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

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: March 27, 2020

J. Gould
VeriSign, Inc.
M. Casanova
SWITCH

September 24, 2019

# Extensible Provisioning Protocol (EPP) Unhandled Namespaces draft-gould-casanova-regext-unhandled-namespaces-02

## Abstract

The Extensible Provisioning Protocol (EPP), as defined in RFC 5730, includes a method for the client and server to determine the objects to be managed during a session and the object extensions to be used during a session. The services are identified using namespace URIs. How should the server handle service data that needs to be returned in the response when the client does not support the required service namespace URI, which is referred to as an unhandled namespace? An unhandled namespace is a significant issue for the processing of RFC 5730 poll messages, since poll messages are inserted by the server prior to knowing the supported client services, and the client needs to be capable of processing all poll messages. This document defines an operational practice that enables the server to return information associated with unhandled namespace URIs that is compliant with the negotiated services defined in RFC 5730.

## 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 https://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 March 27, 2020.

Table of Contents



## 1. Introduction

The Extensible Provisioning Protocol (EPP), as defined in [RFC5730], includes a method for the client and server to determine the objects to be managed during a session and the object extensions to be used during a session. The services are identified using namespace URIs. How should the server handle service data that needs to be returned in the response when the client does not support the required service namespace URI, which is referred to as an unhandled namespace? An unhandled namespace is a significant issue for the processing of [RFC5730] poll messages, since poll messages are inserted by the server prior to knowing the supported client services, and the client

needs to be capable of processing all poll messages. An unhandled namespace is an issue also for general EPP responses when the server has information that it cannot return to the client due to the client's supported services. The server should be able to return unhandled namespace information that the client can process later. This document defines an operational practice that enables the server to return information associated with unhandled namespace URIs that is compliant with the negotiated services defined in [RFC5730].

### 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, "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 examples reference XML namespace prefixes that are used for the associated XML namespaces. Implementations MUST NOT depend on the example XML namespaces and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents. The example namespace prefixes used and their associated XML namespaces include:

"changePoll": urn:ietf:params:xml:ns:changePoll-1.0
"domain": urn:ietf:params:xml:ns:domain-1.0
"secDNS": urn:ietf:params:xml:ns:secDNS-1.1

In the template example XML, placeholder content is represented by the following variables:

"[NAMESPACE-XML]": XML content associated with a login service namespace URI. An example is the <domain:infData> element content in [RFC5731].
"[NAMESPACE-URI]": XML namespace URI associated with the [NAMESPACE-XML] XML content. An example is "urn:ietf:params:xml:ns:domain-1.0" in [RFC5731].

## 2. Unhandled Namespaces

An Unhandled Namespace is an XML namespace that is associated with a response extension that is not included in the client-specified EPP login services of [RFC5730]. The EPP login services consists of the set of XML namespace URIs included in the <objURI> or <extURI> elements of the [RFC5730] EPP <login> command. The services supported by the server are included in the <objURI> and <extURI> elements of the [RFC5730] EPP <greeting>, which should be a superset of the login services included in the EPP <login> command. A server may have information associated with a specific namespace that it needs to return in the response to a client. The unhandled namespaces problem exists when the server has information, that it needs to return to the client, that is not supported by the client based on the negotiated EPP <login> command services.

## 3. Use of EPP <extValue> for Unhandled Namespace Data

When a server has data to return to the client, that the client does not support based on the login services, the server MAY return a successful response, with the data for each unsupported namespace moved into an [RFC5730] <extValue> element. The unhandled namespace will not cause an error response, but the unhandled namespace data will instead be moved to an <extValue> element, along with a reason why the unhandled namespace data could not be included in the appropriate location of the response. The <extValue> element XML will not be processed by the XML processor. The <extValue> element contains the following child elements:

<value>: Contains a child-element with the unhandled namespace XML. The XML namespace and namespace prefix of the child element MUST be defined, which MAY be defined in the <value> element or in the the child element. XML processing of the <value> element is disabled in [RFC5730], so the information can safely be returned in the <value> element.

<reason>: A formatted human-readable message that indicates the reason the unhandled namespace data was not returned in the appropriate location of the response. The formatted reason SHOULD follow the Augmented Backus-Naur Form (ABNF) grammar [RFC5234] format: NAMESPACE-URI "not in login services", where NAMESPACE-URI is the unhandled XML namespace like "urn:ietf:params:xml:ns:domain-1.0" for [RFC5731].

This document supports handling of unsupported namespaces for [RFC3735] object-level extensions and command-response extensions. This document does not support [RFC3735] protocol-level extensions or authentication information extensions. Refer to the following

sections on how to handle an unsupported object-level extension namespace or an unsupported command-response extension namespace.

### 3.1. Unhandled Object-Level Extension

An object-level extension in [RFC5730] is a child element of the <resData> element. If the client does not handle the namespace of the object-level extension, then the <resData> element is removed and its object-level extension child element is moved into a [RFC5730] <extValue> <value> element, with the namespace URI included in the corresponding <extValue> <reason> element. The response becomes a general EPP response without the <resData> element.

Template response for a supported object-level extension. The [NAMESPACE-XML] variable represents the object-level extension XML.

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      [NAMESPACE-XML]
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Template unhandled namespace response for an unsupported object-level extension. The [NAMESPACE-XML] variable represents the object-level extension XML and the [NAMESPACE-URI] variable represents the object-level extension XML namespace URI.

```
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:      <extValue>
S:        <value>
S:          [NAMESPACE-XML]
S:        </value>
S:        <reason>
S:          [NAMESPACE-URI] not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The EPP response is converted from an object response to a general EPP response by the server when the client does not support the object-level extension namespace URI. Below is example of converting the <transfer> query response example in [RFC5730] to an unhandled namespace response.

[RFC5730] example <transfer> query response converted into an unhandled namespace 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:      <extValue>
S:        <value>
S:          <domain:trnData
S:            xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:            <domain:name>example.com</domain:name>
S:            <domain:trStatus>pending</domain:trStatus>
S:            <domain:reID>ClientX</domain:reID>
S:            <domain:reDate>2000-06-06T22:00:00.0Z</domain:reDate>
S:            <domain:acID>ClientY</domain:acID>
S:            <domain:acDate>2000-06-11T22:00:00.0Z</domain:acDate>
S:            <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
S:          </domain:trnData>
S:        </value>
S:        <reason>
S:          urn:ietf:params:xml:ns:domain-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

### 3.2. Unhandled Command-Response Extension

A command-response extension in [RFC5730] is a child element of the <extension> element. If the client does not handle the namespace of the command-response extension, the command-response child element is moved into a [RFC5730] <extValue> <value> element, with the namespace URI included in the corresponding <extValue> <reason> element. If after moving the command-response child element there are no additional command-response child elements, the <extension> element MUST be removed.

Template response for a supported command-response extension. The [NAMESPACE-XML] variable represents the command-response extension XML.

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      [NAMESPACE-XML]
S:    </extension>
S:    <trID>
S:     <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Template unhandled namespace response for an unsupported command-response extension. The [NAMESPACE-XML] variable represents the command-response extension XML and the [NAMESPACE-URI] variable represents the command-response extension XML namespace URI.

```
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:      <extValue>
S:        <value>
S:         [NAMESPACE-XML]
S:        </value>
S:        <reason>
S:          [NAMESPACE-URI] not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The EPP response is converted to an unhandled namespace response by moving the unhandled command-response extension from under the

<extension> to an <extValue> element. Below is example of converting the DS Data Interface <info> response example in [RFC5910] to an unhandled namespace response.

[RFC5910] DS Data Interface <info> response converted into an unhandled namespace 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:      <extValue>
S:        <value>
S:        <secDNS:infData
S:          xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
S:          <secDNS:dsData>
S:          <secDNS:keyTag>12345</secDNS:keyTag>
S:          <secDNS:alg>3</secDNS:alg>
S:          <secDNS:digestType>1</secDNS:digestType>
S:          <secDNS:digest>49FD46E6C4B45C55D4AC</secDNS:digest>
S:          </secDNS:dsData>
S:          </secDNS:infData>
S:        </value>
S:        <reason>
S:          urn:ietf:params:xml:ns:secDNS-1.1 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:          <domain:hostObj>ns2.example.com</domain:hostObj>
S:        </domain:ns>
S:        <domain:host>ns1.example.com</domain:host>
S:        <domain:host>ns2.example.com</domain:host>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientX</domain:upID>
```

```
S:        <domain:upDate>1999-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

## 4. Usage with General EPP Responses

The unhandled namespace approach defined in Section 3 MAY be used for a general EPP response to an EPP command. A general EPP response includes any non-poll message EPP response. The use of the unhandled namespace approach for poll message EPP responses is defined in Section 5. The server MAY exclude the unhandled namespace information in the general EPP response or MAY include it using the unhandled namespace approach.

The unhandled namespace approach for general EPP responses SHOULD only be applicable to command-response extensions, defined in Section 3.2, since the server SHOULD NOT accept an object-level EPP command if the client did not include the object-level namespace URI in the login services. An object-level EPP response extension is returned when the server successfully executes an object-level EPP command extension. The server MAY return an unhandled object-level extension to the client as defined in Section 3.1.

Returning domain name Redemption Grace Period (RGP) data, based on [RFC3915], provides an example of applying the unhandled namespace approach for a general EPP response. If the client does not include the "urn:ietf:params:xml:ns:rgp-1.0" namespace URI in the login services, and the domain <info> response of a domain name does have RGP information, the server MAY exclude the <rgp:infData> element from the EPP response or MAY include it under in the <extValue> element per Section 3.2.

