Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

J. Gould
L. Jia
VeriSign, Inc.
R. Carney
J. Kolker
GoDaddy Inc.
October 22, 2018

Registry Mapping for the Extensible Provisioning Protocol (EPP) draft-gould-carney-regext-registry-04

Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for provisioning registry zones (e.g. top-level domains) in a Domain Name Registry. The attributes of a registry zone include the features and policies of the registry zone.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 2019.

Table of Contents

1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This document describes a Domain Name Registry Mapping, referred to as Registry Mapping, for the Extensible Provisioning Protocol (EPP) [RFC5730]. A Domain Name Registry can service one or more registry zones (e.g. top-level domains) with a variety of supported services and policies. A registry zone, also referred to as a "zone" in this document, is a domain name that the Domain Name Registry supports provisioning operations to manage. The registry zone and the associated DNS zone has an overlapping data set, where the registry zone is the source for the generation of a DNS zone. A registry zone is typically a top-level domain name, but it can be a domain name at any domain name level. A registry zone can be the source for multiple resolution services like DNS and WHOIS.

This mapping enables the provisioning of the features and policies of the registry zones in the Domain Name Registry. A Domain Name Registry MAY support a subset of all of the commands defined in this mapping and can authorize different clients to execute specific commands. For example, all clients may be capable of executing the EPP Query Commands (Section 3.1), while internal clients or pre-defined external clients may be capable of executing the EPP Transform Commands (Section 3.2) for a specific set of zones. It is up to server policy what commands are supported and to define the clients that are authorized to execute the commands for the registry zones. The server MUST return a 2101 error response for an unimplemented command and MUST return a 2201 error response for an unauthorized command. The server policy can be defined out-of-band or in a seperate EPP extension.

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.

The XML namespace prefix "registry" is used for the namespace "urn:ietf:params:xml:ns:epp:registry-0.2", 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

An EPP registry object has attributes and associated values that may be viewed and modified by the sponsoring client or the server. This section describes each attribute type in detail. The formal syntax for the attribute values described here can be found in the "Formal Syntax" section of this document and in the appropriate normative references.

2.1. Zone Name

The zone name is an element that includes an OPTIONAL "form" attribute that defines the form of the zone name as either "aLabel" or "uLabel", with the default value of "aLabel". The "aLabel" form of a zone name contains all ASCII name labels that conform to [RFC0952] and [RFC1123]. The "uLabel" form of a zone name that includes one or more non-ASCII name labels that can be represented as ASCII labels using [RFC5890].

At the time of this writing, [RFC5890] describes a standard to use certain ASCII name labels to represent non-ASCII name labels. These conformance requirements might change in the future as a result of progressing work in developing standards for internationalized names.

2.2. Dates and Times

Date and time attribute values MUST be represented in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in XML Schema Part 2 [1] MUST be used to represent date-time values, as XML Schema does not support truncated date-time forms or lower case "T" and "Z" characters.

2.3. Schedule

A schedule is defined using the <registry:schedule> element, with the required "frequency" attribute that defines the frequency of execution. The "frequency" attribute has the possible values of "daily", "weekly", and "monthy". The time zone is defined using the XML schema "time" type conventions of UTC and offsets from UTC, or using the OPTIONAL "tz" attribute that defines the named time zone. For example, the named Eastern time zone can be specified using the setting "tz=EST5EDT".

When the "frequency" attribute is set to "weekly", the "dayOfWeek" attribute MUST be set with a value between 0 (Sunday) to 6 (Saturday), to define the day of week of execution.

When the "frequency" attribute is set to "monthy", the "dayOfMonth" attribute MUST be set with a value between 1 and 31, to define the day of month of execution. Execution will not occur in the current month if the "dayOfMonth" value is out-of-range for the current month (e.g, 29 - 31).

The following are examples of different <registry:schedule> element definitions:

Example daily schedule at 2 PM in the Eastern time zone:

```
<registry:schedule frequency="daily" tz="EST5EDT">
  14:00:00
</registry:schedule>
```

Example daily schedule at 5 PM EST (5 UTC offset):

```
<registry:schedule frequency="daily">
  07:00:00-05:00
</registry:schedule>
```

Example weekly schedule at midnight UTC on Sunday:

```
<registry:schedule frequency="weekly" dayOfWeek="0">
  00:00:00Z
</registry:schedule>
```

Example monthly schedule at 5 PM UTC on the 15th of the month:

```
<registry:schedule frequency="monthly" dayOfMonth="15">
  17:00:00Z
</registry:schedule>
```

2.4. Regular Expressions

A regular expression element contains a <registry:expression> child element that defines the regular expression to apply with an OPTIONAL <registry:description> child element that describes the regular expression with an OPTIONAL "lang" attribute that defines the language of the description, with a default value of "en" (English). The <registry:expression> element MUST conform to the Perl-compatible Regular Expression (PCRE) [pcre] syntax. Programming languages support different sets of PCRE features, so the server SHOULD define

a PCRE that leverages features that are supported by a broad set of client programming languages.

2.5. Zone Object

The Zone object, represented by the <registry:zone> element, is the primary object managed by this mapping. The Zone object can apply to any zone level (top level, second level, third level, etc.). The <registry:zone> element contains the following child elements:

<registry:name>: The zone name that can be at any level (top level, second level, third level, etc.), as described in Section 2.1.
<registry:group>: An OPTIONAL server defined grouping of zones where the zones belong to the same deployable unit.
<registry:services>: The OPTIONAL EPP namespace URIs of the objects and object extensions supported by the server based on [RFC5730]. The <registry:services> element contains the following child elements:

<registry:objURI>: One or more <registry:objURI> elements that contain namespace URIs representing the objects that the server is capable of managing for the zone with the required "required" attribute that defines whether the server requires the use of object represented by the URI.
<registry:svcExtension>: An OPTIONAL element that contains one or more <registry:extURI> elements that contain namespace URIs representing object extensions support by the server for the zone with the required "required" attribute that defines whether the server requires the use of the object extension represented by the URI.
<registry:crID>: The OPTIONAL identifier of the client that created the zone.
<registry:crDate>: The OPTIONAL date and time of zone object creation. The <registry:crDate> element MUST be set if the zone object has already been created.
<registry:upID>: The OPTIONAL identifier of the client that last updated the zone object. This element MUST NOT be present if the zone has never been modified.
<registry:upDate>: The OPTIONAL date and time of the most recent zone object modification. This element MUST NOT be present if the domain object has never been modified.
<registry:unsupportedData>: The OPTIONAL policy associated with receipt of unsupported data sent by the client to the server. The unsupported data may be an unsupported element or extension. The server SHOULD be consistent in the handling of unsupported data. The possible values for the <registry:unsupportedData> element include:

"fail": The server will fail the command that includes unsupported data.
"ignore": The server will ignore the unsupported data and execute the command.

<registry:batch>: The OPTIONAL list of batch jobs. The <registry:batch> element contains the following child elements:

<registry:batchJob>: One or more <registry:batchJob> elements containing the batch job information. The <registry:batchJob> element contains the following child elements:

<registry:name>: Name of the batch job, like "autoRenew" or "pendingDelete".
<registry:description>: OPTIONAL free-form description of the batch job, like "Auto Renew Batch" or "Pending Delete Batch".
<registry:schedule>: One or more <registry:schedule> elements, as defined in Section 2.3, that specifies when the batch job executes.

<registry:system>: The OPTIONAL list of zones that makeup the system when the "perSystem" share policy is used for the internal hosts, external hosts, or contacts. The list of zones are listed independent of the client's privileges to provision domains in the zone. The <registry:system> element contains the following child elements:

<registry:zone>: One or more <registry:zone> elements, as described in Section 2.1, containing the name of the zone that is a member of the system.

<registry:domain>: The domain name object policy information per [RFC5731]. The <registry:domain> element contains the following child elements:

<registry:domainName>: One or more <registry:domainName> that define the policies for a domain name label for a specific level, defined with the "level" attribute, with a minimum value of "2" for the second level domain name label level. The <registry:domainName> element contains the following child elements:

<registry:minLength>: An OPTIONAL minimum length of the domain name label.

<registry:maxLength>: An OPTIONAL maximum length of the domain name label.
<registry:alphaNumStart>: An OPTIONAL flag indicating whether the label must start with an alphanumeric character, with a default of "false".
<registry:alphaNumEnd>: An OPTIONAL flag indicating whether the label must end with an alphanumeric character, with a default value of "false".
<registry:aLabelSupported>: An OPTIONAL flag indicating whether ASCII domain names are supported with a default value of "true".
<registry:uLabelSupported>: An OPTIONAL flag indicating whether non-ASCII domain names are supported with a default value of "false".
<registry:nameRegex>: The OPTIONAL regular expression, as defined in Section 2.4, used to validate the domain name label.
<registry:reservedNames>: An OPTIONAL element that defines the set of reserved domain names starting from that label level. The reserved names can refer to values with more than one level which is relative to the level of the parent <registry:domainName> element. The <registry:reservedNames> element contains the following child elements:

<registry:reservedName>: Zero or more <registry:reservedName> elements containing a reserved domain name relative to the level of the parent <registry:domainName> element.
<registry:reservedNameURI>: An OPTIONAL URI to an externally defined list of reserved domain names relative to the level of the parent <registry:domainName> element.

<registry:idn>: The OPTIONAL Internationalized Domain Name (IDN) policy information. The <registry:idn> element contains the following child elements:

<registry:idnVersion>: The OPTIONAL server unique version of the IDN language rules.
<registry:idnaVersion>: An Internationalizing Domain Names in Applications (IDNA) version supported by the server. IDNA represents a collection of documents that describe the protocol and usage for Internationalized Domain for Applications like IDNA 2003, with value of 2003, or IDNA 2008, with value of 2008.

