

Internet Engineering Task Force (IETF)  
Request for Comments: 8334  
Category: Standards Track  
ISSN: 2070-1721

J. Gould  
VeriSign, Inc.  
W. Tan  
Cloud Registry  
G. Brown  
CentralNic Ltd  
March 2018

## Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)

### 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 is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc8334>.

### Copyright Notice

Copyright (c) 2018 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 (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1.	Introduction . . . . .	3
1.1.	Conventions Used in This Document . . . . .	3
2.	Object Attributes . . . . .	4
2.1.	Application Identifier . . . . .	4
2.2.	Validator Identifier . . . . .	5
2.3.	Launch Phases . . . . .	5
2.3.1.	Trademark Claims Phase . . . . .	6
2.4.	Status Values . . . . .	9
2.4.1.	State Transition . . . . .	11
2.5.	Poll Messaging . . . . .	12
2.6.	Mark Validation Models . . . . .	14
2.6.1.	<launch:codeMark> Element . . . . .	15
2.6.2.	<mark:mark> Element . . . . .	16
2.6.3.	Digital Signature . . . . .	16
2.6.3.1.	<smd:signedMark> Element . . . . .	16
2.6.3.2.	<smd:encodedSignedMark> Element . . . . .	16
3.	EPP Command Mapping . . . . .	17
3.1.	EPP <check> Command . . . . .	17
3.1.1.	Claims Check Form . . . . .	17
3.1.2.	Availability Check Form . . . . .	22
3.1.3.	Trademark Check Form . . . . .	23
3.2.	EPP <info> Command . . . . .	26
3.3.	EPP <create> Command . . . . .	30
3.3.1.	Sunrise Create Form . . . . .	30
3.3.2.	Claims Create Form . . . . .	36
3.3.3.	General Create Form . . . . .	39
3.3.4.	Mixed Create Form . . . . .	40
3.3.5.	Create Response . . . . .	42
3.4.	EPP <update> Command . . . . .	43
3.5.	EPP <delete> Command . . . . .	44
3.6.	EPP <renew> Command . . . . .	46
3.7.	EPP <transfer> Command . . . . .	46
4.	Formal Syntax . . . . .	46
4.1.	Launch Schema . . . . .	46
5.	IANA Considerations . . . . .	54
5.1.	XML Namespace . . . . .	54
5.2.	EPP Extension Registry . . . . .	55
6.	Security Considerations . . . . .	55
7.	References . . . . .	56
7.1.	Normative References . . . . .	56
7.2.	Informative References . . . . .	57
	Acknowledgements . . . . .	57
	Authors' Addresses . . . . .	58

## 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 as they begin operation 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. Multiple launch phases and multiple models are supported to enable the launch of a domain name registry. Server policy determines what is supported and validated. Communication of the server policy is typically performed using an out-of-band mechanism that is not specified in this document.

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, RFC 7848 [RFC7848] 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

XML [W3C.REC-xml11-20060816] 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 whitespace in examples are provided only to illustrate element relationships and are not a REQUIRED feature of this protocol. The use of "..." is used as shorthand for elements defined outside this document.

A Launch Registration is a domain name registration during a launch phase when the server uses a "first-come, first-served" model. Only a single registration for a domain name can exist in the server at a time.

A Launch Application represents the intent to register a domain name during a launch phase when the server accepts multiple applications for a domain name, and the server later selects one of the applications to allocate as a registration. Many Launch Applications for a domain name can exist in the server at a time.

The XML namespace prefix "launch" is used for the namespace "urn:ietf:params:xml:ns:launch-1.0", 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.

The XML namespace prefix "smd" is used for the namespace "urn:ietf:params:xml:ns:signedMark-1.0" [RFC7848], 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.

The XML namespace prefix "mark" is used for the namespace "urn:ietf:params:xml:ns:mark-1.0" [RFC7848], 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 <domain:create> command 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 pendingCreate status [RFC5731], 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.