[RFC5731] domain name <info> response with the unhandled [RFC3915] <rgp:infData> element included under an <extValue> element:

```
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:      <extValue>
S:        <value>
S:          <rgp:infData xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0">
S:            <rgp:rgpStatus s="redemptionPeriod"/>
S:          </rgp:infData>
S:        </value>
S:        <reason>
S:          urn:ietf:params:xml:ns:rgp-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="pendingDelete"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:          <domain:hostObj>ns1.example.net</domain:hostObj>
S:        </domain:ns>
S:        <domain:host>ns1.example.com</domain:host>
S:        <domain:host>ns2.example.com</domain:host>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientX</domain:upID>
S:        <domain:upDate>1999-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

## 5. Usage with Poll Message EPP Responses

The unhandled namespace approach, defined in Section 3, MUST be used if there is unhandled namespace information included in an EPP <poll> message response. The server inserts poll messages into the client's poll queue independent of knowing the supported client login services, therefore there may be unhandled object-level and command-response extensions included in a client's poll queue. In [RFC5730], the <poll> command is used by the client to retrieve and acknowledge poll messages that have been inserted by the server. The <poll> message response is an EPP response that includes the <msgQ> element that provides poll queue meta-data about the message. The unhandled namespace approach, defined in Section 3, is used for an unhandled object-level extension and for each of the unhandled command-response extensions attached to the <poll> message response. The resulting EPP <poll> message response MAY have either or both the object-level extension or command-response extensions moved to <extValue> elements, as defined in Section 3.

The Change Poll Message, as defined in [I-D.ietf-regext-change-poll], which is an extension of any EPP object, is an example of applying the unhandled namespace approach for EPP <poll> message responses. The object that will be used in the examples is a [RFC5731] domain name object.

[RFC5731] domain name <info> <poll> message response with the unhandled [I-D.ietf-regext-change-poll] <changePoll:changeData> element included under an <extValue> element:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:      <extValue>
S:        <value>
S:          <changePoll:changeData
S:          xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:            state="after">
S:            <changePoll:operation>update</changePoll:operation>
S:            <changePoll:date>
S:              2013-11-22T05:00:00.000Z</changePoll:date>
S:            <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:            <changePoll:who>URS Admin</changePoll:who>
S:            <changePoll:caseId type="urs">urs123
S:            </changePoll:caseId>
S:            <changePoll:reason>URS Lock</changePoll:reason>
S:          </changePoll:changeData>
```

```
S:        </value>
S:        <reason>
S:      urn:ietf:params:xml:ns:changePoll-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <msgQ
S:      count="15"
S:      id="1"
S:    >
S:      <qDate>2018-08-24T19:21:51.087Z</qDate>
S:      <msg>Registry initiated update of domain.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>change-poll.tld</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="serverUpdateProhibited"/>
S:        <domain:status s="serverDeleteProhibited"/>
S:        <domain:status s="serverTransferProhibited"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-05-03T04:00:00.000Z</domain:crDate>
S:        <domain:upID>ClientZ</domain:upID>
S:        <domain:upDate>2013-11-22T05:00:00.000Z</domain:upDate>
S:        <domain:exDate>2014-05-03T04:00:00.000Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Unhandled [RFC5731] domain name <info> <poll> message response and the unhandled [I-D.ietf-regext-change-poll] <changePoll:changeData> element included under an <extValue> element:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
```

```
S:      <extValue>
S:        <value>
S:          <domain:infData
S:            xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:            <domain:name>change-poll.tld</domain:name>
S:            <domain:roid>EXAMPLE1-REP</domain:roid>
S:            <domain:status s="serverUpdateProhibited"/>
S:            <domain:status s="serverDeleteProhibited"/>
S:            <domain:status s="serverTransferProhibited"/>
S:            <domain:registrant>jd1234</domain:registrant>
S:            <domain:contact type="admin">sh8013</domain:contact>
S:            <domain:contact type="tech">sh8013</domain:contact>
S:            <domain:clID>ClientX</domain:clID>
S:            <domain:crID>ClientY</domain:crID>
S:            <domain:crDate>2012-05-03T04:00:00.000Z</domain:crDate>
S:            <domain:upID>ClientZ</domain:upID>
S:            <domain:upDate>2013-11-22T05:00:00.000Z</domain:upDate>
S:            <domain:exDate>2014-05-03T04:00:00.000Z</domain:exDate>
S:          </domain:infData>
S:        </value>
S:        <reason>
S:        urn:ietf:params:xml:ns:domain-1.0 not in login services
S:        </reason>
S:      </extValue>
S:      <extValue>
S:        <value>
S:          <changePoll:changeData
S:          xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:            state="after">
S:            <changePoll:operation>update</changePoll:operation>
S:            <changePoll:date>
S:              2013-11-22T05:00:00.000Z</changePoll:date>
S:            <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:            <changePoll:who>URS Admin</changePoll:who>
S:            <changePoll:caseId type="urs">urs123
S:            </changePoll:caseId>
S:            <changePoll:reason>URS Lock</changePoll:reason>
S:          </changePoll:changeData>
S:        </value>
S:        <reason>
S:      urn:ietf:params:xml:ns:changePoll-1.0 not in login services
S:        </reason>
S:      </extValue>
S:    </result>
S:    <msgQ
S:      count="15"
S:      id="1"
S:    >
```

```
S:      <qDate>2018-08-24T19:23:12.822Z</qDate>
S:      <msg>Registry initiated update of domain.</msg>
S:    </msgQ>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

## 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 an implementation of the unhandled namespaces for the processing of the poll queue messages.

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

### 6.2. SWITCH Automated DNSSEC Provisioning Process

Organization: SWITCH

Name: Registry of .CH and .LI

Description: SWITCH uses poll messages to inform the registrar about DNSSEC changes at the registry triggered by CDS records. These poll messages are enriched with the 'urn:ietf:params:xml:ns:changePoll-1.0' and the 'urn:ietf:params:xml:ns:secDNS-1.1' extension that are rendered in the poll msg response according to this draft.

Level of maturity: Operational

Coverage: All aspects of the protocol are implemented.

Licensing: Proprietary

Contact: martin.casanova@switch.ch

URL: https://www.nic.ch/cds

## 7. Security Considerations

The 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

TBD

## 9. Normative References

[I-D.ietf-regext-change-poll]
Gould, J. and K. Feher, "Change Poll Extension for the Extensible Provisioning Protocol (EPP)", draft-ietf-regext-change-poll-12 (work in progress), January 2019.

[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>.

[RFC3735] Hollenbeck, S., "Guidelines for Extending the Extensible Provisioning Protocol (EPP)", RFC 3735, DOI 10.17487/RFC3735, March 2004, <https://www.rfc-editor.org/info/rfc3735>.

[RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <https://www.rfc-editor.org/info/rfc3915>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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>.

[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>.

[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>.

## Appendix A. Change History

### A.1. Change from 00 to 01

1. Removed xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" reference from examples.
2. removed <extension></extension> block from example.

3. added SWITCH Automated DNSSEC Provisioning Process at Implementation Status

### A.2. Change from 01 to 02

1. Minor changes

## Authors' Addresses

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

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

Martin Casanova
SWITCH
P.O. Box
Zurich, ZH 8021
CH

Email: martin.casanova@switch.ch
URI: http://www.switch.ch

Network Working Group J. Gould
Internet-Draft M. Pozun
Intended status: Standards Track VeriSign, Inc.
Expires: July 19, 2019 January 15, 2019

# Login Security Extension for the Extensible Provisioning Protocol (EPP) draft-gould-regext-login-security-03

## Abstract

The Extensible Provisioning Protocol (EPP) includes a client authentication scheme that is based on a user identifier and password. The structure of the password field is defined by an XML Schema data type that specifies minimum and maximum password length values, but there are no other provisions for password management other than changing the password. This document describes an EPP extension that allows longer passwords to be created and adds additional security features to the EPP login command and response.

## Status of This Memo

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

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

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

This Internet-Draft will expire on July 19, 2019.

## Copyright Notice

Table of Contents



## 1. Introduction

This document describes an Extensible Provisioning Protocol (EPP) extension for enhancing the security of the EPP login command in EPP RFC 5730. The enhancements include supporting longer passwords (or passphrases) than the 16-character maximum and providing a list of security events in the login response. The password (current and new) in EPP RFC 5730 can be overridden by the password included in the extension to extend past the 16-character maximum. The security events supported include: password expiry, client certificate expiry, insecure cipher, insecure TLS protocol, new pasword complexity, login security statistical warning, and a custom event. The attributes supported by the security events include identifying the event type or sub-type, indicating the security level of warning or error, a future or past-due expiration date, the value that resulted in the

event, the duration of the statistical event, and a free-form description with an optional language.

### 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.

"loginSec-0.3" is used as an abbreviation for "urn:ietf:params:xml:ns:epp:loginSec-0.3". The XML namespace prefix "loginSec" is used, but implementations MUST NOT depend on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

## 2. Migrating to Newer Versions of This Extension

(Note to RFC Editor: remove this section before publication as an RFC.)

Servers which implement this extension SHOULD provide a way for clients to progressively update their implementations when a new version of the extension is deployed.

Servers SHOULD (for a temporary migration period) provide support for older versions of the extension in parallel to the newest version, and allow clients to select their preferred version via the <svcExtension> element of the <login> command.

If a client requests multiple versions of the extension at login, then, when preparing responses to commands which do not include extension elements, the server SHOULD only include extension elements in the namespace of the newest version of the extension requested by the client.

When preparing responses to commands which do include extension elements, the server SHOULD only include extension elements for the extension versions present in the command.

## 3. Object Attributes

This extension adds additional elements to [RFC5730] login command and response. Only those new elements are described here.

### 3.1. Event

A security event, using the <loginSec:event> element, represents either a warning or error identified by the server after the client has connected and submitted the login command. There MAY be multiple events returned that provides information for the client to address. The <loginSec:event> MAY include a free form description. All of the security events use a consistent set of attributes, where the exact set of applicable attributes is based on the event type. The supported set of <loginSec:event> element attributes include:

"type": A REQUIRED attribute that defines the type of security event. The enumerated list of "type" values include:

- "password": Identifies a password expiry event, where the password expires in the future or has expired based on the "exDate" date and time.
- "certificate": Identifies a client certificate expiry event, where the client certificate will expire at the "exDate" date and time.
- "cipher": Identifies the use of an insecure or deprecated TLS cipher suite.
- "tlsProtocol": Identifies the use of an insecure or deprecated TLS protocol.
- "newPW": The new password does not meet the server password complexity requirements.
- "stat": Provides a login security statistical warning that MUST set the "name" of the statistic.
- "custom": Custom event type that MUST set the "name" attribute with the custom event type name.

"name": Used to define a sub-type or the type name when the "type" attribute is "custom".

"level": Defines the level of the event as either "warning" for a warning event that needs action, or "error" for an error event that requires immediate action.

"exDate": Contains the date and time that a "warning" level has or will become an "error" level. At expiry there MAY be an error to connect or MAY be an error to login. An example is an expired certificate that will result in a error to connect or an expired password that may result in a failed login.

"value": Identifies the value that resulted in the login security event. An example is the negotiated insecure cipher suite or the negotiated insecure TLS protocol.
"duration": Defines the duration that a statistical event is associated with.
"lang": Identifies the language of the free form description if the negotiated language is something other than the default value of "en" (English).

Example login security event for a password expiring in a week:

```
<loginSec:event
  type="password"
  level="warning"
  exDate="2018-04-01T22:00:00.0Z"
  lang="en">
  Password expiration soon
</loginSec:event>
```

Example login security event for identifying 100 failed logins over the last day, using the "stat" sub-type of "failedLogins":