<registry:unicodeVersion>: The Unicode version supported by the server like the value of "6.0" for Unicode 6.0.
<registry:encoding>: The OPTIONAL encoding for transforming Unicode characters uniquely and reversibly into DNS compatible characters, with a default value of "Punycode".
<registry:commingleAllowed>: An OPTIONAL value that indicates whether commingling of scripts is allowed, with a default value of "false".
<registry:language>: Zero or more <registry:language> elements that defines the supported language codes and character code point policy. The required "code" attribute defines the language code for the supported language. The language code SHOULD be an ISO 639 (ISO 639-1 or ISO 639-2) value. The <registry:language> element contains the following child elements:

<registry:table>: The OPTIONAL language table URI that contains the set of code points for the language.
<registry:variantStrategy>: An OPTIONAL strategy for the handling of variants for the language. If no <registry:variantStrategy> element is specified then variants are not supported by the language. The possible values for the <registry:variantStrategy> element include:

"blocked": Variant registrations are blocked for all clients.
"restricted": Variant registrations are allowed for client of the original IDN registration.
"open": Variant registrations are open to all clients.

<registry:premiumSupport>: The OPTIONAL boolean value that indicates whether the server supports premium domain names, with a default value of "false".
<registry:contactsSupported>: The OPTIONAL boolean value that indicates whether contacts are supported, with a default value of "true".
<registry:contact>: Zero or more <registry:contact> elements that defines the minimum and maximum number of contacts by contact type. The contact type is defined with the required "type" attribute with the possible values of "admin", "tech", and "billing", and "custom". The OPTIONAL "name" attribute is an identifier, represented in the 7-bit US-ASCII character set, that is used to define the name of the "custom" type. If "custom" is the contact "type" value, then the "name"

attribute MUST be set. The OPTIONAL "description" attribute can be set with a description of the contact type. The <registry:contact> element contains the following child elements:

<registry:min>: The minimum number of contacts for the contact type.
<registry:max>: The OPTIONAL maximum number of contacts for the contact type. If the <registry:max> element is not defined the maximum number is unbounded. The <registry:max> element MUST NOT be less than the <registry:min> element.

<registry:ns>: Defines the minimum and maximum number of delegated host objects (name servers) that can be associated with a domain object. The <registry:ns> element contains the following child elements:

<registry:min>: The minimum number of name servers associated with a domain object.
<registry:max>: The OPTIONAL maximum number of name servers associated with a domain object. If the <registry:max> element is not defined the maximum number is unbounded. The <registry:max> element MUST NOT be less than the <registry:min> element.

<registry:childHost>: Defines the OPTIONAL minimum and maximum number of subordinate host objects (child hosts) for a domain object. This element is only applicable when using the host object model in [RFC5731]. The <registry:childHost> element contains the following child elements:

<registry:min>: The minimum number of child hosts for a domain object.
<registry:max>: The OPTIONAL maximum number of child hosts for a domain object. If the <registry:max> element is not defined the maximum number is unbounded. The <registry:max> element MUST NOT be less than the <registry:min> element.

<registry:period>: Zero or more <registry:period> elements that defines the supported registration periods and default periods by command type. The required "command" attribute defines the command type with sample values of "create", "renew", and "transfer". The <registry:period> element contains one of the following elements:

<registry:length>: The default, minimum, and maximum period length for the command type. The <registry:length> element contains the following child elements, where all of the child elements require the "unit" attribute with possible values of "y" for year and "m" for month:

<registry:min>: The minimum supported period length.
<registry:max>: The maximum supported period length. The <registry:max> element MUST NOT be less than the <registry:min> element.
<registry:default>: The default period length if not defined by the client.

or <registry:serverDecided>: The registration period is decided by the server based on the relationship to a related object that MUST have the same expiration date.

<registry:exceedMaxExDate>: Zero or more <registry:exceedMaxExDate> elements that defines the action taken by the server when executing commands that will result in an expiration date that exceeds the maximum expiration date. The required "command" attribute is used to define the command with a renewal feature, such as "renew" or "transfer". New commands can be defined that include a renewal feature, such as "sync". The possible values for the <registry:exceedMaxExDate> element include:

"fail": The server will fail the renewal command when the expiration date exceeds the maximum expiration date. An example is if the maximum expiration date is 10 years, and a client renews a domain name to 10.5 years, the server will fail the renew.
"clip": The server will clip the fractional period when the expiration date exceeds the maximum expiration date by a fraction of a period and will fail the renewal command when the expiration date exceeds the maximum expiration date by a whole period and above. An example is if the maximum expiration date is 10 years, and the client renews a domain to 10.5 years, the server will clip the .5 fractional year so that the domain name will expire exactly in 10 years.
"disableRenewal": The server will execute the command with the renewal feature disabled when the expiration date exceeds the maximum expiration date. This may be the case for a command like "transfer" that includes a renewal feature in [RFC5731].

<registry:transferHoldPeriod>: The period of time a domain object is in the pending transfer before the transfer is auto

approved by the server. The <registry:transferHoldPeriod> element MUST have the "unit" attribute with the possible values of "y" for year, "m" for month, and "d" for day.

<registry:gracePeriod>: Zero or more <registry:gracePeriod> elements that defines the grace periods by operation type. The required "command" attribute defines the operation type with the sample values of "create", "renew", "transfer", and "autoRenew". The <registry:gracePeriod> element requires the "unit" attribute with the possible values of "d" for day, "h" for hour, and "m" for minute.

<registry:rgp>: The OPTIONAL Registry Grace Period (RGP) status periods. The <registry:rgp> element contains the following child elements, where each child element supports the "unit" attribute with the possible values of "y" for year, "m" for month, "d" for day, and "h" for hour:

<registry:redemptionPeriod>: The length of time that a domain object will remain in the redemptionPeriod status unless the restore request command is received.

<registry:pendingRestore>: The length of time that the domain object will remain in the pendingRestore status unless the restore report command is received.

<registry:pendingDelete>: The length of time that the domain object will remain in the pendingDelete status prior to being purged.

<registry:dnssec>: The OPTIONAL DNS Security Extensions (DNSSEC) policies for the server. The <registry:dnssec> element contains the following child elements:

<registry:dsDataInterface>: Defines the DS Data Interface, as defined in [RFC5910], policies. The <registry:dsDataInterface> element contains the following child elements:

<registry:min>: The minimum number of DS associated with the domain object.

<registry:max>: The maximum number of DS associated with the domain object. The <registry:max> element MUST NOT be less than the <registry:min> element.

<registry:alg>: Zero or more <registry:alg> elements that define the supported algorithms as described in section 5.1.2 of [RFC4034].

<registry:digestType>: Zero or more <registry:digestType> elements that define the

supported digest types as described in section 5.1.3 of [RFC4034].

<registry:keyDataInterface>: Defines the Key Data Interface, as defined in [RFC5910], policies. The <registry:keyDataInterface> element contains the following child elements:

<registry:min>: The minimum number of keys associated with the domain object.

<registry:max>: The maximum number of keys associated with the domain object. The <registry:max> element MUST NOT be less than the <registry:min> element.

<registry:flags>: Zero or more <registry:flags> elements that define the supported flags field values, as described in section 2.1.1 of [RFC4034].

<registry:protocol>: Zero or more <registry:protocol> elements that define the supported protocols, as described in section 2.1.2 of [RFC4034].

<registry:alg>: Zero or more <registry:alg> elements that define the supported algorithms, as described in section 2.1.3 of [RFC4034].

<registry:maxSigLife>: Defines the maximum signature lifetime policies. The <registry:maxSigLife> element contains the following child elements:

<registry:clientDefined>: An OPTIONAL boolean flag indicating whether the client can set the maximum signature lifetime, with a default value of "false".

<registry:default>: The OPTIONAL default maximum signature lifetime set by the server.

<registry:min>: An OPTIONAL minimum signature lifetime supported. The <registry:min> element MUST NOT be defined if the <registry:clientDefined> element value is "false".

<registry:max>: An OPTIONAL maximum signature lifetime supported. The <registry:max> element MUST NOT be defined if the <registry:clientDefined> element value is "false". The <registry:max> element MUST NOT be less than the <registry:min> element.

<registry:urgent>: An OPTIONAL flag that of whether the client can specify the urgent attribute for DNSSEC updates, with a default value of "false".

<registry:maxCheckDomain>: The maximum number of domain names (<domain:name> elements) that can be included in a domain check command defined in [RFC5731].

<registry:supportedStatus>: The OPTIONAL set of supported domain statuses that SHOULD match the statuses defined in [RFC5731].
<registry:authInfoRegEx>: The OPTIONAL regular expression, as defined in Section 2.4, used to validate the domain object authorization information value.
<registry:expiryPolicy>: The OPTIONAL expiry policy used to define what happens when the domain object expires, with a default value of "autoRenew". The possible values for the <registry:expiryPolicy> element include:

"autoRenew": The domain object will auto-renew at expiry. The client can receive a credit for the auto-renew if the domain object is deleted or transferred within the auto-renew grace period.
"autoDelete": The domain object will auto-delete at expiry. The client needs to explicitly renew the domain object prior to its expiry to ensure that it does not get deleted.
"autoExpire": The domain object will auto-expire at expiry that may include the server placing the domain object on serverHold.
"autoParked": The domain object will be auto-parked at expiry that results in the resolution of the domain object going to a parked page.

<registry:nullAuthInfoSupported>: An OPTIONAL flag indicating whether the <domain:null> element in [RFC5731] is supported to remove the authorization information, with a default value of "false".
<registry:hostModelSupported>: The OPTIONAL definition of which [RFC5731] host model is used by the server. The possible values include "hostObj" for the host object model and "hostAttr" for the host attribute model, with the default value of "hostObj".

<registry:host>: The host object policy information per [RFC5732]. The <registry:host> element contains the following child elements:

<registry:internal>: Defines the minimum and maximum number of IP addresses supported for an internal host. The <registry:internal> elements contains the following child elements:

<registry:minIP>: Minimum number of IP addresses supported for an internal host.

