:     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
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:       includeMark="true">
C:      <launch:phase>sunrise</launch:phase>
C:      <launch:applicationID>abc123</launch:applicationID>
C:    </launch:info>
C:   </extension>
C:   <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> domain command with the <launch:info> extension to retrieve information for the sunrise registration for 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
C:     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
C:     xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:      <launch:phase>sunrise</launch:phase>
C:    </launch:info>
C:   </extension>
C:   <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

<launch:phase> The phase during which the application was submitted,
or is associated with, that matches the associated <info> command
<launch:phase>.
<launch:applicationID> OPTIONAL Application Identifier of the Launch
Application.
<launch:status> OPTIONAL status of the Launch Application using one of the supported status values (Section 2.4).
<mark:mark> Zero or more <mark:mark> (Section 2.6.2) elements.

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>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:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.3. EPP <create> Command

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

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

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

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

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

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:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>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:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <mark:mark
C:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:            ...
C:          </mark:mark>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>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="sample">
C:            49FD46E6C4B45C55D4AC</launch:code>
C:          <mark:mark
C:           xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:           ...
C:          </mark:mark>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>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>sunrise</launch:phase>
C:        <smd:signedMark id="signedMark"
C:         xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:         ...
C:        </smd:signedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>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:        <launch:phase>sunrise</launch:phase>
C:        <smd:encodedSignedMark
C:         xmlns:smd="urn:ietf:params:xml:ns:signedMark-1.0">
C:         ...
C:        </smd:encodedSignedMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.3.2. Claims Create Form

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

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> MUST contain the value of "claims" to indicate the claims launch phase.
<launch:notice>
<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 "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:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:       xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>claims</launch:phase>
C:        <launch:notice>
C:          <launch:noticeID validatorID="tmch">
C:            49FD46E6C4B45C55D4AC
C:          </launch:noticeID>
C:          <launch:notAfter>2012-06-19T10:00:10.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2012-06-19T09:01:30.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.3.3. General Create Form

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

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

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

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

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

3.3.4. Mixed Create Form

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

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>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
C:          </launch:notAfter>
C:          <launch:acceptedDate>2012-06-19T09:01:30.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

3.3.5. Create Response

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

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

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <domain:creData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:       <domain:name>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:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.4. EPP <update> Command

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

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

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

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

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>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:

```
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

THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE

OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BEGIN

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

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

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

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

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

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

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

```
        <element name="applicationID" type="launch:applicationIDType"/
>
      </sequence>
    </complexType>

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

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

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


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

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

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

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

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

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

```
      </restriction>
    </simpleType>

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

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

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

```
      <attribute name="type" type="launch:objectType"/>
    </complexType>

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


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


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


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

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

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

<!--
 <check> response elements.
 -->
<complexType name="chkDataType">
  <sequence>
    <element name="phase" type="launch:phaseType"/>
    <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"/>
  </sequence>
</complexType>

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

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

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

  </schema>
  END
```

5. 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, 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.

6. Change History

6.1. 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]".

6.2. 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

6.3. 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.

6.4. 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.

6.5. Change from 04 to 05

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

6.6. 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.

6.7. 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.

6.8. 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.

6.9. 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.

6.10. 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.

6.11. 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.

6.12. 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.

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 Launch namespace:

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

8. 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.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.

[RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, August 2009.

[draft-lozano-smd] Lozano, G., "Mark and Signed Mark Objects Mapping".

[1] <http://tools.ietf.org/html/draft-lozano-tmch-func-spec>

Authors' Addresses

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

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

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

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

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

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