```
<loginSec:event
  type="stat"
  name="failedLogins"
  level="warning"
  value="100"
  duration="P1D">
  Excessive invalid daily logins
</loginSec:event>
```

### 3.2. "[LOGIN-SECURITY]" Password

The <loginSec:pw> element MUST override the [RFC5730] <pw> element only if the <pw> contains the predefined value of "[LOGIN-SECURITY]", which is a constant value for the server to use the <loginSec:pw> element for the password. Similarly, the <loginSec:newPW> element MUST override the [RFC5730] <newPW> element only if the <newPW> contains the predefined value of "[LOGIN-SECURITY]", which is a constant value for the server to use the <loginSec:newPW> element for the new password. The "[LOGIN-SECURITY]" pre-defined string MUST be supported by the server for the client to explicitly indicate to the server whether to use <loginSec:pw> element in place of the [RFC5730] <pw> element or to use the <loginSec:newPW> in place of the [RFC5730] <newPW> element.

### 3.3. 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 [W3C.REC-xmlschema-2-20041028] 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.

## 4. EPP Command Mapping

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

### 4.1. EPP <login> Command

This extension defines additional elements to extend the EPP <login> command and response to be used in conjunction with [RFC5730].

The EPP <login> command is used to establish a session with an EPP server. This extension overrides the password that is passed with the [RFC5730] <pw> or the <newPW> element as defined in Section 3.2. A <loginSec:loginSec> element is sent along with the [RFC5730] <login> command and MUST contain at least one of the following child elements:

<loginSec:userAgent>: OPTIONAL client user agent that identifies the client software and platform used by the server to identify functional or security constraints, current security issues, and potential future functional or security issues for the client.

<loginSec:pw>: OPTIONAL plain text password that is case sensitive, has a minimum length of 6 characters, and has a maximum length that is up to server policy. All leading and trailing whitespace is removed, and all internal contiguous whitespace that includes #x9 (tab), #xA (linefeed), #xD (carriage return), and #x20 (space) is replaced with a single #x20 (space). This element MUST only be used if the [RFC5730] <pw> element is set to the "[LOGIN-SECURITY]" value.

<loginSec:newPW>: OPTIONAL plain text new password that is case sensitive, has a minimum length of 6 characters, and has a maximum length that is up to server policy. All leading and trailing whitespace is removed, and all internal contiguous whitespace that includes #x9 (tab), #xA (linefeed), #xD (carriage return), and #x20 (space) is replaced with a single #x20 (space). This element MUST only be used if the [RFC5730] <newPW> element is set to the "[LOGIN-SECURITY]" value.

Example login command that uses the <loginSec:pw> element instead of the [RFC5730] <pw> element to establish the session and includes the <loginSec:userAgent> element:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>[LOGIN-SECURITY]</pw>
C:      <options>
C:        <version>1.0</version>
C:        <lang>en</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>urn:ietf:params:xml:ns:epp:loginSec-0.3</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
C:        <loginSec:userAgent>EPP SDK/1.0.0
C:        (Java 1.7.0_15; x86_64 Mac OS X 10.11.6)
C:        </loginSec:userAgent>
C:        <loginSec:pw>this is a long password</loginSec:pw>
C:      </loginSec:loginSec>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example login command that uses the <loginSec:pw> element instead of the [RFC5730] <pw> element to establish the session, and uses the <loginSec:newPW> element instead of the [RFC5730] <newPW> element to set the new password:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>[LOGIN-SECURITY]</pw>
C:      <newPW>[LOGIN-SECURITY]</newPW>
C:      <options>
C:        <version>1.0</version>
C:        <lang>en</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>urn:ietf:params:xml:ns:epp:loginSec-0.3</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
C:        <loginSec:pw>this is a long password
C:        </loginSec:pw>
C:        <loginSec:newPW>new password that is still long
C:        </loginSec:newPW>
C:      </loginSec:loginSec>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example login command that uses the [RFC5730] <pw> element to establish the session, and uses the <loginSec:newPW> element instead of the [RFC5730] <newPW> element to set the new password:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>shortpassword</pw>
C:      <newPW>[LOGIN-SECURITY]</newPW>
C:      <options>
C:        <version>1.0</version>
C:        <lang>en</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>urn:ietf:params:xml:ns:epp:loginSec-0.3</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <extension>
C:      <loginSec:loginSec
C:        xmlns:loginSec=
C:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
C:        <loginSec:newPW>new password that is still long
C:        </loginSec:newPW>
C:      </loginSec:loginSec>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Upon a completed login command (success or failed), the extension MUST be included in the response based on the following conditions:

Client supports extension: client supports the extension based on the <svcExtension> element of the <login> command.

At least one login security event: The server has identified at least one login security event to communicate to the client.

The extension to the EPP response uses the <loginSec:loginSecData> element that contains the following child elements:

<loginSec:event>: One or more <loginSec:event> elements defined in Section 3.1.

Example EPP response to a successful login command where the password will expire in a week:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <loginSec:loginSecData
S:        xmlns:loginSec=
S:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
S:        <loginSec:event
S:          type="password"
S:          level="warning"
S:          exDate="2018-04-01T22:00:00.0Z"
S:          lang="en">
S:          Password expiring in a week
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example EPP response to a failed login command where the password has expired and the new password does not meet the server complexity requirements:

```
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="2200">
S:      <msg>Authentication error</msg>
S:    </result>
S:    <extension>
S:      <loginSec:loginSecData
S:        xmlns:loginSec=
S:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
S:        <loginSec:event
S:          type="password"
S:          level="error"
S:          exDate="2018-03-26T22:00:00.0Z">
S:          Password has expired
S:        </loginSec:event>
S:        <loginSec:event
S:          type="newPW"
S:          level="error">
S:          New password does not meet complexity requirements
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example EPP response to a successful login command where there is a set of login security events:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <loginSec:loginSecData
S:        xmlns:loginSec=
S:          "urn:ietf:params:xml:ns:epp:loginSec-0.3">
S:        <loginSec:event
```

```
S:          type="password"
S:          level="warning"
S:          exDate="2018-04-01T22:00:00.0Z"
S:          lang="en">
S:          Password expiration soon
S:        </loginSec:event>
S:        <loginSec:event
S:          type="certificate"
S:          level="warning"
S:          exDate="2018-04-02T22:00:00.0Z"/>
S:        <loginSec:event
S:          type="cipher"
S:          level="warning"
S:          value="TLS_RSA_WITH_AES_128_CBC_SHA">
S:          Non-PFS Cipher negotiated
S:        </loginSec:event>
S:        <loginSec:event
S:          type="tlsProtocol"
S:          level="warning"
S:          value="TLSv1.0">
S:          Insecure TLS protocol negotiated
S:        </loginSec:event>
S:        <loginSec:event
S:          type="stat"
S:          name="failedLogins"
S:          level="warning"
S:          value="100"
S:          duration="P1D">
S:          Excessive invalid daily logins
S:        </loginSec:event>
S:        <loginSec:event
S:          type="custom"
S:          name="myCustomEvent"
S:          level="warning">
S:          A custom login security event occured
S:        </loginSec:event>
S:      </loginSec:loginSecData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

## 5. Formal Syntax

One schema is presented here that is the EPP Login Security Extension 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.

### 5.1. Login Security Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
 <schema targetNamespace="urn:ietf:params:xml:ns:epp:loginSec-0.3"
         xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
         xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
         xmlns:loginSec="urn:ietf:params:xml:ns:epp:loginSec-0.3"
         xmlns="http://www.w3.org/2001/XMLSchema"
         elementFormDefault="qualified">

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


   <annotation>
     <documentation>
       Extensible Provisioning Protocol v1.0
       Login Security Extension Schema.
     </documentation>
   </annotation>

   <!-- login command extension elements -->
   <element name="loginSec" type="loginSec:loginSecType"/>

   <!--
    Attributes associated with the login command extension.
   -->
   <complexType name="loginSecType">
    <sequence>
      <element name="userAgent" type="token"
       minOccurs="0"/>
      <element name="pw" type="loginSec:pwType"
       minOccurs="0"/>
```

```
        <element name="newPW" type="loginSec:pwType"
         minOccurs="0"/>
       </sequence>
      </complexType>

     <simpleType name="pwType">
       <restriction base="token">
         <minLength value="6"/>
       </restriction>
     </simpleType>


      <!-- login response extension elements -->
      <element name="loginSecData" type="loginSec:loginSecDataType"/>

      <!--
       Attributes associated with the change.
      -->
      <complexType name="loginSecDataType">
        <sequence>
          <element name="event" type="loginSec:eventType"
             minOccurs="1" maxOccurs="unbounded"/>
        </sequence>
      </complexType>

      <complexType name="eventType">
        <simpleContent>
          <extension base="normalizedString">
            <attribute name="type"
              type="loginSec:typeEnum" use="required"/>
            <attribute name="name" type="token"/>
            <attribute name="level"
              type="loginSec:levelEnum" use="required"/>
            <attribute name="exDate" type="dateTime"/>
            <attribute name="value" type="token"/>
            <attribute name="duration"
              type="duration"/>
            <attribute name="lang"
              type="language" default="en"/>
           </extension>
        </simpleContent>
      </complexType>

      <!--
       Enumerated list of event types, with extensibility via "custom".
       -->
      <simpleType name="typeEnum">
        <restriction base="token">
```

```
          <enumeration value="password"/>
          <enumeration value="certificate"/>
          <enumeration value="cipher"/>
          <enumeration value="tlsProtocol"/>
          <enumeration value="newPW"/>
          <enumeration value="stat"/>
          <enumeration value="custom"/>
        </restriction>
      </simpleType>

      <!--
       Enumerated list of levels.
       -->
      <simpleType name="levelEnum">
        <restriction base="token">
          <enumeration value="warning"/>
          <enumeration value="error"/>
        </restriction>
      </simpleType>

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

## 6. IANA Considerations

### 6.1. XML Namespace

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

Registration request for the loginSec namespace:

URI: urn:ietf:params:xml:ns:epp:loginSec-0.3
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the loginSec XML schema:

URI: urn:ietf:params:xml:schema:epp:loginSec-0.3
Registrant Contact: IESG
XML: See the "Formal Syntax" section of this document.

### 6.2. 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: "Login Security Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

## 7. Implementation Status

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

TBD

## 8. Security Considerations

The extension leaves the password (<pw> element) and new password (<newPW> element) minimum length beyond 6 characters and the maximum length up to sever policy. The server SHOULD enforce minimum and maximum length requirements that are appropriate for their operating environment. One example of a guideline for password length policies can be found in section 5 of NIST Special Publication 800-63B [1].