<registry:maxIP>: Maximum number of IP addresses supported for an internal host. The <registry:maxIP> element MUST NOT be less than the <registry:minIP> element.
<registry:sharePolicy>: The OPTIONAL policy for the sharing of internal hosts in the server. The possible shared policy values include:

"perZone": The internal hosts are shared across all domains of the zone. There is a single pool of internal hosts defined for the zone.
"perSystem": The internal hosts are shared across all zones of the system. There is a single pool of internal hosts across all of the zones supported by the system. The system MUST be defined using the <registry:system> element.
<registry:uniqueIpAddressesRequired>: The OPTIONAL boolean value that indicates that all of the IP addresses for the host object must be unique, with a default value of "false".
<registry:external>: Defines the policies for external hosts. The <registry:external> elements contains the following child elements:

<registry:minIP>: Minimum number of IP addresses supported for an external host.
<registry:maxIP>: Maximum number of IP addresses supported for an external host. The <registry:maxIP> element MUST NOT be less than the <registry:minIP> element.
<registry:sharePolicy>: The OPTIONAL policy for the sharing of external hosts in the server. The possible shared policy values include:

"perRegistrar": The external hosts are shared across all domains of the registrar. There is a single pool of external hosts defined per registrar.
"perZone": The external hosts are shared across all domains of the zone. There is a single pool of external hosts defined for the zone.
"perSystem": The external hosts are shared across all zones of the system. There is a single pool of external hosts across all of the zones supported by the system. The system MUST be defined using the <registry:system> element.
<registry:uniqueIpAddressesRequired>: The OPTIONAL boolean value that indicates that all of the IP addresses for the

host object must be unique, with a default value of "false".

<registry:nameRegex>: The OPTIONAL regular expression, as defined in Section 2.4, used to validate the host name value.

<registry:maxCheckHost>: The OPTIONAL maximum number of host names (<host:name> elements) that can be included in a host check command defined in [RFC5732]. This element is only applicable when using the host object model in [RFC5731] and supporting host objects in [RFC5732].

<registry:supportedStatus>: The OPTIONAL set of supported host statuses that SHOULD match the statuses defined in [RFC5732].

<registry:invalidIP>: Zero or more <registry:invalidIP> elements that defines the URI of an externally defined list of invalid IP addresses. The IP addresses referenced by the list of <registry:invalidIP> elements should be combined and normalized by the client to define the complete set of invalid IP addresses.

<registry:contact>: The OPTIONAL contact object policy information per [RFC5733]. The <registry:contact> element contains the following child elements:

<registry:contactIdRegex>: The OPTIONAL regular expression, as defined in Section 2.4, used to validate the <contact:id> element defined in [RFC5733].

<registry:contactIdPrefix>: The OPTIONAL client-specific prefix that must be used for the <contact:id> element defined in [RFC5733]. For example, if the client is assigned the client-specific prefix "EX", every contact created by the client must have a <contact:id> element value prefixed with "EX", as in "EX123".

<registry:sharePolicy>: The OPTIONAL policy for the sharing of contacts in the server. The possible shared policy values include:

"perZone": The contacts are shared across all objects of the zone. There is a single pool of contacts defined for the zone.

"perSystem": The contacts are shared across all zones of the system. There is a single pool of contacts across all of the zones supported by the system. The system MUST be defined using the <registry:system> element.

<registry:postalInfoTypeSupport>: The policy associated with the postal-address information, represented by the <contact:postalInfo> element in [RFC5733], supported with the following possible values:

"loc": Indicates that a single <contact:postalInfo> element is supported with the type "loc".
"int": Indicates that a single <contact:postalInfo> element is supported with the type "int".
"locOrInt": Indicates that a single <contact:postalInfo> element is supported with the type "loc" or "int".
"locAndInt": Indicates that up to two <contact:postalInfo> elements is supported for defining both the "loc" and the "int" type. This policy does not indicate that both must be provided.
"intOptLoc": Indicates that the <contact:postalInfo> element with type "int" is required and a second <contact:postalInfo> element with the type "loc" is optional.
"locOptInt": Indicates that the <contact:postalInfo> element with type "loc" is required and a second <contact:postalInfo> element with the type "int" is optional.

<registry:postalInfo>: The postal-address information policy information. The <registry:postalInfo> element contains the following child elements:

<registry:locCharRegex>: The OPTIONAL regular expression , as defined in Section 2.4, that represents the character set that can be used for the <contact:postalInfo> localized form (type="loc") element content. The regular expression MUST be applicable to all <contact:postalInfo> element content.
<registry:name>: The minimum and maximum length of <contact:name> element defined [RFC5733] using the <registry:minLength> and <registry:maxLength> child elements, respectively.
<registry:org>: The minimum and maximum length of the <contact:org> element defined in [RFC5733] using the <registry:minLength> and <registry:maxLength> child elements, respectively.
<registry:address>: The address information policy information. The <registry:address> element contains the following child elements:

<registry:street>: The minimum and maximum length and the minimum and maximum number of the <contact:street> elements defined in [RFC5733]. The <registry:street> element contains the following child elements:

<registry:minLength>: The minimum length of the <contact:street> elements.
<registry:maxLength>: The maximum length of the <contact:street> elements. The <registry:maxLength> element MUST NOT be less than the <registry:minLength> element.
<registry:minEntry>: The minimum number of <contact:street> elements.
<registry:maxEntry>: The maximum number of <contact:street> elements. The <registry:maxEntry> element MUST NOT be less than the <registry:minEntry> element.
<registry:city>: The minimum and maximum length of the <contact:city> element defined in [RFC5733] using the <registry:minLength> and <registry:maxLength> child elements, respectively.
<registry:sp>: The minimum and maximum length of the <contact:sp> element defined in [RFC5733] using the <registry:minLength> and <registry:maxLength> child elements, respectively.
<registry:pc>: The minimum and maximum length of the <contact:pc> element defined in [RFC5733] using the <registry:minLength> and <registry:maxLength> child elements, respectively.
<registry:voiceRequired>: An OPTIONAL boolean flag indicating whether the server requires the <contact:voice> element to be defined, with a default value of "false".
<registry:voiceExt>: The OPTIONAL minimum and maximum length of the <contact:voice> extension "x" attribute defined in [RFC5733] using the <registry:minLength> and <registry:maxLength> child elements, respectively.
<registry:emailRegex>: An OPTIONAL <registry:emailRegex> element that defines the regular expression, as defined in Section 2.4, used to validate the <contact:email> in [RFC5733].
<registry:maxCheckContact>: The maximum number of contact identifiers (<contact:id> elements) that can be included in a contact check command defined in [RFC5733].
<registry:authInfoRegex>: The OPTIONAL regular expression, as defined in Section 2.4, used to validate the contact object authorization information value.
<registry:clientDisclosureSupported>: The OPTIONAL flag that indicates whether the server supports the client to identify elements that require exception server-operator handling to allow or restrict disclosure to third parties defined in [RFC5733] with a default of "false".

<registry:supportedStatus>: The OPTIONAL set of supported contact statuses that SHOULD match the statuses defined in [RFC5733].
<registry:transferHoldPeriod>: The OPTIONAL period of time a contact object is in the pending transfer before the transfer is auto approved by the server. The <registry:transferHoldPeriod> element MUST have the "unit" attribute with the possible values of "y" for year, "m" for month, and "d" for day.
<registry:privacyContactSupported>: An OPTIONAL boolean value that indicates whether a privacy contact is supported, with a default value of "true".
<registry:proxyContactSupported>: An OPTIONAL boolean value that indicates whether a proxy contact is supported, with a default value of "true".