## 2.2. Validator Identifier

The Validator Identifier is unique to the server and is the identifier for a Trademark Validator, which 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 unique set of Validator Identifier values supported by the server is up to server policy. The Internet Corporation for Assigned Names and Numbers (ICANN) Trademark Clearinghouse (TMCH) is the default Trademark Validator and is reserved for 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>` and `<smd:id>` elements defined in [RFC7848]. Both the Validator Identifier and the Issuer Identifier used MUST be unique in the server. If the ICANN TMCH is not used or multiple Trademark Validators are used, the server MUST define the list of supported validator identifiers and MUST make this information available to clients using a mutually acceptable, out-of-band mechanism.

The Validator Identifier may define a non-Trademark Validator that supports a form of claims, where claims and a Validator Identifier can be used for purposes beyond trademarks.

## 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, and return an EPP error result code of 2306 [RFC5730] 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" launch phase when non-trademark holders are allowed to register domain names with steps taken to address a large volume of initial registrations.

**claims:** The phase, as defined in Section 2.3.1, in which a claims notice must be displayed to a prospective registrant of a domain name that matches trademarks.

**open:** A 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 Section 2.3.1, 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>.

### 2.3.1. Trademark Claims Phase

The Trademark Claims Phase is when a claims notice must be displayed to a prospective registrant of a domain name that matches trademarks. See [ICANN-TMCH] for additional details of trademark claims handling. The source of the trademarks is a Trademark Validator, and the source of the claims notice information is a Claims Notice Information Service (CNIS), which may be directly linked to a Trademark Validator. The client interfaces with 1) the server to determine if a trademark exists for a domain name, 2) a CNIS to get the claims notice information, and 3) the server to pass the claims notice acceptance information in a create command. This document supports the Trademark Claims Phase in two ways, including:

**Claims Check Form:** Is defined in Section 3.1.1 and is used to determine whether or not there are any matching trademarks for a domain name. If there is at least one matching trademark that exists for the domain name, a claims key is returned. The mapping of domain names and the claims keys is based on an out-of-band interface between the server and the Trademark Validator. The

CNIS associated with the claims key Validator Identifier (Section 2.2) **MUST** accept the claims key as the basis for retrieving the claims information.

**Claims Create Form:** Is defined in Section 3.3.2 and is used to pass the claims notice acceptance information in a create command. The notice identifier (<launch:noticeID>) format, validation rules, and server processing is up to the interface between the server and the Trademark Validator. The CNIS associated with the Validator Identifier (Section 2.2) **MUST** generate a notice identifier compliant with the <launch:noticeID> element.

The following shows the Trademark Claims Phase registration flow:

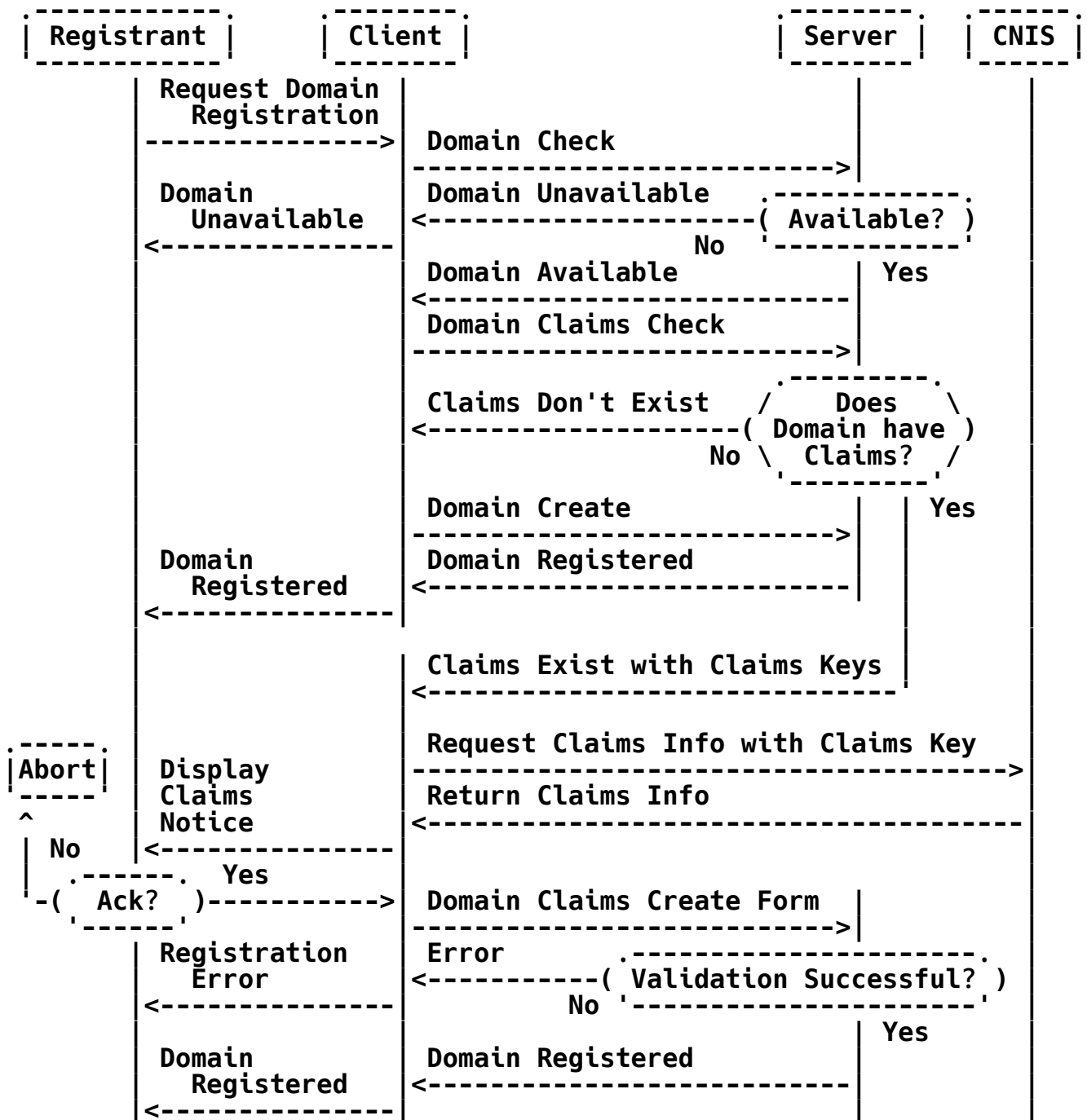


Figure 1



## 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 "pendingCreate" status [RFC5731] if a launch status is supported and is not one of the final statuses ("allocated" and "rejected").

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. This is a possible end state of an application or registration object.

**rejected:** The application or registration object was not provisioned. This 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, as defined in [RFC5646], 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 use one of the non-"custom" status values.

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

The transitions between the states is a matter of server policy. This diagram defines one possible set of permitted transitions.