The client SHOULD NOT decrease the security of a new password by decreasing the length of the current password. For example, a client with a 20 character password set using the extension, should not use the login command in [RFC5730] without using the extension, to set a new password that is less than or equal to 16 characters.

The extension provides an extensible list of login security events to inform clients of connection and login warnings and errors.

## 9. Acknowledgements

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

- Patrick Mevzek
- Scott Hollenbeck

## 10. References

### 10.1. Normative References

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

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

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[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>.

[W3C.REC-xmlschema-2-20041028] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

### 10.2. Informative References

[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>.

### 10.3. URIs

[1] https://pages.nist.gov/800-63-3/sp800-63b.html

## Appendix A. Change History

### A.1. Change from 00 to 01

1. Based on the feedback from Patrick Mevzek and a proposal from Scott Hollenbeck, changed the minimum length of the password from 8 to 6, revised the description of the password, and added text in the Security Considerations section for the server password length policy.

### A.2. Change from 01 to 02

1. Changed the XML namespace from urn:ietf:params:xml:ns:loginSec-0.3 to urn:ietf:params:xml:ns:epp:loginSec-0.3, and changed the XML schema registration from urn:ietf:params:xml:ns:loginSec-0.3 to urn:ietf:params:xml:schema:epp:loginSec-0.3 based on a request from IANA with draft-ietf-regext-allocation-token.

### A.3. Change from 02 to 03

Updates based on the review by Patrick Mevzek, that include:

1. Fix the inconsistent case for newPW, that required a global change in the draft text and an update to the XML schema to "urn:ietf:params:xml:ns:loginSec-0.3".
2. Changed "contains the following child elements" to "MUST contain at least one of the following child elements", section "EPP <login> Command" to ensure that an empty <loginSec:loginSec> element is not passed.
3. Add "The client SHOULD NOT decrease the security of a new password by decreasing the length of the current password." along with an example to the "Security Considerations" section.

## Authors' Addresses

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

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

Matthew Pozun
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
US

Email: mpozun@verisign.com
URI: http://www.verisign.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 7, 2019

J. Gould
VeriSign, Inc.
K. Feher
Neustar

October 04, 2018

# Allocation Token Extension for the Extensible Provisioning Protocol (EPP)
## draft-ietf-regext-allocation-token-12

## Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension for including an Allocation Token in "query" and "transform" commands. The Allocation Token is used as a credential that authorizes a client to request the allocation of a specific object from the server, using one of the EPP transform commands including create and transfer.

## 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 https://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 7, 2019.

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

## 1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], supports passing an Allocation Token as a credential that authorizes a client to request the allocation of a specific object from the server, using one of the EPP transform commands including create and transfer.

Allocation is when a server assigns the sponsoring client of an object based on the use of an Allocation Token credential. Examples include allocating a registration based on a pre-eligibility Allocation Token, allocating a premium domain name registration based on an auction Allocation Token, allocating a registration based on a founders Allocation Token, and allocating an existing domain name held by the server or by a different sponsoring client based on an Allocation Token passed with a transfer command.

Clients pass an Allocation Token to the server for validation, and the server determines if the supplied Allocation Token is one supported by the server. It is up to server policy which EPP transform commands and which objects require the Allocation Token. The Allocation Token MAY be returned to an authorized client for passing out-of-band to a client that uses it with an EPP transform command.

### 1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

XML 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 the examples are provided only to illustrate element relationships and are not REQUIRED in the protocol.

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

The "abc123" token value is used as a placeholder value in the examples. The server MUST support token values that follow the Security Considerations (Section 7) section.

The domain object attribute values, including the "2fooBAR" <domain:pw> value, in the examples are provided for illustration purposes only. Refer to [RFC5731] for details on the domain object attributes.

## 2. Object Attributes

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

### 2.1. Allocation Token

The Allocation Token is a simple XML "token" type. The exact format of the Allocation Token is up to server policy. The server MAY have the Allocation Token for each object to match against the Allocation Token passed by the client to authorize the allocation of the object. The <allocationToken:allocationToken> element is used for all of the supported EPP commands as well as the info response. If the supplied Allocation Token passed to the server does not apply to the object, the server MUST return an EPP error result code of 2201.

Authorization information, like what is defined in the EPP domain name mapping [RFC5731], is associated with objects to facilitate transfer operations. The authorization information is assigned when an object is created. The Allocation Token and the authorization information are both credentials, but used for different purposes and used in different ways. The Allocation Token is used to facilitate the allocation of an object instead of transferring the sponsorship of the object. The Allocation Token is not managed by the client, but is validated by the server to authorize assigning the initial sponsoring client of the object.

An example <allocationToken:allocationToken> element with value of "abc123":

```
<allocationToken:allocationToken xmlns:allocationToken=
          "urn:ietf:params:xml:ns:allocationToken-1.0">
  abc123
</allocationToken:allocationToken>
```

## 3. EPP Command Mapping

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

### 3.1. EPP Query Commands

EPP provides three commands to retrieve object information: <check> to determine if an object can be provisioned, <info> to retrieve information associated with an object, and <transfer> to retrieve object transfer status information.

#### 3.1.1. EPP <check> Command

This extension defines additional elements to extend the EPP <check> command of an object mapping like [RFC5731].

This extension allows clients to check the availability of an object with an Allocation Token, as described in Section 2.1. Clients can check if an object can be created using the Allocation Token. The Allocation Token is applied to all object names included in the EPP <check> command.

Example <check> command for the allocation.example domain name using the <allocationToken:allocationToken> extension with the allocation token of 'abc123':

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

If the query was successful, the server replies with a <check> response providing the availability status of the queried object based on the following Allocation Token cases, where the object is otherwise available:

1. If an object requires an Allocation Token and the Allocation Token does apply to the object, then the server MUST return the availability status as available (e.g., "avail" attribute is "1" or "true").
2. If an object requires an Allocation Token and the Allocation Token does not apply to the object, then the server SHOULD return the availability status as unavailable (e.g., "avail" attribute is "0" or "false").
3. If an object does not require an Allocation Token, the server MAY return the availability status as available (e.g., "avail" attribute is "1" or "true").

Example <check> domain response for a <check> command using the <allocationToken:allocationToken> extension:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S: <response>
S:  <result code="1000">
S:   <msg lang="en-US">Command completed successfully</msg>
S:  </result>
S:  <resData>
S:   <domain:chkData
S:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:    <domain:cd>
S:     <domain:name avail="1">allocation.example</domain:name>
S:    </domain:cd>
S:   </domain:chkData>
S:  </resData>
S:  <trID>
S:   <clTRID>ABC-DEF-12345</clTRID>
S:   <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

Example <check> command with the <allocationToken:allocationToken> extension for the allocation.example and allocation2.example domain names. Availability of allocation.example and allocation2.example domain names are based on the Allocation Token 'abc123':

```
C:<?xml version="1.0" encoding="UTF-8"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C: <command>
C:  <check>
C:   <domain:check
C:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:    <domain:name>allocation.example</domain:name>
C:    <domain:name>allocation2.example</domain:name>
C:   </domain:check>
C:  </check>
C:  <extension>
C:   <allocationToken:allocationToken
C:     xmlns:allocationToken=
C:       "urn:ietf:params:xml:ns:allocationToken-1.0">
C:     abc123
C:   </allocationToken:allocationToken>
C:  </extension>
C:  <clTRID>ABC-DEF-12345</clTRID>
C: </command>
C:</epp>
```