Example of a <registry:zone> element:

```
<registry:zone>
  <registry:name>EXAMPLE</registry:name>
  <registry:group>STANDARD</registry:group>
  <registry:services>
    <registry:objURI required="true">
      urn:ietf:params:xml:ns:domain-1.0
    </registry:objURI>
    <registry:objURI required="true">
      urn:ietf:params:xml:ns:host-1.0
    </registry:objURI>
    <registry:objURI required="true">
      urn:ietf:params:xml:ns:contact-1.0
    </registry:objURI>
    <registry:svcExtension>
      <registry:extURI required="true">
        urn:ietf:params:xml:ns:rgp-1.0
      </registry:extURI>
      <registry:extURI required="true">
        urn:ietf:params:xml:ns:secDNS-1.1
      </registry:extURI>
      <registry:extURI required="true">
        http://www.verisign-grs.com/epp/namestoreExt-1.1
      </registry:extURI>
      <registry:extURI required="false">
        http://www.verisign.com/epp/idnLang-1.0
      </registry:extURI>
    </registry:svcExtension>
  </registry:services>
  <registry:crID>clientX</registry:crID>
  <registry:crDate>2012-10-01T00:00:00.0Z
```

```
    </registry:crDate>
    <registry:upID>clientY</registry:upID>
    <registry:upDate>2012-10-15T00:00:00.0Z
    </registry:upDate>
    <registry:unsupportedData>fail
    </registry:unsupportedData>
    <registry:batch>
      <registry:batchJob>
        <registry:name>localTzBatch</registry:name>
        <registry:description>
          Batch with multiple local time schedules (name and offset)
        </registry:description>
        <registry:schedule frequency="daily" tz="EST5EDT">
         04:00:00
        </registry:schedule>
        <registry:schedule frequency="daily">
         07:00:00-05:00
        </registry:schedule>
       </registry:batchJob>
      <registry:batchJob>
        <registry:name>multiBatchSchedule</registry:name>
        <registry:description>
          Batch with multiple UTC schedules
        </registry:description>
        <registry:schedule frequency="daily">
         12:00:00Z
        </registry:schedule>
        <registry:schedule frequency="weekly" dayOfWeek="0">
         00:00:00Z
        </registry:schedule>
        <registry:schedule frequency="monthly" dayOfMonth="15">
         17:00:00Z
        </registry:schedule>
       </registry:batchJob>
    </registry:batch>
    <registry:system>
      <registry:zone form="aLabel">EXAMPLE
      </registry:zone>
      <registry:zone form="aLabel">EXAMPLE2
      </registry:zone>
    </registry:system>
    <registry:domain>
      <registry:domainName level="2">
        <registry:minLength>5
        </registry:minLength>
        <registry:maxLength>50
        </registry:maxLength>
        <registry:alphaNumStart>true
```

```
        </registry:alphaNumStart>
        <registry:alphaNumEnd>false
        </registry:alphaNumEnd>
        <registry:aLabelSupported>true
        </registry:aLabelSupported>
        <registry:uLabelSupported>false
        </registry:uLabelSupported>
        <registry:nameRegex>
          <registry:expression>
          ^[a-zA-Z\d][a-zA-Z\d\-]{4,49}$
          </registry:expression>
          <registry:description>
            5 to 50 DNS characters starting with alphanumeric
          </registry:description>
        </registry:nameRegex>
        <registry:reservedNames>
          <registry:reservedName>reserved1
          </registry:reservedName>
        </registry:reservedNames>
      </registry:domainName>
      <registry:idn>
        <registry:idnVersion>4.1
        </registry:idnVersion>
        <registry:idnaVersion>2008
        </registry:idnaVersion>
        <registry:unicodeVersion>6.0
        </registry:unicodeVersion>
        <registry:encoding>Punycode
        </registry:encoding>
        <registry:commingleAllowed>false
        </registry:commingleAllowed>
        <registry:language code="LANG-1">
          <registry:table>
            http://www.iana.org/idn-tables/test_tab1_1.1.txt
          </registry:table>
          <registry:variantStrategy>blocked
          </registry:variantStrategy>
        </registry:language>
      </registry:idn>
      <registry:premiumSupport>false
      </registry:premiumSupport>
      <registry:contact type="admin">
        <registry:min>1</registry:min>
        <registry:max>1</registry:max>
      </registry:contact>
      <registry:contact type="tech">
        <registry:min>1</registry:min>
        <registry:max>1</registry:max>
```

```
      </registry:contact>
      <registry:contact type="billing">
        <registry:min>0</registry:min>
        <registry:max>0</registry:max>
      </registry:contact>
      <registry:contact
        type="custom"
        name="abuse"
        description="Abuse Contact"
      >
        <registry:min>0</registry:min>
        <registry:max>1</registry:max>
      </registry:contact>
      <registry:ns>
        <registry:min>0</registry:min>
        <registry:max>13</registry:max>
      </registry:ns>
      <registry:childHost>
        <registry:min>0</registry:min>
      </registry:childHost>
      <registry:period command="create">
        <registry:length>
          <registry:min unit="y">1</registry:min>
          <registry:max unit="y">10</registry:max>
          <registry:default unit="y">1</registry:default>
        </registry:length>
      </registry:period>
      <registry:exceedMaxExDate command="renew">
        fail
      </registry:exceedMaxExDate>
      <registry:exceedMaxExDate command="transfer">
        clip
      </registry:exceedMaxExDate>
      <registry:transferHoldPeriod unit="d">5
      </registry:transferHoldPeriod>
      <registry:gracePeriod
        command="create"
        unit="d"
      >5
      </registry:gracePeriod>
      <registry:gracePeriod
        command="renew"
        unit="d"
      >5
      </registry:gracePeriod>
      <registry:gracePeriod
        command="transfer"
        unit="d"
```

```
      >5
      </registry:gracePeriod>
      <registry:gracePeriod
        command="autoRenew"
        unit="d"
      >45
      </registry:gracePeriod>
      <registry:rgp>
        <registry:redemptionPeriod unit="d">30
        </registry:redemptionPeriod>
        <registry:pendingRestore unit="d">7
        </registry:pendingRestore>
        <registry:pendingDelete unit="d">5
        </registry:pendingDelete>
      </registry:rgp>
      <registry:dnssec>
        <registry:dsDataInterface>
          <registry:min>0</registry:min>
          <registry:max>13</registry:max>
          <registry:alg>3</registry:alg>
          <registry:digestType>1</registry:digestType>
        </registry:dsDataInterface>
        <registry:maxSigLife>
          <registry:clientDefined>false
          </registry:clientDefined>
        </registry:maxSigLife>
      </registry:dnssec>
      <registry:maxCheckDomain>5
      </registry:maxCheckDomain>
      <registry:supportedStatus>
        <registry:status>ok
        </registry:status>
        <registry:status>clientDeleteProhibited
        </registry:status>
        <registry:status>serverDeleteProhibited
        </registry:status>
        <registry:status>clientHold
        </registry:status>
        <registry:status>serverHold
        </registry:status>
        <registry:status>clientRenewProhibited
        </registry:status>
        <registry:status>serverRenewProhibited
        </registry:status>
        <registry:status>clientTransferProhibited
        </registry:status>
        <registry:status>serverTransferProhibited
        </registry:status>
```

```
        <registry:status>clientUpdateProhibited
        </registry:status>
        <registry:status>serverUpdateProhibited
        </registry:status>
        <registry:status>inactive
        </registry:status>
        <registry:status>pendingDelete
        </registry:status>
        <registry:status>pendingTransfer
        </registry:status>
      </registry:supportedStatus>
      <registry:authInfoRegex>
        <registry:expression>^.*$</registry:expression>
      </registry:authInfoRegex>
      <registry:expiryPolicy>autoRenew
      </registry:expiryPolicy>
      <registry:nullAuthInfoSupported>false
      </registry:nullAuthInfoSupported>
      <registry:hostModelSupported>hostObj
      </registry:hostModelSupported>
    </registry:domain>
    <registry:host>
      <registry:internal>
        <registry:minIP>1</registry:minIP>
        <registry:maxIP>13</registry:maxIP>
        <registry:sharePolicy>perSystem
        </registry:sharePolicy>
        <registry:uniqueIpAddressesRequired>false
        </registry:uniqueIpAddressesRequired>
      </registry:internal>
      <registry:external>
        <registry:minIP>0</registry:minIP>
        <registry:maxIP>0</registry:maxIP>
        <registry:sharePolicy>perSystem
        </registry:sharePolicy>
      </registry:external>
      <registry:nameRegex>
        <registry:expression>^.*$
        </registry:expression>
      </registry:nameRegex>
      <registry:maxCheckHost>5
      </registry:maxCheckHost>
      <registry:supportedStatus>
        <registry:status>ok</registry:status>
        <registry:status>clientDeleteProhibited
        </registry:status>
        <registry:status>serverDeleteProhibited
        </registry:status>
```

```
        <registry:status>clientUpdateProhibited
        </registry:status>
        <registry:status>serverUpdateProhibited
        </registry:status>
        <registry:status>linked
        </registry:status>
        <registry:status>pendingDelete
        </registry:status>
        <registry:status>pendingTransfer
        </registry:status>
      </registry:supportedStatus>
      <registry:invalidIP>http://www.example.com/invalidip-1.txt
      </registry:invalidIP>
      <registry:invalidIP>http://www.example.com/invalidip-2.txt
      </registry:invalidIP>
    </registry:host>
    <registry:contact>
      <registry:contactIdRegex>
        <registry:expression>^.*$
        </registry:expression>
      </registry:contactIdRegex>
      <registry:contactIdPrefix>EX
      </registry:contactIdPrefix>
      <registry:sharePolicy>perZone
      </registry:sharePolicy>
      <registry:postalInfoTypeSupport>locOrInt
      </registry:postalInfoTypeSupport>
      <registry:postalInfo>
        <registry:locCharRegex>
          <registry:expression>^.*$
          </registry:expression>
        </registry:locCharRegex>
        <registry:name>
          <registry:minLength>5</registry:minLength>
          <registry:maxLength>15</registry:maxLength>
        </registry:name>
        <registry:org>
          <registry:minLength>2</registry:minLength>
          <registry:maxLength>40</registry:maxLength>
        </registry:org>
        <registry:address>
          <registry:street>
            <registry:minLength>1</registry:minLength>
            <registry:maxLength>40</registry:maxLength>
            <registry:minEntry>1</registry:minEntry>
            <registry:maxEntry>3</registry:maxEntry>
          </registry:street>
          <registry:city>
```

```
            <registry:minLength>1</registry:minLength>
            <registry:maxLength>40</registry:maxLength>
          </registry:city>
          <registry:sp>
            <registry:minLength>1</registry:minLength>
            <registry:maxLength>40</registry:maxLength>
          </registry:sp>
          <registry:pc>
            <registry:minLength>1</registry:minLength>
            <registry:maxLength>40</registry:maxLength>
          </registry:pc>
        </registry:address>
        <registry:voiceRequired>false
        </registry:voiceRequired>
        <registry:voiceExt>
          <registry:minLength>1</registry:minLength>
          <registry:maxLength>40</registry:maxLength>
        </registry:voiceExt>
        <registry:faxExt>
          <registry:minLength>1</registry:minLength>
          <registry:maxLength>40</registry:maxLength>
        </registry:faxExt>
        <registry:emailRegex>
          <registry:expression>^.+\..+$
          </registry:expression>
        </registry:emailRegex>
      </registry:postalInfo>
      <registry:maxCheckContact>5</registry:maxCheckContact>
      <registry:authInfoRegex>
        <registry:expression>^.*$</registry:expression>
      </registry:authInfoRegex>
      <registry:clientDisclosureSupported>false
      </registry:clientDisclosureSupported>
      <registry:supportedStatus>
        <registry:status>ok
        </registry:status>
        <registry:status>clientDeleteProhibited
        </registry:status>
        <registry:status>serverDeleteProhibited
        </registry:status>
        <registry:status>clientTransferProhibited
        </registry:status>
        <registry:status>serverTransferProhibited
        </registry:status>
        <registry:status>clientUpdateProhibited
        </registry:status>
        <registry:status>serverUpdateProhibited
        </registry:status>
```

```
        <registry:status>linked
        </registry:status>
        <registry:status>pendingDelete
        </registry:status>
        <registry:status>pendingTransfer
        </registry:status>
      </registry:supportedStatus>
      <registry:transferHoldPeriod unit="d">5
      </registry:transferHoldPeriod>
      <registry:privacyContactSupported>true
      </registry:privacyContactSupported>
      <registry:proxyContactSupported>true
      </registry:proxyContactSupported>
    </registry:contact>
  </registry:zone>
```

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 provisioning and managing TLD names via EPP.