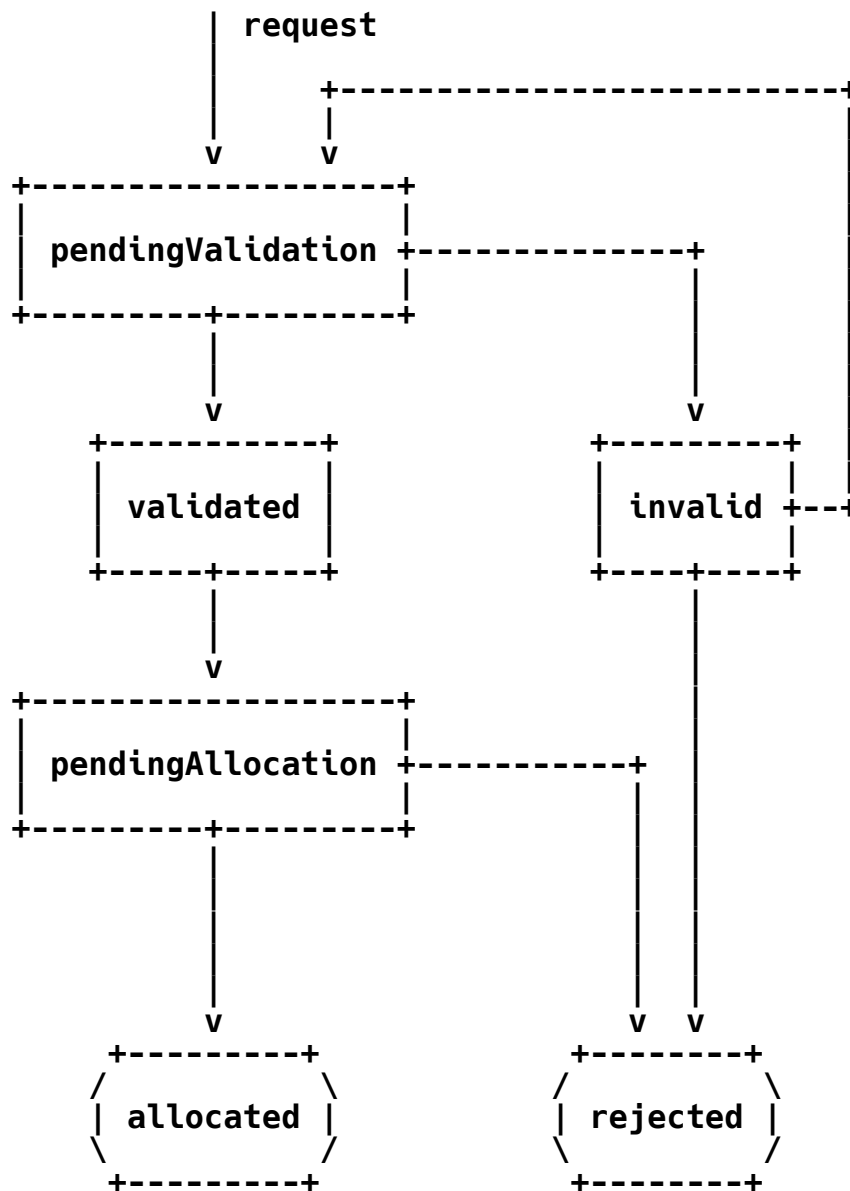


Figure 2

## 2.5. Poll Messaging

A Launch Application **MUST** be handled as an EPP domain name object as specified in RFC 5731 [RFC5731], with the "pendingCreate" status and launch status values defined in Section 2.4. A Launch Registration **MAY** be handled as an EPP domain name object as specified in RFC 5731 [RFC5731], with the "pendingCreate" status and launch status values defined in Section 2.4. As a Launch Application or Launch Registration transitions between the status values defined in Section 2.4, the server **SHOULD** insert poll messages, per [RFC5730], for the applicable intermediate statuses, including the "pendingValidation", "validated", "pendingAllocation", and "invalid" statuses, using the <domain:infData> element with the <launch:infData> extension. The <domain:infData> element **MAY** contain non-mandatory information, like contact and name server information. Also, further extensions that would normally be included in the response of a <domain:info> command, per [RFC5731], **MAY** be included. For the final statuses, including the "allocated" and "rejected" statuses, the server **MUST** insert a <domain:panData> poll message, per [RFC5731], with the <launch:infData> extension.

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>2013-04-04T22:01:00.0Z</qDate>
S:      <msg>Application pendingAllocation.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>abc123</launch:applicationID>
S:        <launch:status s="pendingAllocation"/>
S:      </launch:infData>
```

```

S:    </extension>
S:    <trID>
S:        <clTRID>ABC-12345</clTRID>
S:        <svTRID>54322-XYZ</svTRID>
S:    </trID>
S: </response>
S:</epp>

```

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

```

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>2013-04-04T22:01:00.0Z</qDate>
S:      <msg>Application successfully allocated.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:panData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name paResult="1">domain.example</domain:name>
S:        <domain:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </domain:paTRID>
S:        <domain:paDate>2013-04-04T22:00:00.0Z</domain:paDate>
S:      </domain:panData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>abc123</launch:applicationID>
S:        <launch:status s="allocated"/>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>

```

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