Example <check> domain response for multiple domain names in the <check> command using the <allocationToken:allocationToken> extension, where the Allocation Token 'abc123' matches allocation.example but does not match allocation2.example:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S: <response>
S:  <result code="1000">
S:   <msg lang="en-US">Command completed successfully</msg>
S:  </result>
S:  <resData>
S:   <domain:chkData
S:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:    <domain:cd>
S:     <domain:name avail="1">allocation.example</domain:name>
S:    </domain:cd>
S:    <domain:cd>
S:     <domain:name avail="0">allocation2.example</domain:name>
S:     <domain:reason>Allocation Token mismatch</domain:reason>
S:    </domain:cd>
S:   </domain:chkData>
S:  </resData>
S:  <trID>
S:   <clTRID>ABC-DEF-12345</clTRID>
S:   <svTRID>54321-XYZ</svTRID>
S:  </trID>
S: </response>
S:</epp>
```

This extension does not add any elements to the EPP <check> response described in the [RFC5730].

### 3.1.2. EPP <info> Command

This extension defines additional elements to extend the EPP <info> command of an object mapping like [RFC5731].

The EPP <info> command allows a client to request information associated with an existing object. Authorized clients MAY retrieve the Allocation Token (Section 2.1) along with the other object information by supplying the <allocationToken:info> element in the command. The <allocationToken:info> element is an empty element that serves as a marker to the server to return the <allocationToken:allocationToken> element in the info response. If the client is not authorized to receive the Allocation Token, the server MUST return an EPP error result code of 2201. If the client

is authorized to receive the Allocation Token, but there is no Allocation Token associated with the object, the server MUST return an EPP error result code of 2303. The authorization is subject to server policy.

Example <info> command with the allocationToken:info extension for the allocation.example domain name:

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

If the query was successful, the server replies with an <allocationToken:allocationToken> element along with the regular EPP <resData>. The <allocationToken:allocationToken> element is described in Section 2.1.

Example <info> domain response using the <allocationToken:allocationToken> extension:

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

### 3.1.3. EPP <transfer> Query Command

This extension does not add any elements to the EPP <transfer> query command or <transfer> query response described in [RFC5730].

### 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

This extension defines additional elements to extend the EPP <create> command of an object mapping like [RFC5731].

The EPP <create> command provides a transform operation that allows a client to create an instance of an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MUST contain a child <allocationToken:allocationToken> element for the client to be authorized to create and allocate the object. If the Allocation Token does not apply to the object, the server MUST return an EPP error result code of 2201.

Example <create> command to create a domain object with an Allocation Token:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>allocation.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken=
C:          "urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

This extension does not add any elements to the EPP <create> response described in the [RFC5730].

### 3.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the [RFC5730].

### 3.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the [RFC5730].

### 3.2.4. EPP <transfer> Command

This extension defines additional elements to extend the EPP <transfer> request command of an object mapping like [RFC5731].

The EPP <transfer> request command provides a transform operation that allows a client to request the transfer of an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MUST contain a child <allocationToken:allocationToken> element for the client to be authorized to transfer and allocate the object. The authorization associated with the Allocation Token is in addition to and does not replace the authorization mechanism defined for the object's <transfer> request command. If the Allocation Token is invalid or not required for the object, the server MUST return an EPP error result code of 2201.

Example <transfer> request command to allocate the domain object with the Allocation Token:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      <domain:transfer
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example1.tld</domain:name>
C:        <domain:period unit="y">1</domain:period>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:transfer>
C:    </transfer>
C:    <extension>
C:      <allocationToken:allocationToken
C:        xmlns:allocationToken=
C:          "urn:ietf:params:xml:ns:allocationToken-1.0">
C:        abc123
C:      </allocationToken:allocationToken>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

This extension does not add any elements to the EPP <transfer> response described in the [RFC5730].

### 3.2.5. EPP <update> Command

This extension does not add any elements to the EPP <update> command or <update> response described in the [RFC5730].

## 4. Formal Syntax

One schema is presented here that is the EPP Allocation Token Extension 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. Allocation Token Extension Schema

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:allocationToken="urn:ietf:params:xml:ns:allocationToken-1.0"
  targetNamespace="urn:ietf:params:xml:ns:allocationToken-1.0"
  elementFormDefault="qualified">
  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Allocation Token Extension
    </documentation>
  </annotation>

  <!-- Element used in info command to get allocation token. -->
  <element name="info">
    <complexType>
      <complexContent>
        <restriction base="anyType" />
      </complexContent>
    </complexType>
  </element>

  <!-- Allocation Token used in transform
    commands and info response -->
  <element name="allocationToken"
    type="allocationToken:allocationTokenType" />
  <simpleType name="allocationTokenType">
    <restriction base="token">
      <minLength value="1" />
    </restriction>
  </simpleType>

<!-- End of schema. -->
</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 allocationToken namespace:

URI: urn:ietf:params:xml:ns:allocationToken-1.0
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the allocationToken XML schema:

URI: urn:ietf:params:xml:schema:allocationToken-1.0
Registrant Contact: IESG
XML: See the "Formal Syntax" section of this document.

### 5.2. EPP Extension Registry

The following registration of the EPP Extension Registry, described in [RFC7451], is requested:

Name of Extension: "Allocation Token Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

## 6. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 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-ietf-regext-allocation-token.

Level of maturity: Production

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

### 6.2. Neustar EPP SDK

Organisation: Neustar Inc.

Name: Neustar EPP SDK

Description: The Neustar EPP SDK includes a full client implementation of draft-ietf-regext-allocation-token.

Level of maturity: Production

Coverage: All aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: quoc-anh.np@team.neustar

URL: http://registrytoolkit.neustar

### 6.3. Neustar gTLD SRS

Organisation: Neustar Inc.

Name: Neustar generic Top Level Domain (gTLD) Shared Registry System (SRS).

Description: The Neustar gTLD SRS implements the server side of draft-ietf-regext-allocation-token for several Top Level Domains.

Level of maturity: Production

Coverage: All server side aspects of the protocol are implemented.

Licensing: Proprietary

Contact: quoc-anh.np@team.neustar

### 6.4. Net::DRI

Organization: Dot and Co

Name: Net::DRI

Description: Net::DRI implements the client-side of draft-ietf-regext-allocation-token.

Level of maturity: Production

Coverage: All client-side aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: netdri@dotandco.com

## 7. Security Considerations

The mapping described in this document does 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.

The mapping acts as a conduit for the passing of Allocation Tokens between a client and a server. The definition of the Allocation Token SHOULD be defined outside of this mapping. The following are security considerations in the definition and use of an Allocation Token:

1. An Allocation Token should be considered secret information by the client and SHOULD be protected at rest and MUST be protected in transit.
2. An Allocation Token should be single use, meaning it should be unique per object and per allocation operation.
3. An Allocation Token should have a limited life with some form of expiry in the Allocation Token if generated by a trusted 3rd third party, or with a server-side expiry if generated by the server.
4. An Allocation Token should use a strong random value if it is based on an unsigned code.
5. An Allocation Token should leverage digital signatures to confirm its authenticity if generated by a trusted 3rd party.
6. An Allocation Token that is signed XML should be encoded (e.g., base64 [RFC4648]) to mitigate server validation issues.

## 8. Acknowledgements

The authors wish to acknowledge the original concept for this draft and the efforts in the initial versions of this draft by Trung Tran and Sharon Wodjenski.

Special suggestions that have been incorporated into this document were provided by Ben Campbell, Scott Hollenbeck, Benjamin Kaduk, Mirja Kuehlewind, Rubens Kuhl, Alexander Mayrhofer, Patrick Mevzek, Eric Rescoria, and Adam Roach.

## 9. References

### 9.1. Normative References

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

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

[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>.

[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

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

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

## Appendix A. Change History

### A.1. Change from 00 to 01

1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.
2. Moved Change History to the back section as an Appendix.

### A.2. Change from 01 to 02

1. Ping update.

### A.3. Change from 02 to 03

1. Ping update.

### A.4. Change from 03 to 04

1. Updated the authors for the draft.

### A.5. Change from 04 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-allocation-token to draft-ietf-regext-allocation-token.

### A.6. Change from REGEXT 00 to REGEXT 01

1. Ping update.

### A.7. Change from REGEXT 01 to REGEXT 02

1. Added the Implementation Status section.

### A.8. Change from REGEXT 02 to REGEXT 03

1. Changed Neustar author to Kal Feher.

### A.9. Change from REGEXT 03 to REGEXT 04

1. Added Neustar implementation to the Implementation Status section.

### A.10. Change from REGEXT 04 to REGEXT 05

1. Updates based on feedback from Patrick Mevzek, that include:
    1. Remove "or code" from the Abstract section.
    2. Add a missing "to" in "an allocation token TO one of the EPP..." in the Introduction section.
    3. Reword the "The allocation token is known to the server..." sentence in the Introduction section.
    4. Modify the "The allocation token MAY be returned to an authorized client for passing out-of-band to a client that uses it with an EPP transform command" to clarify who the two separate clients are.
    5. Removed an unneeded ":" from the EPP <transfer> Command and EPP <update> Command sections.

### A.11. Change from REGEXT 05 to REGEXT 06

1. Fix description of Neustar gTLD SRS based on feedback from Rubens Kuhl.
2. Updates based on feedback from Alexander Mayrhofer, that include:
    1. Making all references to Allocation Token to use the upper case form.
    2. Revise the language of the abstract to include "for including an Allocation Token in query and transform commands. The Allocation Token is used as a credential that authorizes a client to request the allocation of a specific object from the server, using one of the EPP transform commands..."
    3. Replace the title "EPP <transfer> Command" with "EPP <transfer> Query Command" for section 3.1.3.
    4. Revise the second sentence of the Introduction to "The mapping, ..., supports passing an Allocation Token..."
    5. Change "support" to "require" in the Introduction sentence "It is up to server policy which EPP transform commands and which objects support the Allocation Token."
    6. Add the definition of Allocation to the Introduction.
    7. Removed "transform" from "all of the supported EPP transform commands" in the "Allocation Token" section, since the Allocation Token can be used with the "check" command as well.
    8. Remove the word "same" from "The same <allocationToken:allocationToken> element is used for all..." in the "Allocation Token" section.
    9. Change the description of the use of the 2201 error in the "Allocation Token" section, the "EPP <create> Command" section, the "EPP <transfer> Command" section, and the "EPP <update> Command" section.
    10. Revise "<check> to determine if an object is known to the server..." to "<check> to determine if an object can be provisioned..." and remove "detailed" in the description of the <info> in the "EPP Query Commands" section.
    11. Add missing description of the expected <check> response behavior.
    12. Replaced the example reason "Invalid domain-token pair" with "Allocation Token mismatch".
    13. Replace "information on" with "information associated with" in the "EPP <info> Command" section.
    14. Removed the "that identifies the extension namespace", the ", defined in...", the Allocation Token links from the error response sentences, and the "object referencing the <allocationToken:info> element" in the "EPP <info> Command" section.

15. Added "The authorization is subject to server policy." to the "EPP <info> Command" section.
16. Replace "or <transfer> response>" with "or <transfer> query response>" in the EPP <transfer> Query Command" section.
17. Replace "create an object" with "create an instance of an object" in the "EPP <create> Command" section.
18. Revised the sentence to include "the command MUST contain a child <allocationToken:allocationToken> element for the client to be authorized to create and allocate the object" in the "EPP <create> Command" section.
19. Removed the reference to section 2.1 and the namespace identification text in the "EPP <transfer> Command" section.
20. Added "The authorization associated with the Allocation Token is in addition to and does not replace the authorization mechanism defined for the object's <transfer> request command." to the "EPP <transfer> Command" section.
21. Modified the first sentence of the "EPP Extension Registry" section to read "The following registration of the EPP Extension Registry, described in RFC7451, is requested"
22. Removed support with using the Allocation Token with an empty extension of update (e.g., release command), based on the confusion and lack of known applicability.

3. Updates based on feedback from Scott Hollenbeck, that include:

    1. Revised XML schema to included a minimum length of 1 for the allocationTokenType.
    2. Revised the "IANA Considerations" section to include the registration of the XML schema.
    3. Revised the "Security Considerations" section to include considerations for the definition of the Allocation Tokens.

### A.12. Change from REGEXT 06 to REGEXT 07

1. Updates based on feedback from Patrick Mevzek:

    1. Updated obsoleted RFC 7942 to RFC 7942.
    2. Moved RFC 7451 to an informational reference.

### A.13. Change from REGEXT 07 to REGEXT 08

1. Changed Kal Feher's contact e-mail address.
2. Changed Neustar's Implementation Status contact e-mail address.
3. Added the Net::DRI sub-section to the Implementation Status section.

### A.14. Change from REGEXT 08 to REGEXT 09

1. Updates based on the AD review by Adam Roach, that include:
    1. In "Abstract", set "query" and "transform" off in some way (e.g., using quotation marks)
    2. In "Conventions Used in This Document", please update to use the boilerplate from RFC 8174.
    3. Remove "allocationToken-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:allocationToken-1.0".
    4. In "Allocation Token", change "The server MUST have the Allocation Token" to "The server MAY have the Allocation Token".
    5. In "EPP <check> Command", change "This extension allow clients" to "This extension allows clients".
    6. Use domains reserved by RFC 2026 for the examples. The example domain "example.tld" was changed to "allocation.example" and the example domain "example2.tld" was changed to "allocation2.example".
    7. In "EPP <info> Command", change "...the server MUST return an EPP error result code of 2303 object referencing the <allocationToken:info> element." to "...the server MUST return an EPP error result code of 2303."
    8. In "EPP <transfer> Query Command", remove "the" before "RFC5730".
    9. In "EPP <transfer> Command", change "If the Allocation Token does not apply to the object..." to "If the Allocation Token is invalid or not required for the object...".
    10. In "XML Namespace", remove the sentence "The following URI assignment is requested of IANA:"
    11. In "Security Considerations", change "An Allocation Token should is" to "An Allocation Token that is". Also informatively cite RFC 4648 for the base64 reference.
2. Change "ietf:params:xml:ns:allocationToken-1.0" to "ietf:params:xml:schema:allocationToken-1.0" for the XML schema IANA registration.

### A.15. Change from REGEXT 09 to REGEXT 10

1. Changed "auhorization" to "authorization" in the "EPP <info> Command" section.
2. Added ’If an object does not require an Allocation Token, the server MAY return the availability status as available (e.g., "avail" attribute is "1" or "true").’ to the check response cases, based on feedback by Mirja Kuehlewind.
3. Changed the definition of the <info> element in the XML schema to only allow an empty element, based on IANA’s expert review.

4. Added normative language to the storage and transport of the Allocation Token, in the "Security Considerations" section, based on feedback from Eric Rescoria.
5. Changed "The definition of the Allocation Token is defined outside of this mapping" to "The definition of the Allocation Token SHOULD be defined outside of this mapping", in the "Security Considerations" section, based on feedback from Eric Rescoria.
6. Added the missing "urn:" prefix with the IANA URI registrations.
7. The URL for the BCP 14 was removed based on feedback from Alissa Cooper.
8. Updates based on review by Benjamin Kaduk, that include:

    1. Added the second paragraph to the "Allocation Token" section to describe the difference (motivation) of using the Allocation Token versus the EPP RFC authorization mechanism.
    2. Added a paragraph to the "Conventions Used in This Document" section for the use of the "abc123" token value and the use of domain object "2fooBAR" password value in the examples.
    3. Changed the "A client MUST pass an Allocation Token known to the server to be authorized to use one of the supported EPP transform commands." sentence in the "Introduction" section to "Clients pass an Allocation Token to the server for validation, and the server determines if the supplied Allocation Token is one supported by the server."
    4. Changed the "Indentation and white space in the examples are provided only to illustrate element relationships and are not REQUIRED in the protocol." sentence in the "Conventions Used in This Document" section to "Indentation and white space in the examples are provided only to illustrate element relationships and are not REQUIRED in the protocol."
    5. Changed the "Authorized clients MAY retrieve..." sentence in the "EPP <info> Command" section.
    6. Changed the "If the query was successful..." sentence in the "EPP <info> Command" section.
    7. Added "supplied" to the "If the supplied Allocation Token passed..." sentence in the "Allocation Token" section.
    8. Removed an extra newline in the <annotation> element in the "Allocation Token Extension Schema" section.

### A.16. Change from REGEXT 10 to REGEXT 11

1. Removed the old duplicate "Authorized clients MAY retrieve..." sentence from section 3.1.2 "EPP <info> Command".

### A.17. Change from REGEXT 11 to REGEXT 12

1. Revised the example <check> domain response to first include the positive case for allocation.example, and to second include the negative case for allocation2.example, based on feedback from Ben Campbell. The caption was revised for the example to include the text ", where the Allocation Token 'abc123' matches allocation.example but does not match allocation2.example".

## Authors' Addresses

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

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

Kal Feher
Neustar
lvl 8/10 Queens Road
Melbourne, VIC 3004
AU

Email: ietf@feherfamily.org
URI: http://www.neustar.biz

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 7, 2020

N. Kong
Consultant
J. Yao
L. Zhou
CNNIC
W. Tan
Cloud Registry
J. Xie

October 5, 2019

# Extensible Provisioning Protocol (EPP) Domain Name Mapping Extension for Strict Bundling Registration
## draft-ietf-regext-bundling-registration-11

## Abstract

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

## 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 https://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 7, 2020.

Table of Contents

## 1. Introduction

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

Bundled domain names normally share some attributes. Policy-wise bundling can be implemented in three ways. The first one is strict bundling, which requires all bundled names to share many same attributes. When creating, updating, or transferring of any of the bundled domain names, all bundled domain names will be created, updated or transferred atomically. The second one is partial bundling, which requires the bundled domain names to be registered by the same registrant. The third one is relaxed bundling, which has no specific requirements on the domain registration. This document mainly addresses the strict bundling names registration.

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

The basic Extensible Provisioning Protocol (EPP) domain name mapping [RFC5731] provides the facility for single domain name registration. It does not specify how to register the strict bundled names which share many of the attributes.

In order to meet the above requirements of strict bundled name registration, this document describes an extension of the EPP domain name mapping [RFC5731] for the provisioning and management of bundled names. This document describes a non-standard proprietary extension.

This extension is specially useful for registries of practising Chinese domain name registration. This document is specified using Extensible Markup Language (XML) 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

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

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

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

uLabel in this document is used to express the U-label of an internationalized domain name as a series of characters where non-ASCII characters will be represented in the format of "&#xXXXX;" where XXXX is a UNICODE point by using the XML escaping mechanism. U-Label is defined in [RFC5890].

The XML namespace prefix "b-dn" is used for the namespace "urn:ietf:params:xml:ns:epp:b-dn", but implementations MUST NOT rely on it and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents.

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

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

## 3. Definitions

The following definitions are used in this document:

- Registered Domain Name (RDN), represents the valid domain name that users submitted for the initial registration.

- Bundled Domain Name (BDN), represents the bundled domain name produced according to the bundled domain name registration policy.

## 4. Overview

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

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

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

## 5. Requirement for Bundling Registration of Names

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

- Registrar Ownership
- Registration and Expiry Dates
- Registrant, Admin, Billing, and Technical Contacts
- Name Server Association
- Domain Status

o Applicable grace periods (Add Grace Period, Renewal Grace Period, Auto-Renewal Grace Period, Transfer Grace Period, and Redemption Grace Period)

Because the domain names are bundled and share the same parameters or attributes, the EPP command should do some processing for these requirements:
o When performing a domain check, either BDN or RDN can be queried for the EPP command, and will return the same response.
o When performing a domain info, either BDN or RDN can be queried, the same response will include both BDN and RDN information with the same attributes.
o When performing a domain Create, either of the bundle names will be accepted. If the domain name is available, both BDN and RDN will be registered.
o When performing a domain Delete, either BDN or RDN will be accepted. If the domain name is registered, both BDN and RDN will be deleted.
o When performing a domain renew, either BDN or RDN will be accepted. Upon a successful domain renewal, both BDN and RDN will have their expiry date extended by the requested term. Upon a successful domain renewal, both BDN and RDN will conform to the same renew grace period.
o When performing a domain transfer, either BDN or RDN will be accepted. Upon successful completion of a domain transfer request, both BDN and RDN will enter a pendingTransfer status. Upon approval of the transfer request, both BDN and RDN will be owned and managed by the same new registrant.
o When performing a domain update, either BDN or RDN will be accepted. Any modifications to contact associations, name server associations, domain status values and authorization information will be applied to both BDN and RDN.

## 6. Object Attributes

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

### 6.1. RDN

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

For example: <b-dn:rdn uLabel="实例.example"> xn--fsq270a.example</b-dn:rdn>

### 6.2. BDN

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

For example: <b-dn:bdn uLabel="實例.example"> xn--fsqz41a.example</b-dn:bdn>

## 7. EPP Command Mapping

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

### 7.1. EPP Query Commands

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

#### 7.1.1. EPP <check> Command

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