3.1. EPP Query Commands

EPP [RFC5730] provides three commands to retrieve object information: <check> to determine if an object is known to the server, <info> to retrieve detailed information associated with an object, and <transfer> to retrieve object transfer status information.

3.1.1. EPP <check> Command

The EPP <check> command is used to determine if the server currently supports a zone. If the response indicates that the zone is not available, then it is currently supported; otherwise it MAY be available to be created by an authorized client.

In addition to the standard EPP command elements, the <check> command MUST contain a <registry:check> element that identifies the registry namespace. The <registry:check> element contains the following child elements:

<registry:name>: One or more <registry:name> elements, as described in Section 2.1, that contain the fully qualified names of the zone objects to be queried.

Example <check> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <registry:check
C:       xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
C:        <registry:name>EXAMPLE1</registry:name>
C:        <registry:name>EXAMPLE2</registry:name>
C:        <registry:name>EXAMPLE3</registry:name>
C:      </registry:check>
C:    </check>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <check> command has been processed successfully, the EPP <resData> element MUST contain a child <registry:chkData> element that identifies the registry namespace. The <registry:chkData> element contains one or more <registry:cd> elements that contain the following child elements:

<registry:name>: element that contains the fully qualified name of the queried zone object, as described in Section 2.1. This element MUST contain an "avail" attribute whose value indicates zone is currently supported or availability at the moment the <check> command was completed for an authorized client. A value of "1" or "true" means that the zone object is available for an authorized client. A value of "0" or "false" means that the zone object is currently supported by the server.

<registry:reason>: The OPTIONAL element that MAY be provided when a zone object is not available for provisioning. If present, this element contains server-specific text to help explain why the zone object is unavailable. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute MAY be present to identify the language if the negotiated value is something other than a default value of "en" (English).

Example <check> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <registry:chkData
S:        xmlns:registry=
S:        "urn:ietf:params:xml:ns:epp:registry-0.2">
S:        <registry:cd>
S:          <registry:name avail="0">EXAMPLE1</registry:name>
S:          <registry:reason>Client not authorized
S:          </registry:reason>
S:        </registry:cd>
S:        <registry:cd>
S:          <registry:name avail="0">EXAMPLE2
S:          </registry:name>
S:          <registry:reason>Already supported
S:          </registry:reason>
S:        </registry:cd>
S:        <registry:cd>
S:          <registry:name avail="1">EXAMPLE3
S:          </registry:name>
S:        </registry:cd>
S:      </registry:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.1.2. EPP <info> Command

The EPP <info> command is used to retrieve information associated with a zone object. The response to this command MAY vary depending on the identity of the querying client, use of authorization information, and server policy towards unauthorized clients. Server policy determines which OPTIONAL elements are returned.

In addition to the standard EPP command elements, the <info> command MUST contain a <registry:info> element that identifies the registry namespace. The <registry:info> element contains one of the following three child elements:

<registry:all>: Element that is empty and that indicates to return the client accessible and/or available zone objects with a summary set of attributes per zone object. The scope of the zones to return is defined by the "scope" attribute, with the possible values of "accessible" to indicate the zones that are assessible to the client, "available" to indicate the zones that are not accessible to the client but available on the server, and "both" to indicate both accessible and available zones. The default value for the "scope" attribute is "accessible". It is up to server policy what available zones the client is authorized to get information for.

<registry:name>: Element that contains the fully qualified name of the zone object, as described in Section 2.1, to be queried for a full set of attributes for the zone object.

<registry:system>: Element that is empty and that indicates that the registry system attributes, like maximum connections and timeouts, are queried.

Example <info> command to query for a summary set of attributes for all of the accessible and available zone objects:

```
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:      <registry:info
C:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
C:        <registry:all scope="both"/>
C:      </registry:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> command to query for the full set of "EXAMPLE" zone object attributes:

```
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:      <registry:info
C:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
C:        <registry:name>EXAMPLE</registry:name>
C:      </registry:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> command to query for registry system attributes:

```
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:      <registry:info
C:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
C:        <registry:system/>
C:      </registry:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <registry:infData> element that identifies the registry namespace. The <registry:infData> element contains one of the three following child elements:

<registry:zoneList>: Element that contains the list of <registry:zone> elements representing the zones accessible or available to the client with a set of summary attributes per zone. It is up to server policy what available zones the client is authorized to get information for. The <registry:zone> element includes the boolean "accessible" attribute that indicates whether the zone is accessible to the client, with a default value of "true". If the "accessible" attribute value is "false", the client does not have access to the zone, but the zone is available on the server. The <registry:zone> element contains the following child elements:

<registry:name>: Element that contains the fully qualified name of the queried zone object, as described in Section 2.1.
<registry:crDate>: The date and time of zone object creation.
<registry:upDate>: The OPTIONAL date and time of the most recent zone object modification. This element MUST NOT be present if the zone object has never been modified.

<registry:zone>: Element that contains the full set of attributes for the zone name as defined in Section 2.5. The <registry:zone> element includes the boolean "accessible" attribute that indicates whether the zone is accessible to the client, with a default value of "true". If the "accessible" attribute value is "false", the client does not have access to the zone, but the zone is available on the server.

<registry:system>: Element that contains registry system attributes. The <registry:system> element contains the following child elements:

<registry:maxConnections>: The OPTIONAL element that contains the maximum number of connections that the client can establish with the registry system.
<registry:idleTimeout>: The OPTIONAL element that contains the idle timeout for a connection in milliseconds. If a connection does not receive a command within <registry:idleTimeout> milliseconds, the server will close the connection.
<registry:absoluteTimeout>: The OPTIONAL element that contains the absolute timeout for a connection in milliseconds. The absolute timeout represents the maximum duration in milliseconds that a connection can be established. The server will close a connection that has been established for more than <registry:absoluteTimeout> milliseconds.
<registry:commandTimeout>: The OPTIONAL element that contains the command timeout for a connection in milliseconds. The server will close a connection that has an active command that exceeds <registry:commandTimeout> milliseconds.
<registry:transLimit>: The OPTIONAL element that contains the maximum number of transactions that can be submitted on the connection per the "perMs" attribute milliseconds. It is up to server policy what to do with the connection when the client exceeds the <registry:transLimit>.

Example <info> response to a query for a summary of all of the supported zone objects:

```
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:      <registry:infData
S:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
S:        <registry:zoneList>
S:          <registry:zone accessible="true">
S:            <registry:name>EXAMPLE1</registry:name>
S:            <registry:crDate>2012-10-01T00:00:00.0Z
S:            </registry:crDate>
S:            <registry:upDate>2012-10-15T00:00:00.0Z
S:            </registry:upDate>
S:          </registry:zone>
S:          <registry:zone accessible="false">
S:            <registry:name>EXAMPLE2</registry:name>
S:            <registry:crDate>2012-09-01T00:00:00.0Z
S:            </registry:crDate>
S:            <registry:upDate>2012-09-19T00:00:00.0Z
S:            </registry:upDate>
S:          </registry:zone>
S:        </registry:zoneList>
S:      </registry:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> response to query for the full set of "EXAMPLE" zone object attributes:

```
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:      <registry:infData
S:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
S:        <registry:zone accessible="true">
S:          <registry:name>EXAMPLE</registry:name>
S:          ...
S:        </registry:zone>
S:      </registry:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> response to query for the registry system attributes:

```
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:      <registry:infData
S:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
S:        <registry:system>
S:          <registry:maxConnections>200
S:          </registry:maxConnections>
S:          <registry:idleTimeout>600000
S:          </registry:idleTimeout>
S:          <registry:absoluteTimeout>86400000
S:          </registry:absoluteTimeout>
S:          <registry:commandTimeout>10000
S:          </registry:commandTimeout>
S:          <registry:transLimit perMs="1000">10
S:          </registry:transLimit>
S:       </registry:system>
S:      </registry:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.1.3. EPP <transfer> Query Command

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

3.2. EPP Transform Commands

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

3.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create a zone object. In addition to the standard EPP command elements, the <create> command MUST contain a <registry:create> element that identifies the registry namespace. The <registry:create> element contains the following child elements:

<registry:zone>: Element that contains the full set of attributes for the zone to create, as defined in Section 2.5.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
C:  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
C:  <command>
C:    <create>
C:      <registry:create
C:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
C:        <registry:zone>
C:          <registry:name>EXAMPLE</registry:name>
C:          ...
C:        </registry:zone>
C:      </registry:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <create> command has been processed successfully, the EPP <resData> element MUST contain a child <registry:creData> element that identifies the registry namespace. The <registry:creData> element contains the following child elements:

<registry:name>: element that contains the fully qualified name of the zone object, as described in Section 2.1.
<registry:crDate>: element that contains the date and time of zone object creation.

Example <create> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <registry:creData
S:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
S:        <registry:name>EXAMPLE</registry:name>
S:        <registry:crDate>2012-10-30T22:00:00.0Z
S:        </registry:crDate>
S:      </registry:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

An EPP error response MUST be returned if a <create> command can not be processed for any reason.

3.2.2. EPP <delete> Command

The EPP <delete> command provides a transform operation that allows a client to delete a zone object. In addition to the standard EPP command elements, the <delete> command MUST contain a <registry:delete> element that identifies the registry namespace. The <registry:delete> element contains the following child elements:

<registry:name>: element that contains the fully qualified name of the zone object to be deleted, as described in Section 2.1.

Example <delete> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <delete>
C:      <registry:delete
C:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
C:        <registry:name>EXAMPLE</registry:name>
C:      </registry:delete>
C:    </delete>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <delete> zone has been processed successfully, a server MUST respond with an EPP response with no <resData> element.

Example <delete> response:

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

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

3.2.3. EPP <renew> Command

Renew semantics do not directly apply to zone objects, so there is no mapping defined for the EPP <renew> command.

3.2.4. EPP <transfer> Command

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

3.2.5. EPP <update> Command

The EPP <update> command provides a transform operation that allows a client to modify the attributes of a zone object. In addition to the standard EPP command elements, the <update> command MUST contain a <registry:update> element that identifies the registry namespace. The <registry:update> element contains the following child elements:

<registry:zone>: One or more elements that contain the full set of attributes for the zones as defined in Section 2.5. The update completely replaces the prior version of the zone.