```

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>2013-04-04T22:01:00.0Z</qDate>
S:      <msg>Registration successfully allocated.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:panData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name paResult="1">domain.example</domain:name>
S:        <domain:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </domain:paTRID>
S:        <domain:paDate>2013-04-04T22:00:00.0Z</domain:paDate>
S:      </domain:panData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:status s="allocated"/>
S:      </launch:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>

```

## 2.6. Mark Validation Models

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

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

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

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

**signed mark:** The mark information is digitally signed as described in the Digital Signature section (Section 2.6.3). 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 is used by the "code", "mark", and "code with mark" validation models and 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 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 the code originated from, with no default value.

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

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 "signed mark" validation models. The `<mark:mark>` element is defined in [RFC7848]. 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 [RFC7848].

### 2.6.3. Digital Signature

Digital signatures MAY be used by the server to validate the mark information, when using the "signed mark" validation model with the `<smd:signedMark>` (Section 2.6.3.1) and `<smd:encodedSignedMark>` (Section 2.6.3.2) elements. When using digital signatures, the server MUST validate the digital signature.

#### 2.6.3.1. `<smd:signedMark>` Element

The `<smd:signedMark>` element contains the digitally signed mark information. The `<smd:signedMark>` element is defined in [RFC7848]. 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 [RFC7848].

#### 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 [RFC7848]. 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 [RFC7848].



### 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 commands 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), "avail" for the Availability Check Form (Section 3.1.2), and "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 [RFC5730] 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 require 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, which 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 the 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 element:

<launch:phase>: Contains the value of the active launch phase of the server. The server SHOULD validate the value according to Section 2.3.

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <domain:check
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>domain1.example</domain:name>
C:          <domain:name>domain2.example</domain:name>
C:          <domain:name>domain3.example</domain:name>
C:        </domain:check>
C:      </check>
C:    <extension>
C:      <launch:check
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        type="claims">
C:          <launch:phase>claims</launch:phase>
C:        </launch:check>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

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

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

<launch:name>: Contains the fully qualified name of the queried domain name. This element MUST contain an "exists" attribute whose value indicates if a matching trademark exists for the domain name that requires the use of the "Claims Create Form" on a Domain Create Command. A value of "1" (or "true") means that a matching trademark does exist and that the "Claims Create Form" is required on a Domain Create Command. A value

of "0" (or "false") means that a matching trademark does not exist or that the "Claims Create Form" is NOT required on a Domain Create Command.

**<launch:claimKey>**: Zero or more OPTIONAL claim keys that MAY be passed to a third-party Trademark Validator such as the ICANN 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.

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <launch:chkData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>claims</launch:phase>
S:        <launch:cd>
S:          <launch:name exists="0">domain1.example</launch:name>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain2.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJ1NrD092vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain3.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJ1NrD092vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJ1Nr2vDsAzasdff7EasdfgjX4R0000000002
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 element:

<launch:phase>: The launch phase to which domain name availability should be determined. The server **SHOULD** validate the value and return an EPP error result code of 2306 [RFC5730] if it is invalid.

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <domain:check
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>domain1.example</domain:name>
C:          <domain:name>domain2.example</domain:name>
C:        </domain:check>
C:      </check>
C:    <extension>
C:      <launch:check
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        type="avail">
C:        <launch:phase name="idn-release">custom</launch:phase>
C:      </launch:check>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The Availability Check Form does not define any extension to the response of a <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 Trademark 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 the 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":

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <domain:check
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>domain1.example</domain:name>
C:          <domain:name>domain2.example</domain:name>
C:          <domain:name>domain3.example</domain:name>
C:        </domain:check>
C:      </check>
C:    <extension>
C:      <launch:check
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        type="trademark"/>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

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

<launch:name>: Contains the fully qualified name of the queried domain name. This element MUST contain an "exists" attribute whose value indicates if a matching trademark exists for the domain name. A value of "1" (or "true") means that a matching trademark does exist. A value of "0" (or "false") means that a matching trademark does not exist.

<launch:claimKey>: Zero or more OPTIONAL claim keys that MAY be passed to a third-party Trademark Validator such as the ICANN 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.

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

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <launch:chkData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:cd>
S:          <launch:name exists="0">domain1.example</launch:name>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain2.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJ1NrD092vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:        </launch:cd>
S:        <launch:cd>
S:          <launch:name exists="1">domain3.example</launch:name>
S:          <launch:claimKey validatorID="tmch">
S:            2013041500/2/6/9/rJ1NrD092vDsAzf7EQzgjX4R0000000001
S:          </launch:claimKey>
S:          <launch:claimKey validatorID="custom-tmch">
S:            20140423200/1/2/3/rJ1Nr2vDsAzasdff7EasdfgjX4R0000000002
S:          </launch:claimKey>
S:        </launch:cd>
S:      </launch:chkData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

### 3.2. EPP <info> Command

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

The EPP <info> command is used to retrieve information for a Launch Registration or Launch Application. The Application Identifier (Section 2.1) returned in the <launch:creData> element of the create response (Section 3.3) can be 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. The server **SHOULD** validate the value and return an EPP error result code of 2306 [RFC5730] if it is invalid.

<launch:applicationID>: OPTIONAL application identifier of the Launch Application.

The following is an example `<info>` domain command with the `<launch:info>` extension to retrieve information for the sunrise application for domain.example and application identifier "abc123":

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <domain:info
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>domain.example</domain:name>
C:        </domain:info>
C:      </info>
C:    <extension>
C:      <launch:info
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        includeMark="true">
C:          <launch:phase>sunrise</launch:phase>
C:          <launch:applicationID>abc123</launch:applicationID>
C:        </launch:info>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The following is an example `<info>` domain command with the `<launch:info>` extension to retrieve information for the sunrise registration for domain.example:

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

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

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

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

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

`<mark:mark>`: Zero or more `<mark:mark>` (Section 2.6.2) elements only if the "includeMark" attribute is "true" in the command.

The following is an example <info> domain response using the <launch:infData> extension with the mark information:

```

S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="pendingCreate"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <launch:infData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>abc123</launch:applicationID>
S:        <launch:status s="pendingValidation"/>
S:        <mark:mark
S:          xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
S:            ...
S:          </mark:mark>
S:        </launch:infData>
S:      </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>

```

### 3.3. EPP <create> Command

There are four forms of the extension to the EPP <create> command that include the Sunrise Create Form (Section 3.3.1), the Claims Create Form (Section 3.3.2), the General Create Form (Section 3.3.3), and the Mixed Create Form (Section 3.3.4). The form used 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. 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 Mark 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, and return an EPP error result code of 2306 [RFC5730] if the type is incorrect. The <launch:create> element contains the following child elements:

<launch:phase>: The identifier for the launch phase. The server SHOULD validate the value according to Section 2.3.

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

<launch:codeMark>: Zero or more <launch:codeMark> elements. The <launch:codeMark> child elements are defined in "<launch:codeMark> Element" (Section 2.6.1).

<smd:signedMark>: Zero or more <smd:signedMark> elements. The <smd:signedMark> child elements are defined in "<smd:signedMark> Element" (Section 2.6.3.1).

<smd:encodedSignedMark>: Zero or more <smd:encodedSignedMark> elements. The <smd:encodedSignedMark> child elements are defined in "<smd:encodedSignedMark> Element" (Section 2.6.3.2).

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

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



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

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrar>jd1234</domain:registrar>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:codeMark>
C:          <launch:code validatorID="sample">
C:            49FD46E6C4B45C55D4AC</launch:code>
C:          <mark:mark
C:            xmlns:mark="urn:ietf:params:xml:ns:mark-1.0">
C:            ...
C:          </mark:mark>
C:        </launch:codeMark>
C:      </launch:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

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

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

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

### 3.3.2. Claims Create Form

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

A <launch:create> element is sent along with the regular <create> domain command. The <launch:create> element has an OPTIONAL "type" attribute that defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created, and return an EPP error result code of 2306 [RFC5730] if the type is incorrect. 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 according to Section 2.3.

**<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 that is used to define the Validator Identifier (Section 2.2); it's value indicates which Trademark Validator is the source of the claims notice, with the default being the ICANN TMCH.

**<launch:notAfter>:** Expiry of the claims notice.

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

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrar>jd1234</domain:registrar>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>claims</launch:phase>
C:        <launch:notice>
C:          <launch:noticeID validatorID="tmch">
C:            370d0b7c9223372036854775807</launch:noticeID>
C:          <launch:notAfter>2014-06-19T10:00:00.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:00.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:        <launch:notice>
C:          <launch:noticeID validatorID="custom-tmch">
C:            470d0b7c9223654313275808</launch:noticeID>
C:          <launch:notAfter>2014-06-19T10:00:00.0Z
C:          </launch:notAfter>
C:          <launch:acceptedDate>2014-06-19T09:00:30.0Z
C:          </launch:acceptedDate>
C:        </launch:notice>
C:      </launch:create>
C:    </extension>
C:    <cLTRID>ABC-12345</cLTRID>
C:  </command>
C:</epp>
```