```
Example <check> response:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:chkData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:          <domain:cd>
S:           <domain:name avail="1">
S:            xn--fsq270a.example</domain:name>
S:          </domain:cd>
S:          <domain:cd>
S:            <domain:name avail="1">
S:              xn--fsqz41a.example
S:            </domain:name>
S:            <domain:reason>This associated domain name is
S:              a produced name based on bundle name policy.
S:            </domain:reason>
S:          </domain:cd>
S:      </domain:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

#### 7.1.2. EPP <info> Command

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

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

- An <b-dn:rdn> element that contains the RDN, along with the attribute described below.

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

The above elements contain the following attribute:

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

Example <info> response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--fsq270a.example</domain:name>
S:        <domain:roid>58812678-domain</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>123</domain:registrant>
S:        <domain:contact type="admin">123</domain:contact>
S:        <domain:contact type="tech">123</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.cn
S:          </domain:hostObj>
S:        </domain:ns>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2011-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2012-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <b-dn:infData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
```

```
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

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

### 7.1.3. EPP <transfer> Query Command

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

## 7.2. EPP Transform Commands

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

When theses commands have been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP domain mapping [RFC5731]. Unless some registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:bundle> which has the following child elements:

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

The above elements contain the following attribute:

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

#### 7.2.1. EPP <create> Command

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

In addition to the EPP command elements described in the EPP domain mapping [RFC5731], the <create> command SHALL contain an <extension> element. Unless some registration policy has some special processing, the <extension> element SHOULD contain a child <b-dn:create> element that identifies the bundle namespace, and a child <b-dn:rdn> element that identifies the U-Label form of the registered domain name with the uLabel attribute.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>xn--fsq270a.example</domain:name>
C:        <domain:period unit="y">2</domain:period>
C:        <domain:registrant>123</domain:registrant>
C:        <domain:contact type="admin">123</domain:contact>
C:        <domain:contact type="tech">123</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <b-dn:create
C:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
C:        <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
C:          xn--fsq270a.example
C:        </b-dn:rdn>
C:      </b-dn:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

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:      <domain:creData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--fsq270a.example</domain:name>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:creData>
S:    </resData>
S:    <extension>
S:      <b-dn:creData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example" >
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:creData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

### 7.2.2. EPP <delete> Command

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

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

Example <delete> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <b-dn:delData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:delData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

### 7.2.3. EPP <renew> Command

This extension does not add any element to the EPP <renew> command described in the EPP domain name mapping [RFC5731]. However, when either RDN or BDN is sent for renew, response SHOULD contain both RDN and BDN information. When the command has been processed successfully, the EPP <extension> element SHALL be contained in the response if the domain object has data associated with bundled names. Unless some registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:renData> which contains <b-dn:bundle> element.

Example <renew> 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:      <domain:renData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--fsq270a.example</domain:name>
S:        <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:renData>
S:    </resData>
S:    <extension>
S:      <b-dn:renData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example" >
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:renData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

#### 7.2.4. EPP <transfer> Command

This extension does not add any element to the EPP <transfer> command described in the EPP domain name mapping [RFC5731]. However, additional elements are defined for the <transfer> response in the EPP object mapping. When the command has been processed successfully, the EPP <extension> element SHALL be contained in the response if the domain object has data associated with bundled names. Unless some registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:trnData> which contains <b-dn:bundle> element.