Example <update> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <registry:update
C:        xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2">
C:        <registry:zone>
C:          <registry:name>EXAMPLE</registry:name>
C:          ...
C:        </registry:zone>
C:      </registry:update>
C:    </update>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When an <update> command has been processed successfully, a server MUST respond with an EPP response with no <resData> element.

Example <update> command:

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

An EPP error response MUST be returned if an <update> command can not be processed for any reason.

4. Formal Syntax

One schema is presented here that is the EPP Registry 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. Registry Mapping Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema
  xmlns:registry="urn:ietf:params:xml:ns:epp:registry-0.2"
  xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
  xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:ietf:params:xml:ns:epp:registry-0.2"
  elementFormDefault="qualified"
>
  <!--
    Import common element types.
  -->
  <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>
  <import namespace="urn:ietf:params:xml:ns:epp-1.0"/>

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Registry
      Mapping Schema.
    </documentation>
  </annotation>
  <!--
    Child elements found in EPP commands.
  -->
  <element
    name="check"
    type="registry:mNameType"/>
  <element
    name="create"
    type="registry:createType"/>
  <element
```

```
      name="delete"
      type="registry:sNameType"/>
    <element
      name="info"
      type="registry:infoType"/>
    <element
      name="update"
      type="registry:updateType"/>
    <!--
      Child elements of the <check> command.
    -->
    <complexType name="mNameType">
      <sequence>
        <element
          name="name"
          type="registry:zoneNameType"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <!--
      Child elements of the <delete> command.
    -->
    <complexType name="sNameType">
      <sequence>
        <element
          name="name"
          type="registry:zoneNameType"/>
      </sequence>
    </complexType>
    <!--
      Child elements of the <create> command.
    -->
    <complexType name="createType">
      <sequence>
        <element
          name="zone"
          type="registry:zoneType"/>
      </sequence>
    </complexType>
    <complexType name="updateType">
      <sequence>
        <element
          name="zone"
          type="registry:zoneType"/>
      </sequence>
    </complexType>
    <!--
      Child elements of the <info> command.
```

```
    -->
    <complexType name="infoType">
      <sequence>
        <choice>
          <element name="all">
            <complexType>
              <attribute
                name="scope"
                default="accessible">
                <simpleType>
                  <restriction base="token">
                    <enumeration value="accessible"/>
                    <enumeration value="available"/>
                    <enumeration value="both"/>
                  </restriction>
                </simpleType>
              </attribute>
            </complexType>
          </element>
          <element
            name="name"
            type="registry:zoneNameType"/>
          <element name="system">
            <complexType/>
          </element>
        </choice>
      </sequence>
    </complexType>

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

    <!--
      <create> response elements.
    -->
    <complexType name="creDataType">
      <sequence>
        <element
```

```
          name="name"
          type="registry:zoneNameType"/>
        <element
          name="crDate"
          type="dateTime"/>
      </sequence>
    </complexType>
    <!--
      <check> response elements.
    -->
    <complexType name="chkDataType">
      <sequence>
        <element
          name="cd"
          type="registry:checkType"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <complexType name="checkType">
      <sequence>
        <element
          name="name"
          type="registry:checkNameType"/>
        <element
          name="reason"
          type="eppcom:reasonType"
          minOccurs="0"/>
      </sequence>
    </complexType>
    <complexType name="checkNameType">
      <simpleContent>
        <extension base="registry:zoneNameType">
          <attribute
            name="avail"
            type="boolean"
            use="required"/>
        </extension>
      </simpleContent>
    </complexType>
    <!--
      <info> response elements.
    -->
    <complexType name="infDataType">
      <choice>
        <element
          name="zoneList"
          type="registry:zoneListType"/>
        <element
```

```
          name="zone"
          type="registry:zoneInfDataType"/>
        <element
          name="system"
          type="registry:systemType"/>
      </choice>
    </complexType>
    <complexType name="zoneListType">
      <sequence>
        <element
          name="zone"
          type="registry:zoneSummaryType"
          minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <complexType name="zoneSummaryType">
      <sequence>
        <element
          name="name"
          type="registry:zoneNameType"/>
        <element
          name="crDate"
          type="dateTime"/>
        <element
          name="upDate"
          type="dateTime"
          minOccurs="0"/>
      </sequence>
      <attribute
        name="accessible"
        type="boolean"
        default="true"/>
    </complexType>
    <complexType name="zoneType">
      <sequence>
        <element
          name="name"
          type="registry:zoneNameType"/>
        <element
          name="group"
          type="token"
          minOccurs="0"/>
        <element
          name="services"
          type="registry:servicesType"
          minOccurs="0"/>
        <element
```

```
          name="crID"
          type="eppcom:clIDType"
          minOccurs="0"/>
        <element
          name="crDate"
          type="dateTime"
          minOccurs="0"/>
        <element
          name="upID"
          type="eppcom:clIDType"
          minOccurs="0"/>
        <element
          name="upDate"
          type="dateTime"
          minOccurs="0"/>
        <element
          name="unsupportedData"
          type="registry:unsupportedDataType"
          minOccurs="0"/>
        <element
          name="batch"
          type="registry:batchType"
          minOccurs="0"/>
        <element
          name="system"
          type="registry:zoneSystemType"
          minOccurs="0"/>
        <element
          name="domain"
          type="registry:domainType"/>
        <element
          name="host"
          type="registry:hostType"/>
        <element
          name="contact"
          type="registry:contactType"
          minOccurs="0"/>
      </sequence>
    </complexType>
    <complexType name="zoneInfDataType">
      <complexContent>
        <extension base="registry:zoneType">
          <attribute
            name="accessible"
            type="boolean"
            default="true"/>
        </extension>
      </complexContent>
```

```
    </complexType>
    <complexType name="fieldsType">
      <sequence>
        <element
          name="field"
          type="token"
          maxOccurs="unbounded"/>
      </sequence>
      <attribute
        name="type"
        use="required"
      >
        <simpleType>
          <restriction base="token">
            <enumeration value="shared"/>
            <enumeration value="sync"/>
          </restriction>
        </simpleType>
      </attribute>
    </complexType>
    <complexType name="servicesType">
      <sequence>
        <element
          name="objURI"
          type="registry:uriType"
          maxOccurs="unbounded"/>
        <element
          name="svcExtension"
          type="registry:svcExtensionType"
          minOccurs="0"/>
      </sequence>
    </complexType>
    <complexType name="svcExtensionType">
      <sequence>
        <element
          name="extURI"
          type="registry:uriType"
          minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <complexType name="uriType">
      <simpleContent>
        <extension base="anyURI">
          <attribute
            name="required"
            type="boolean"
            use="required"/>
```

```
        </extension>
      </simpleContent>
    </complexType>
    <complexType name="reservedNamesType">
      <choice>
        <element
          name="reservedName"
          type="normalizedString"
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="reservedNameURI"
          type="anyURI"
          minOccurs="0"/>
      </choice>
    </complexType>
    <complexType name="domainNameType">
      <sequence>
        <element
          name="minLength"
          type="unsignedShort"
          minOccurs="0"/>
        <element
          name="maxLength"
          type="unsignedShort"
          minOccurs="0"/>
        <element
          name="alphaNumStart"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="alphaNumEnd"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="aLabelSupported"
          type="boolean"
          minOccurs="0"
          default="true"/>
        <element
          name="uLabelSupported"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="nameRegex"
```

```
          type="registry:regexType"
          minOccurs="0"/>
        <element
          name="reservedNames"
          type="registry:reservedNamesType"
          minOccurs="0"/>
      </sequence>
      <attribute
        name="level"
        use="required"
      >
        <simpleType>
          <restriction base="unsignedShort">
            <minInclusive value="2"/>
          </restriction>
        </simpleType>
      </attribute>
    </complexType>
    <complexType name="regexType">
      <sequence>
        <element
          name="expression"
          type="string"/>
        <element
          name="description"
          minOccurs="0"
        >
          <complexType>
            <simpleContent>
              <extension base="normalizedString">
                <attribute
                  name="lang"
                  type="language"
                  default="en"/>
              </extension>
            </simpleContent>
          </complexType>
        </element>
      </sequence>
    </complexType>
    <simpleType name="zoneFormType">
      <restriction base="token">
        <enumeration value="aLabel"/>
        <enumeration value="uLabel"/>
      </restriction>
    </simpleType>
    <complexType name="zoneNameType">
      <simpleContent>
```

```
        <extension base="eppcom:labelType">
          <attribute
            name="form"
            type="registry:zoneFormType"
            default="aLabel"/>
        </extension>
      </simpleContent>
    </complexType>
    <simpleType name="variantStrategyType">
      <restriction base="token">
        <enumeration value="blocked"/>
        <enumeration value="restricted"/>
        <enumeration value="open"/>
      </restriction>
    </simpleType>
    <complexType name="languageType">
      <sequence>
        <element
          name="table"
          type="anyURI"
          minOccurs="0"/>
        <element
          name="variantStrategy"
          type="registry:variantStrategyType"
          minOccurs="0"/>
      </sequence>
      <attribute
        name="code"
        type="language"
        use="required"/>
    </complexType>
    <complexType name="idnType">
      <sequence>
        <element
          name="idnVersion"
          type="token"
          minOccurs="0"/>
        <element
          name="idnaVersion"
          type="token"/>
        <element
          name="unicodeVersion"
          type="token"/>
        <element
          name="encoding"
          type="token"
          minOccurs="0"
          default="Punycode"/>
```