### 3.3.3. General Create Form

The General Create Form of the extension to the EPP domain name mapping [RFC5731] includes the launch phase and optionally the object type to create. The OPTIONAL "type" attribute defines the expected type of object ("application" or "registration") to create. The server SHOULD validate the "type" attribute, when passed, against the type of object that will be created, and return an EPP error result code of 2306 [RFC5730] if the type is incorrect.

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

<launch:phase>: Contains the value of the active launch phase of the server. The server SHOULD validate the value according to Section 2.3.

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

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

#### 3.3.4. Mixed Create Form

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



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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domainone.example</domain:name>
C:        <domain:registrar>jd1234</domain:registrar>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <launch:create
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0"
C:        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 add a <launch:creData> element to the regular EPP <resData> to indicate that the server generated an 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.

The following is an example response when multiple overlapping applications are supported by the server:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <domain:creData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:crDate>2010-08-10T15:38:26.623854Z</domain:crDate>
S:      </domain:creData>
S:    </resData>
S:    <extension>
S:      <launch:creData
S:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
S:        <launch:phase>sunrise</launch:phase>
S:        <launch:applicationID>2393-9323-E08C-03B1
S:      </launch: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.

When an EPP <update> command with the extension is sent to a server that does not support Launch Applications, it will fail. A server that does not support Launch Applications during its launch phase **MUST** return an EPP error result code of 2102 [RFC5730] 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. The server **SHOULD** validate the value and return an EPP error result code of 2306 [RFC5730] if it is invalid.

<launch:applicationID>: The application identifier for which the client wishes to update.

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:add>
C:          <domain:ns>
C:            <domain:hostObj>ns2.domain.example</domain:hostObj>
C:          </domain:ns>
C:        </domain:add>
C:        <domain:rem>
C:          <domain:ns>
C:            <domain:hostObj>ns1.domain.example</domain:hostObj>
C:          </domain:ns>
C:        </domain:rem>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <launch:update
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:        <launch:phase>sunrise</launch:phase>
C:        <launch:applicationID>abc123</launch:applicationID>
C:      </launch:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

### 3.5. EPP <delete> Command

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

A client **MUST NOT** pass the extension on an EPP <delete> command to a server that does not support Launch Applications. A server that does

not support Launch Applications during its launch phase MUST return an EPP error result code of 2102 [RFC5730] 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. The server SHOULD validate the value and return an EPP error result code of 2306 [RFC5730] if it is invalid.

<launch:applicationID>: The application identifier for which the client wishes to delete.

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

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <delete>
C:      <domain:delete
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:          <domain:name>domain.example</domain:name>
C:        </domain:delete>
C:      </delete>
C:    <extension>
C:      <launch:delete
C:        xmlns:launch="urn:ietf:params:xml:ns:launch-1.0">
C:          <launch:phase>sunrise</launch:phase>
C:          <launch:applicationID>abc123</launch:applicationID>
C:        </launch:delete>
C:      </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

### 3.6. EPP <renew> Command

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

### 3.7. EPP <transfer> Command

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

## 4. Formal Syntax

The EPP Launch Phase Mapping schema is presented in Section 4.1.