Example <transfer> 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="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <domain:trnData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>xn--fsq270a.example</domain:name>
S:        <domain:trStatus>pending</domain:trStatus>
S:        <domain:reID>ClientX</domain:reID>
S:        <domain:reDate>2011-04-03T22:00:00.0Z</domain:reDate>
S:        <domain:acID>ClientY</domain:acID>
S:        <domain:acDate>2011-04-08T22:00:00.0Z</domain:acDate>
S:        <domain:exDate>2012-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:trnData>
S:    </resData>
S:    <extension>
S:      <b-dn:trnData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example">
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:trnData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

### 7.2.5. EPP <update> Command

This extension does not add any element to the EPP <update> command described in the EPP domain name mapping [RFC5731]. However, additional elements are defined for the <update> response in the EPP object mapping. When the command has been processed successfully, the EPP <extension> element SHALL be contained in the response if the domain object has data associated with bundled names. Unless some

registration policy has some special processing, this EPP <extension> element SHOULD contain the <b-dn:upData> which contains <b-dn:bundle> element.

Example <update> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <extension>
S:      <b-dn:upData
S:        xmlns:b-dn="urn:ietf:params:xml:ns:epp:b-dn">
S:        <b-dn:bundle>
S:          <b-dn:rdn uLabel="&#x5B9E;&#x4F8B;.example" >
S:            xn--fsq270a.example
S:          </b-dn:rdn>
S:          <b-dn:bdn uLabel="&#x5BE6;&#x4F8B;.example">
S:            xn--fsqz41a.example
S:          </b-dn:bdn>
S:        </b-dn:bundle>
S:      </b-dn:upData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

## 8. Formal Syntax

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

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

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

```
    elementFormDefault="qualified">

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

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Bundle Domain Extension Schema v1.0
    </documentation>
  </annotation>

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

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

  <!--
    Child response elements in <b-dn:infData>, <b-dn:delData>,
    <b-dn:creData>, <b-dn:renData>, <b-dn:trnData> and <b-dn:upData>.
  -->
  <element name="infData" type="b-dn:bundleDataType"/>
  <element name="delData" type="b-dn:bundleDataType"/>
  <element name="creData" type="b-dn:bundleDataType"/>
  <element name="renData" type="b-dn:bundleDataType"/>
  <element name="trnData" type="b-dn:bundleDataType"/>
  <element name="upData" type="b-dn:bundleDataType"/>

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

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

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

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

END
```

## 9. Internationalization Considerations

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

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

## 10. IANA Considerations

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

Registration request for the IDN namespace:

- URI: urn:ietf:params:xml:ns:epp:b-dn

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

- XML: None. Namespace URI does not represent an XML specification.

Registration request for the IDN XML schema:

- URI: urn:ietf:params:xml:schema:epp:b-dn

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

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

The EPP extension described in this document should be registered by IANA in the "Extensions for the Extensible Provisioning Protocol (EPP)" registry described in [RFC7451]. The details of the registration are as follows:

- Name of Extension: "Domain Name Mapping Extension for Strict Bundling Registration"

- Document status: Informational

- Reference: This document

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

- Top-Level Domains (TLDs): Any

- IPR Disclosure: https://datatracker.ietf.org/ipr/

- Status: Active

- Notes: None

## 11. Security Considerations

Some registries and registrars have more than 15 years of the bundled registration of domain names (especially Chinese domain names). They have not found any significant security issues. One principle that the registry and registrar should let the registrants know is that bundled registered domain names will be created, transferred, updated, and deleted together as a group. The registrants for bundled domain names should remember this principle when doing some operations to these domain names. [RFC5730] also introduces some security consideration.

This document does not take a position regarding whether or not the bundled domain names share a DS/DNSKEY key. The DNS administrator can choose whether DS/DNSKEY information can be shared or not. If a DS/DNSKEY key is shared then the bundled domain names share fate if there is a key compromise.

## 12. Implementation Status

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

- The Chinese Domain Name Consortium(CDNC) including CNNIC, TWNIC, HKIRC, MONIC, SGNIC and more have followed the principles defined in this document for many years.

- CNNIC and TELEINFO have implemented this extension in their EPP based Chinese domain name registration system.

- Public Interest Registry, has requested to implement technical bundling of second level domains for .NGO and .ONG. This means that by registering and purchasing a domain in the .ngo TLD, for an example, the NGO registrant is also registering and purchasing the corresponding name in the .ong TLD (and vice-versa for registrations in .ong).

- Patrick Mevzek has released a new version of Net::DRI, an EPP client (Perl library, free software) implementing this extension.

## 13. Acknowledgements

The authors especially thank the authors of [RFC5730] and [RFC5731] and the following ones of CNNIC: Weiping Yang, Chao Qi.

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

## 14. Change History

RFC Editor: Please remove this section.

### 14.1. draft-ietf-regext-bundle-registration: Version 00

- accepted as WG document.

### 14.2. draft-ietf-regext-bundle-registration: Version 01

- make this document to focus on the restrict bundled domain name registration.

14.3. draft-ietf-regext-bundle-registration: Version 02

- Update the section of implementation status.

14.4. draft-ietf-regext-bundle-registration: Version 03

- This document is changed to informational category.
- Refine the text.

14.5. draft-ietf-regext-bundle-registration: Version 04

- Update the implementation section.
- Refine the text.

14.6. draft-ietf-regext-bundle-registration: Version 05

- Scope the XML namespaces to include 'epp'.

14.7. draft-ietf-regext-bundle-registration: Version 06

- add some examples for the transfer, update and renew command
- add some text to security consideration

14.8. draft-ietf-regext-bundle-registration: Version 07

- Update IANA consideration section based on Scott's comments
- Update security consideration based on Chair and Patrick Mevzek's comments

14.9. draft-ietf-regext-bundle-registration: Version 08

- Refine some texts.

14.10. draft-ietf-regext-bundle-registration: Version 09

- Refine the texts.

14.11. draft-ietf-regext-bundle-registration: Version 10

- Update the texts based on IETF LC.

### 14.12. draft-ietf-regext-bundle-registration: Version 11

- Update the texts based on AD's comment.

## 15. References

### 15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <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>.

[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>.

[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>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, ""Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204", February 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

[W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, ""XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028", October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

[W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, ""XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028", October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

## 15.2. Informative References

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

## Authors' Addresses

Ning Kong
Consultant

Email: ietfing@gmail.com

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

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

Linlin Zhou
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 2677
Email: zhoulinlin@cnnic.cn

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

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

Jiagui Xie

Email: jiagui1984@163.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 8, 2019

J. Gould
VeriSign, Inc.
K. Feher
Neustar

January 4, 2019

# Change Poll Extension for the Extensible Provisioning Protocol (EPP) draft-ietf-regext-change-poll-12

## Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension for notifying clients of operations on client-sponsored objects that were not initiated by the client through EPP. These operations may include contractual or policy requirements including but not limited to regular batch processes, customer support actions, Uniform Domain-Name Dispute-Resolution Policy (UDRP) or Uniform Rapid Suspension (URS) actions, court-directed actions, and bulk updates based on customer requests. Since the client is not directly involved or knowledgable of these operations, the extension is used along with an EPP object mapping to provide the resulting state of the post-operation object, and optionally a pre-operation object, with the operation meta-data of what, when, who, and why.

## Status of This Memo

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

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

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

This Internet-Draft will expire on July 8, 2019.

Table of Contents

## 1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], is used to notify clients of operations they are not directly involved in, on objects that the client sponsors. It is up to server policy to determine what transform operations and clients to notify. Using this extension, clients can more easily keep their systems in-sync with the objects stored in the server. When a change occurs that a client needs to be notified of, a poll message can be inserted by the server for consumption by the client using the EPP <poll> command and response defined in [RFC5730]. The extension supports including a "before" operation poll message and an "after" operation poll message. The extension only extends the EPP <poll> response in [RFC5730] and does not extend the EPP <poll> command. Please refer to [RFC5730] for information and examples of the EPP <poll> command.

### 1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

## 2. Object Attributes

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

### 2.1. Operation

An operation consists of any transform operation that impacts objects that the client sponsers and should be notified of. The <changePoll:operation> element defines the operation. The OPTIONAL "op" attribute is an identifier, represented in the 7-bit US-ASCII character set defined in [RFC0020], that is used to define a sub-operation or the name of a "custom" operation. The enumerated list of <changePoll:operation> values is:

"create": Create operation as defined in [RFC5730].
"delete": Delete operation as defined in [RFC5730]. If the delete operation results in an immediate purge of the object, then the "op" attribute MUST be set to "purge".
"renew": Renew operation as defined in [RFC5730].
"transfer": Transfer operation as defined in [RFC5730] that MUST set the "op" attribute with one of the possible transfer type values that include "request", "approve", "cancel", or "reject".
"update": Update operation as defined in [RFC5730].
"restore": Restore operation as defined in [RFC3915] that MUST set the "op" attribute with one of the possible restore type values that include "request" or "report".
"autoRenew": Auto renew operation executed by the server.
"autoDelete": Auto delete operation executed by the server. If the "autoDelete" operation results in an immediate purge of the object, then the "op" attribute MUST be set to "purge".
"autoPurge": Auto purge operation executed by the server when removing the object after it had the "pendingDelete" status.

"custom": Custom operation that MUST set the "op" attribute with the custom operation name. The custom operations supported is up to server policy.

## 2.2. State

The state attribute reflects the state of the object "before" or "after" the operation. The state is defined using the OPTIONAL "state" attribute of the <changePoll:changeData> element, with the possible values "before" or "after" and with a default value of "after". The server MAY support both the "before" state and the "after" state of the operation, by using one poll message for the "before" state and one poll message for the "after" state. The "before" state poll message MUST be inserted into the message queue prior to the "after" state poll message.

For operations in Section 2.1 that don't have an "after" state, the server MUST use the "before" state poll message. For example, for the "delete" operation with the "op" attribute set to "purge", or the "autoPurge" operation, the server includes the state of the object prior to being purged in the "before" state poll message.

For operations in Section 2.1 that don't have a "before" state, the server MUST use the "after" state poll message. For example, for the "create" operation, the server includes the state of the object after creation in the "after" state poll message.

## 2.3. Who

The <changePoll:who> element defines who executed the operation for audit purposes. It is a freeform value that is strictly meant for audit purposes and not meant to drive client-side logic. The scheme used for the possible set of <changePoll:who> element values is up to server policy. The server MAY identify the <changePoll:who> element value based on:

"Identifier": Unique user identifier of the user that executed the operation. An example is "ClientX".
"Name": Name of the user that executed the operation. An example is "John Doe".
"Role": Role of the user that executed operation. An example is "CSR" for a Customer Support Representative or "Batch" for a server batch.