```
        <element
          name="commingleAllowed"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="language"
          type="registry:languageType"
          minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <complexType name="dContactType">
      <complexContent>
        <extension base="registry:minMaxType">
          <attribute
            name="type"
            use="required"
          >
            <simpleType>
              <restriction base="token">
                <enumeration value="admin"/>
                <enumeration value="tech"/>
                <enumeration value="billing"/>
                <enumeration value="custom"/>
              </restriction>
            </simpleType>
          </attribute>
          <attribute
            name="name"
            type="token"/>
          <attribute
            name="description"
            type="token"/>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="minMaxType">
      <sequence>
        <element
          name="min"
          type="unsignedShort"/>
        <element
          name="max"
          type="unsignedShort"
          minOccurs="0"/>
      </sequence>
    </complexType>
```

```
    <complexType name="minMaxPeriod">
      <sequence>
        <element
          name="min"
          type="registry:periodType"/>
        <element
          name="max"
          type="registry:periodType"/>
        <element
          name="default"
          type="registry:periodType"/>
      </sequence>
    </complexType>
    <complexType name="dPeriodType">
      <choice>
        <element
          name="length"
          type="registry:minMaxPeriod"/>
        <element name="serverDecided">
          <complexType/>
        </element>
      </choice>
      <attribute
        name="command"
        type="token"
        use="required"/>
    </complexType>
    <complexType name="gPeriodType">
      <simpleContent>
        <extension base="registry:periodType">
          <attribute
            name="command"
            type="token"
            use="required"/>
        </extension>
      </simpleContent>
    </complexType>
    <complexType name="periodType">
      <simpleContent>
        <extension base="unsignedShort">
          <attribute
            name="unit"
            type="registry:pUnitType"
            use="required"/>
        </extension>
      </simpleContent>
    </complexType>
    <simpleType name="pUnitType">
```

```
      <restriction base="token">
        <enumeration value="y"/>
        <enumeration value="m"/>
        <enumeration value="d"/>
        <enumeration value="h"/>
      </restriction>
    </simpleType>
    <simpleType name="exceedMaxExDateEnumType">
      <restriction base="token">
        <enumeration value="fail"/>
        <enumeration value="clip"/>
        <enumeration value="disableRenewal"/>
      </restriction>
    </simpleType>
    <complexType name="exceedMaxExDateType">
      <simpleContent>
        <extension base="registry:exceedMaxExDateEnumType">
          <attribute
            name="command"
            type="token"
            use="required"/>
        </extension>
      </simpleContent>
    </complexType>
    <complexType name="rgpType">
      <sequence>
        <element
          name="redemptionPeriod"
          type="registry:periodType"/>
        <element
          name="pendingRestore"
          type="registry:periodType"/>
        <element
          name="pendingDelete"
          type="registry:periodType"/>
      </sequence>
    </complexType>
    <complexType name="keyInterfaceType">
      <sequence>
        <element
          name="min"
          type="unsignedShort"/>
        <element
          name="max"
          type="unsignedShort"/>
        <element
          name="flags"
          type="unsignedShort"
```

```
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="protocol"
          type="unsignedByte"
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="alg"
          type="token"
          minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <complexType name="dsInterfaceType">
    <sequence>
        <element
          name="min"
          type="unsignedShort"/>
        <element
          name="max"
          type="unsignedShort"/>
        <element
          name="alg"
          type="token"
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="digestType"
          type="token"
          minOccurs="0"
          maxOccurs="unbounded"/>
    </sequence>
    </complexType>
    <complexType name="maxSigLifeType">
      <sequence>
        <element
          name="clientDefined"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="default"
          type="int"
          minOccurs="0"/>
        <element
          name="min"
          type="int"
```

```
          minOccurs="0"/>
        <element
          name="max"
          type="int"
          minOccurs="0"/>
      </sequence>
    </complexType>
    <complexType name="dnssecType">
      <sequence>
        <choice>
          <element
            name="dsDataInterface"
            type="registry:dsInterfaceType"/>
          <element
            name="keyDataInterface"
            type="registry:keyInterfaceType"/>
        </choice>
        <element
          name="maxSigLife"
          type="registry:maxSigLifeType"/>
        <element
          name="urgent"
          type="boolean"
          minOccurs="0"
          default="false"/>
      </sequence>
    </complexType>
    <complexType name="supportedStatusType">
      <sequence>
        <element
          name="status"
          type="token"
          minOccurs="1"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <complexType name="scheduleType">
      <simpleContent>
        <extension base="time">
          <attribute
            name="frequency"
            use="required"
          >
            <simpleType>
              <restriction base="token">
                <enumeration value="daily"/>
                <enumeration value="weekly"/>
                <enumeration value="monthly"/>
```

```
            </restriction>
          </simpleType>
        </attribute>
        <attribute
          name="dayOfWeek"
        >
          <simpleType>
            <restriction base="byte">
              <minInclusive value="0"/>
              <maxInclusive value="6"/>
            </restriction>
          </simpleType>
        </attribute>
        <attribute
          name="dayOfMonth"
        >
          <simpleType>
            <restriction base="byte">
              <minInclusive value="1"/>
              <maxInclusive value="31"/>
            </restriction>
          </simpleType>
        </attribute>
        <attribute
          name="tz"
          type="token"/>
      </extension>
    </simpleContent>
  </complexType>
  <complexType name="batchJobType">
    <sequence>
      <element
        name="name"
        type="token"/>
      <element
        name="description"
        type="token"
        minOccurs="0"/>
      <!-- UNIX crontab job schedule format -->
      <element
        name="schedule"
        type="registry:scheduleType"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <!-- Information about the batch jobs -->
  <complexType name="batchType">
    <sequence>
```

```
        <element
          name="batchJob"
          type="registry:batchJobType"
          minOccurs="1"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <!--
      Information the TLDs that makeup the system, which is associated
      with the "perSystem" sharePolicy.
    -->
    <complexType name="zoneSystemType">
      <sequence>
        <element
          name="zone"
          type="registry:zoneNameType"
          minOccurs="1"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <simpleType name="expiryPolicyType">
      <restriction base="token">
        <enumeration value="autoRenew"/>
        <enumeration value="autoDelete"/>
        <enumeration value="autoExpire"/>
        <enumeration value="autoParked"/>
      </restriction>
    </simpleType>
    <complexType name="domainType">
      <sequence>
        <element
          name="domainName"
          type="registry:domainNameType"
          maxOccurs="unbounded"/>
        <element
          name="idn"
          type="registry:idnType"
          minOccurs="0"/>
        <element
          name="premiumSupport"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="contactsSupported"
          type="boolean"
          minOccurs="0"
          default="true"/>
```

```
        <element
          name="contact"
          type="registry:dContactType"
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="ns"
          type="registry:minMaxType"/>
        <element
          name="childHost"
          type="registry:minMaxType"
          minOccurs="0"/>
        <element
          name="period"
          type="registry:dPeriodType"
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="exceedMaxExDate"
          type="registry:exceedMaxExDateType"
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="transferHoldPeriod"
          type="registry:periodType"/>
        <element
          name="gracePeriod"
          type="registry:gPeriodType"
          minOccurs="0"
          maxOccurs="unbounded"/>
        <element
          name="rgp"
          type="registry:rgpType"
          minOccurs="0"/>
        <element
          name="dnssec"
          type="registry:dnssecType"
          minOccurs="0"/>
        <element
          name="maxCheckDomain"
          type="unsignedShort"/>
        <element
          name="supportedStatus"
          type="registry:supportedStatusType"
          minOccurs="0"/>
        <element
          name="authInfoRegex"
          type="registry:regexType"
```

```
          minOccurs="0"/>
        <element
          name="expiryPolicy"
          type="registry:expiryPolicyType"
          minOccurs="0"
          default="autoRenew"/>
        <element
          name="nullAuthInfoSupported"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="hostModelSupported"
          default="hostObj"
          minOccurs="0">
          <simpleType>
            <restriction base="token">
              <enumeration value="hostObj"/>
              <enumeration value="hostAttr"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
    </complexType>
    <simpleType name="intHostSharePolicyType">
      <restriction base="token">
        <enumeration value="perZone"/>
        <enumeration value="perSystem"/>
      </restriction>
    </simpleType>
    <simpleType name="extHostSharePolicyType">
      <restriction base="token">
        <enumeration value="perRegistrar"/>
        <enumeration value="perZone"/>
        <enumeration value="perSystem"/>
      </restriction>
    </simpleType>
    <simpleType name="postalInfoTypeSupportType">
      <restriction base="token">
        <enumeration value="loc"/>
        <enumeration value="int"/>
        <enumeration value="locOrInt"/>
        <enumeration value="locAndInt"/>
        <enumeration value="intOptLoc"/>
        <enumeration value="locOptInt"/>
      </restriction>
    </simpleType>
    <complexType name="intHostPolicyType">
```

```
      <sequence>
        <element
          name="minIP"
          type="unsignedShort"/>
        <element
          name="maxIP"
          type="unsignedShort"/>
        <element
          name="sharePolicy"
          type="registry:intHostSharePolicyType"
          minOccurs="0"/>
        <element
          name="uniqueIpAddressesRequired"
          type="boolean"
          minOccurs="0"
          default="false"/>
       </sequence>
    </complexType>
    <complexType name="extHostPolicyType">
      <sequence>
        <element
          name="minIP"
          type="unsignedShort"/>
        <element
          name="maxIP"
          type="unsignedShort"/>
        <element
          name="sharePolicy"
          type="registry:extHostSharePolicyType"
          minOccurs="0"/>
        <element
          name="uniqueIpAddressesRequired"
          type="boolean"
          minOccurs="0"
          default="false"/>
      </sequence>
    </complexType>
    <complexType name="hostType">
      <sequence>
        <element
          name="internal"
          type="registry:intHostPolicyType"/>
        <element
          name="external"
          type="registry:extHostPolicyType"/>
        <element
          name="nameRegex"
          type="registry:regexType"
```