The formal syntax presented 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) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT

OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BEGIN

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

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

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

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

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

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

<!-- Definition for launch phase. Name is an
optional attribute used to extend the phase type.
For example, when using the phase type value
of "custom", 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 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"/>
  </restriction>
</simpleType>
```

```
        <enumeration value="allocated"/>
        <enumeration value="rejected"/>
        <enumeration value="custom"/>
    </restriction>
</simpleType>

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

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

<!-- Child elements for the create command -->
<complexType name="createType">
    <sequence>
        <element
            name="phase"
            type="launch:phaseType"/>
        <choice minOccurs="0">
            <element
                name="codeMark"
```

```
        type="launch:codeMarkType"
        maxOccurs="unbounded"/>
      <element
        ref="smd:abstractSignedMark"
        maxOccurs="unbounded"/>
      <element
        ref="smd:encodedSignedMark"
        maxOccurs="unbounded"/>
    </choice>
    <element
      name="notice"
      type="launch:createNoticeType"
      minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
  <attribute
    name="type"
    type="launch:objectType"/>
</complexType>

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

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

<!-- Child elements of check (Claims Check Command) -->
<complexType name="checkType">
  <sequence>
```

```
<element
  name="phase"
  type="launch:phaseType"
  minOccurs="0"/>
</sequence>
<attribute
  name="type"
  type="launch:checkFormType"
  default="claims"/>
</complexType>

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

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

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

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

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

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

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

```
</simpleContent>
</complexType>

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

</schema>
END
```

## 5. IANA Considerations

### 5.1. XML Namespace

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

IANA has registered the launch namespace as follows:

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

Registrant Contact: IESG

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

IANA has registered the launch XML schema as follows:

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

Registrant Contact: IESG

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

## 5.2. EPP Extension Registry

IANA has registered the EPP extension described in this document in the "Extensions for the Extensible Provisioning Protocol (EPP)" 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: RFC 8334

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

## 6. 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.

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 an <info> operation with filtered output by clients

other than the sponsoring client, in which case the <domain:infData> and <launch:infData> responses SHOULD be filtered to include only fields that are publicly accessible.

## 7. References

### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [RFC7848] Lozano, G., "Mark and Signed Mark Objects Mapping", RFC 7848, DOI 10.17487/RFC7848, June 2016, <<https://www.rfc-editor.org/info/rfc7848>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [W3C.REC-xml11-20060816] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., Yergeau, F., and J. Cowan, "Extensible Markup Language (XML) 1.1 (Second Edition)", World Wide Web Consortium Recommendation REC-xml11-20060816, August 2006, <<http://www.w3.org/TR/2006/REC-xml11-20060816>>.



## 7.2. Informative References

### [ICANN-TMCH]

Lozano, G., "ICANN TMCH functional specifications", Work in Progress, draft-ietf-regext-tmch-func-spec-03, July 2017.

### [RFC7451]

Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <<https://www.rfc-editor.org/info/rfc7451>>.

## 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 Harald Alvestrand, Ben Campbell, Spencer Dawkins, Jothan Frakes, Keith Gaughan, Seth Goldman, Scott Hollenbeck, Michael Holloway, Jan Jansen, Rubens Kuhl, Mirja Kuehlewind, Warren Kumari, Ben Levac, Gustavo Lozano, Klaus Malorny, Alexander Mayrhofer, Alexey Melnikov, Patrick Mevzek, James Mitchell, Francisco Obispo, Mike O'Connell, Eric Rescorla, Bernhard Reutner-Fischer, Sabrina Tanamal, Trung Tran, Ulrich Wisser, and Sharon Wodjenski.

Some of the description of the Trademark Claims Phase was based on the work done by Gustavo Lozano in the ICANN TMCH functional specifications.

**Authors' Addresses**

James Gould  
VeriSign, Inc.  
12061 Bluemont Way  
Reston, VA 20190  
United States of America

Email: [jgould@verisign.com](mailto:jgould@verisign.com)  
URI: <http://www.verisign.com>

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

Phone: +61 414 710899  
Email: [wil@cloudregistry.net](mailto:wil@cloudregistry.net)  
URI: <http://www.cloudregistry.net>

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

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