```
          minOccurs="0"/>
        <element
          name="maxCheckHost"
          type="unsignedShort"
          minOccurs="0"/>
        <element
          name="supportedStatus"
          type="registry:supportedStatusType"
          minOccurs="0"/>
        <element
          name="invalidIP"
          type="anyURI"
          minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>
    <complexType name="minMaxLength">
      <sequence>
        <element
          name="minLength"
          type="unsignedShort"/>
        <element
          name="maxLength"
          type="unsignedShort"/>
      </sequence>
    </complexType>
    <simpleType name="contactSharePolicyType">
      <restriction base="token">
        <enumeration value="perZone"/>
        <enumeration value="perSystem"/>
      </restriction>
    </simpleType>
    <complexType name="streetType">
      <complexContent>
        <extension base="registry:minMaxLength">
          <sequence>
            <element
              name="minEntry"
              type="unsignedShort"/>
            <element
              name="maxEntry"
              type="unsignedShort"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="contactAddressType">
      <sequence>
```

```
        <element
          name="street"
          type="registry:streetType"/>
        <element
          name="city"
          type="registry:minMaxLength"/>
        <element
          name="sp"
          type="registry:minMaxLength"/>
        <element
          name="pc"
          type="registry:minMaxLength"/>
      </sequence>
    </complexType>
    <complexType name="postalType">
      <sequence>
        <element
          name="locCharRegex"
          type="registry:regexType"
          minOccurs="0"/>
        <element
          name="name"
          type="registry:minMaxLength"/>
        <element
          name="org"
          type="registry:minMaxLength"/>
        <element
          name="address"
          type="registry:contactAddressType"/>
        <element
          name="voiceRequired"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="voiceExt"
          type="registry:minMaxLength"
          minOccurs="0"/>
        <element
          name="faxExt"
          type="registry:minMaxLength"
          minOccurs="0"/>
        <element
          name="emailRegex"
          type="registry:regexType"
          minOccurs="0"/>
      </sequence>
    </complexType>
```

```
    <complexType name="contactType">
      <sequence>
        <element
          name="contactIdRegex"
          type="registry:regexType"
          minOccurs="0"/>
        <element
          name="contactIdPrefix"
          type="token"
          minOccurs="0"/>
        <element
          name="sharePolicy"
          type="registry:contactSharePolicyType"
          minOccurs="0"/>
        <element
          name="postalInfoTypeSupport"
          type="registry:postalInfoTypeSupportType"/>
        <element
          name="postalInfo"
          type="registry:postalType"/>
        <element
          name="maxCheckContact"
          type="unsignedShort"/>
        <element
          name="authInfoRegex"
          type="registry:regexType"
          minOccurs="0"/>
        <element
          name="clientDisclosureSupported"
          type="boolean"
          minOccurs="0"
          default="false"/>
        <element
          name="supportedStatus"
          type="registry:supportedStatusType"
          minOccurs="0"/>
        <element
          name="transferHoldPeriod"
          type="registry:periodType"
          minOccurs="0"/>
        <element
          name="privacyContactSupported"
          type="boolean"
          minOccurs="0"
          default="true"/>
        <element
          name="proxyContactSupported"
          type="boolean"
```

```
          minOccurs="0"
          default="true"/>
      </sequence>
    </complexType>
    <complexType name="transLimitType">
      <simpleContent>
        <extension base="int">
          <attribute
            name="perMs"
            type="int"
            use="required"/>
        </extension>
      </simpleContent>
    </complexType>
    <complexType name="systemType">
      <sequence>
        <element
          name="maxConnections"
          type="int"
          minOccurs="0"/>
        <element
          name="idleTimeout"
          type="int"
          minOccurs="0"/>
        <element
          name="absoluteTimeout"
          type="int"
          minOccurs="0"/>
        <element
          name="commandTimeout"
          type="int"
          minOccurs="0"/>
        <element
          name="transLimit"
          type="registry:transLimitType"
          minOccurs="0"/>
      </sequence>
    </complexType>
    <simpleType name="unsupportedDataType">
      <restriction base="token">
        <enumeration value="fail"/>
        <enumeration value="ignore"/>
      </restriction>
    </simpleType>
  </schema>
  END
```

5. IANA Considerations

5.1. XML Namespace

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

Registration request for the registry namespace:

URI: urn:ietf:params:xml:ns:epp:registry-0.2
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the registry XML schema:

URI: urn:ietf:params:xml:schema:epp:registry-0.2
Registrant Contact: IESG
XML: See the "Formal Syntax" section of this document.

5.2. EPP Extension Registry

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

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

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: TBD

Status: Active

Notes: None

6. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 7942 [RFC7942] before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942 [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

6.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of draft-gould-carney-regext-registry.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

URL: https://www.verisign.com/en_US/channel-resources/domain-registry-products/epp-sdks

7. Security Considerations

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

8. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions:

- Mario Loffredo, Patrick Mevzek

9. References

9.1. Normative References

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, <https://www.rfc-editor.org/info/rfc952>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[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>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[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>.

[RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, DOI 10.17487/RFC5732, August 2009, <https://www.rfc-editor.org/info/rfc5732>.

[RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <https://www.rfc-editor.org/info/rfc5733>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, DOI 10.17487/RFC5910, May 2010, <https://www.rfc-editor.org/info/rfc5910>.

[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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

9.2. Informative References

[pcre] Hazel, P., "Perl-compatible Regular Expressions (PCRE)", October 2016, <https://www.pcre.org/original/doc/html/pcrepattern.html>.

9.3. URIs

[1] http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

Appendix A. Change History

A.1. Change from 00 to 01

1. Added missing description of the "perRegistrar" value for the <registry:external> <registry:sharePolicy> element.
2. Revised the description of <registry:emailRegex> to be a single optional element instead of an optional list of elements to match the definition in the XML schema.

A.2. Change from 01 to 02

1. Removed the unneeded zoneMemberType from the XML schema.
2. Added reference to the Zone Name section for the <registry:name> elements, since they use the XML schema zoneFormType that supports the "form" attribute with the default value of "aLabel".
3. Made the zoneType crDate element optional to support sending the zone on a create command without the crDate being set by the client.
4. Updated the Implementation Status section to include the leading paragraphs and to include the "Verisign EPP SDK" sub-section.

A.3. Change from 02 to 03

1. Changed the XML namespace from urn:ietf:params:xml:ns:registry-0.1 to urn:ietf:params:xml:ns:epp:registry-0.1, and changed the XML schema registration from urn:ietf:params:xml:ns:registry-0.1 to urn:ietf:params:xml:schema:epp:registry-0.1 based on a request from IANA with draft-ietf-regext-allocation-token.

A.4. Change from 03 to 04

1. Added the optional <registry:contactIdPrefix> element to support a client-specific prefix for the <contact:id> elements in [RFC5733], based on feedback from Patrick Mevzek.
2. Added the optional <registry:unsupportedData> element to define what the server does when unsupported data is sent by the client, based on feedback from Patrick Mevzek.
3. Added the <registry:nullAuthInfoSupported> element to indicate whether the <domain:null> element of [RFC5731] is supported, based on feedback from Patrick Mevzek.
4. Added support for the <registry:flags> and the <registry:protocol> elements under the <registry:keyDataInterface> element to define the supported set of key data interface flags and protocols, based on feedback from Patrick Mevzek.
5. Updated the Introduction sentence "It is up to server policy to define what clients are authorized to execute which commands on which registry zones" to "It is up to server policy what commands are supported and to define the clients that are authorized to execute the commands for the registry zones. The server MUST return a 2101 error response for an unimplemented command and MUST return a 2201 error response for an unauthorized command.", based on feedback from Mario Loffredo.
6. Added two additional <registry:postalInfoTypeSupport> element values, which include "intOptLoc" and "locOptInt", based on feedback from Patrick Mevzek.

7. Added "that SHOULD match the statuses" to the descriptions of the <registry:supportedStatus> elements under the <registry:domain> element, the <registry:host> element, and the <registry:contact> element, based on feedback from Mario Loffredo.
8. Added "or transferred" to the description of the <registry:expiryPolicy> element "autoRenew" value, based on feedback from Mario Laffredo.
9. Added support for an optional list of <registry:invalidIP> elements, under the <registry:host> element, to reference a list of externally defined invalid IP addresses URIs, based on feedback from Patrick Mevzek.
10. Changed all references of urn:ietf:params:xml:ns:epp:registry-0.1 to urn:ietf:params:xml:ns:epp:registry-0.2 in the draft.
11. Added a "Regular Expressions" section that describes the regular expression syntax used in the draft, which is Perl-compatible Regular Expression (PCRE). The elements that use regular expression values reference the new "Regular Expressions" section. Referencing the expected regular expression syntax to use is based on feedback from Patrick Mevzek.
12. Added support for the <registry:locCharRegex> element to define the acceptable set of characters for the "loc" postal information elements, based on feedback from Patrick Mevzek.
13. Updated to make the regular expression elements follow a consistent naming convention and cardinality. Changed <registry:regex> to <registry:nameRegex> under the <registry:domainName> element, and changed to a single element. Changed the <registry:nameRegex> under the <registry:host> element to a single element.
14. Added support for the host attribute model in RFC 5731 by adding the <registry:hostModelSupported> element under the <registry:domain> element, making the <registry:maxCheckHost> element optional, and making the <registry:childHost> element optional, based on feedback from Mario Loffredo.
15. Added the <registry:exceedMaxExDate> element under the <registry:domain> element to support returning the server policy when the client exceeds the maximum expiration date on a per renewal command basis, based on feedback from Patrick Mevzek.
16. Re-defined the <registry:schedule> element to use a simplified XML definition in place of a crontab definition, and added support for one or more <registry:schedule> elements per batch job. The <registry:schedule> element supports multiple frequencies (daily, weekly, monthly), both local and UTC time zones, and a time using the XML schema "time" type. This change is based on feedback from Patrick Mevzek.
17. Made the zone names more consistent by using EXAMPLE and EXAMPLE#.

18. Added support for the <registry:all> element "scope" attribute and the <registry:zone> element "accessible" boolean attribute, to enable the client to explicitly specify which zones are of interest (accessible, available, or both) in the info command and to enable the server to indicate in the info response whether a zone is accessible or not.

Authors' Addresses

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

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

Lin Jia
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US

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

Roger Carney
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: rcarney@godaddy.com
URI: http://www.godaddy.com

Jody Kolker
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

Email: jkolker@godaddy.com
URI: http://www.godaddy.com