### 2.4. 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 [W3C.REC-xmlschema-2-20041028] 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.

## 3. EPP Command Mapping

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

### 3.1. EPP Query Commands

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

#### 3.1.1. EPP <check> Command

This extension does not add any elements to the EPP <check> command or <check> response described in the [RFC5730].

#### 3.1.2. EPP <info> Command

This extension does not add any elements to the EPP <info> command described in the [RFC5730].

This extension adds operation detail of EPP object mapping operations Section 2.1 to an EPP poll response, as described in [RFC5730]. The extension is an extension of the EPP object mapping info response. Any transform operation to an object defined in an EPP object mapping by a client other than the sponsoring client MAY result in extending the <info> response of the object for inserting an EPP poll message with the operation detail. The sponsoring client will then receive the state of the object with operation detail like what, who, when, and why the object was changed. The <changePoll:changeData> element contains the operation detail along with an indication of whether the object reflects the state before or after the operation as defined in Section 2.2. The <changePoll:changeData> element includes the operation detail with the following child elements:

<changePoll:operation>: Transform operation executed on the object as defined in Section 2.1.
<changePoll:date>: Date and time when the operation was executed.

<changePoll:svTRID>: Server transaction identifier of the operation.
<changePoll:who>: Who executed the operation as defined in Section 2.3.
<changePoll:caseId>: OPTIONAL case identifer associated with the operation. The required "type" attribute defines the type of case. The OPTIONAL "name" attribute is an identifier, represented in the 7-bit US-ASCII character set defined in [RFC0020], that is used to define the name of the "custom" case type. The enumerated list of case types is:

udrp: a Uniform Domain-Name Dispute-Resolution Policy (UDRP) case.
urs: a Uniform Rapid Suspension (URS) case.
custom: A custom case that is defined using the "name" attribute.

<changePoll:reason>: OPTIONAL reason for executing the operation. If present, this element contains the server-specific text to help explain the reason the operation was executed. 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 the default value of "en" (English).

Example poll <info> response with the <changePoll:changeData> extension for a URS lock transaction on the domain.example domain name, with the "before" state. The "before" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:      <result code="1301">
S:         <msg lang="en-US">
S:       Command completed successfully; ack to dequeue</msg>
S:      </result>
S:      <msgQ id="201" count="1">
S:         <qDate>2013-10-22T14:25:57.0Z</qDate>
S:         <msg>Registry initiated update of domain.</msg>
S:      </msgQ>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:        state="before">
S:        <changePoll:operation>update</changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:        <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:        <changePoll:who>URS Admin</changePoll:who>
S:        <changePoll:caseId type="urs">urs123</changePoll:caseId>
S:        <changePoll:reason>URS Lock</changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:   </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for a URS lock transaction on the domain.example domain name, with the "after" state. The "after" state is reflected in the

<resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:      <result code="1301">
S:         <msg lang="en-US">
S:       Command completed successfully; ack to dequeue</msg>
S:      </result>
S:      <msgQ id="202" count="1">
S:         <qDate>2013-10-22T14:25:57.0Z</qDate>
S:         <msg>Registry initiated update of domain.</msg>
S:      </msgQ>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="serverUpdateProhibited"/>
S:        <domain:status s="serverDeleteProhibited"/>
S:        <domain:status s="serverTransferProhibited"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientZ</domain:upID>
S:        <domain:upDate>2013-10-22T14:25:57.0Z</domain:upDate>
S:        <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:        state="after">
S:        <changePoll:operation>update</changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:        <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:        <changePoll:who>URS Admin</changePoll:who>
S:        <changePoll:caseId type="urs">urs123</changePoll:caseId>
S:        <changePoll:reason>URS Lock</changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:   </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for a custom "sync" operation on the domain.example domain name, with the default "after" state. The "after" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:      <result code="1301">
S:         <msg>Command completed successfully; ack to dequeue</msg>
S:      </result>
S:      <msgQ id="201" count="1">
S:         <qDate>2013-10-22T14:25:57.0Z</qDate>
S:    <msg>Registry initiated Sync of Domain Expiration Date</msg>
S:      </msgQ>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2012-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientZ</domain:upID>
S:        <domain:upDate>2013-10-22T14:25:57.0Z</domain:upDate>
S:        <domain:exDate>2014-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:        <changePoll:operation op="sync">custom
S:        </changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:        <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:        <changePoll:who>CSR</changePoll:who>
S:        <changePoll:reason lang="en">Customer sync request
S:        </changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:   </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for a "delete" operation on the domain.example domain name that is immediately purged, with the "before" state. The "before" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ id="200" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry initiated delete of
S:         domain resulting in immediate purge.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:clID>ClientX</domain:clID>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:        state="before">
S:        <changePoll:operation op="purge">delete
S:        </changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z
S:        </changePoll:date>
S:        <changePoll:svTRID>12345-XYZ
S:        </changePoll:svTRID>
S:        <changePoll:who>ClientZ
S:        </changePoll:who>
S:        <changePoll:reason>Court order
S:        </changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for an "autoPurge" operation on the domain.example domain name that previously had the "pendingDelete" status, with the "before" state. The "before" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue
S:      </msg>
S:    </result>
S:    <msgQ id="200" count="1">
S:      <qDate>2013-10-22T14:25:57.0Z</qDate>
S:      <msg>Registry purged domain with pendingDelete status.
S:      </msg>
S:    </msgQ>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:clID>ClientX</domain:clID>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0"
S:        state="before">
S:        <changePoll:operation>autoPurge
S:        </changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z
S:        </changePoll:date>
S:        <changePoll:svTRID>12345-XYZ
S:        </changePoll:svTRID>
S:        <changePoll:who>Batch
S:        </changePoll:who>
S:        <changePoll:reason>Past pendingDelete 5 day period
S:        </changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example poll <info> response with the <changePoll:changeData> extension for an "update" operation on the ns1.domain.example host, with the default "after" state. The "after" state is reflected in the <resData> block:

```
S:<?xml version="1.0" encoding="UTF-8"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:      <result code="1301">
S:         <msg>Command completed successfully; ack to dequeue</msg>
S:      </result>
S:      <msgQ id="201" count="1">
S:         <qDate>2013-10-22T14:25:57.0Z</qDate>
S:         <msg>Registry initiated update of host.</msg>
S:      </msgQ>
S:    <resData>
S:      <host:infData
S:       xmlns:host="urn:ietf:params:xml:ns:host-1.0">
S:        <host:name>ns1.domain.example</host:name>
S:        <host:roid>NS1_EXAMPLE1-REP</host:roid>
S:        <host:status s="linked"/>
S:        <host:status s="serverUpdateProhibited"/>
S:        <host:status s="serverDeleteProhibited"/>
S:        <host:addr ip="v4">192.0.2.2</host:addr>
S:        <host:addr ip="v6">2001:db8:0:0:1:0:0:1</host:addr>
S:        <host:clID>ClientX</host:clID>
S:        <host:crID>ClientY</host:crID>
S:        <host:crDate>2012-04-03T22:00:00.0Z</host:crDate>
S:        <host:upID>ClientY</host:upID>
S:        <host:upDate>2013-10-22T14:25:57.0Z</host:upDate>
S:      </host:infData>
S:    </resData>
S:    <extension>
S:      <changePoll:changeData
S:        xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
S:        <changePoll:operation>update</changePoll:operation>
S:        <changePoll:date>2013-10-22T14:25:57.0Z</changePoll:date>
S:        <changePoll:svTRID>12345-XYZ</changePoll:svTRID>
S:        <changePoll:who>ClientZ</changePoll:who>
S:        <changePoll:reason>Host Lock</changePoll:reason>
S:      </changePoll:changeData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:   </response>
S:</epp>
```

#### 3.1.3. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> query command or <transfer> response described in the [RFC5730].

### 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

This extension does not add any elements to the EPP <create> command or <create> response described in the [RFC5730].

#### 3.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the [RFC5730].

#### 3.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the [RFC5730].

#### 3.2.4. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the [RFC5730].

#### 3.2.5. EPP <update> Command

This extension does not add any elements to the EPP <update> command or <update> response described in the [RFC5730].

## 4. Formal Syntax

One schema is presented here that is the EPP Change Poll Extension 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. Change Poll Extension Schema

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

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


   <annotation>
     <documentation>
       Extensible Provisioning Protocol v1.0
       Change Poll Mapping Schema.
     </documentation>
   </annotation>

   <!--
    Change element.
   -->
   <element name="changeData" type="changePoll:changeDataType"/>

   <!--
    Attributes associated with the change.
   -->
   <complexType name="changeDataType">
     <sequence>
       <element name="operation" type="changePoll:operationType"/>
       <element name="date" type="dateTime"/>
       <element name="svTRID" type="epp:trIDStringType"/>
       <element name="who" type="changePoll:whoType"/>
       <element name="caseId" type="changePoll:caseIdType"
         minOccurs="0"/>
       <element name="reason" type="eppcom:reasonType"
         minOccurs="0"/>
     </sequence>
    <attribute name="state" type="changePoll:stateType"
      default="after"/>
   </complexType>
```

```
      <!--
       Enumerated list of operations, with extensibility via "custom".
      -->
      <simpleType name="operationEnum">
        <restriction base="token">
          <enumeration value="create"/>
          <enumeration value="delete"/>
          <enumeration value="renew"/>
          <enumeration value="transfer"/>
          <enumeration value="update"/>
          <enumeration value="restore"/>
          <enumeration value="autoRenew"/>
          <enumeration value="autoDelete"/>
          <enumeration value="autoPurge"/>
          <enumeration value="custom"/>
        </restriction>
      </simpleType>

      <!--
       Enumerated of state of the object in the poll message.
      -->
      <simpleType name="stateType">
        <restriction base="token">
          <enumeration value="before"/>
          <enumeration value="after"/>
        </restriction>
      </simpleType>

      <!--
       Transform operation type
      -->
     <complexType name="operationType">
       <simpleContent>
         <extension base="changePoll:operationEnum">
           <attribute name="op" type="token"/>
         </extension>
       </simpleContent>
     </complexType>

      <!--
       Case identifier type
      -->
     <complexType name="caseIdType">
       <simpleContent>
         <extension base="token">
           <attribute name="type" type="changePoll:caseTypeEnum"
             use="required"/>
           <attribute name="name" type="token"
```

```
            use="optional"/>
        </extension>
      </simpleContent>
    </complexType>

    <!--
     Enumerated list of case identifier types
    -->
    <simpleType name="caseTypeEnum">
      <restriction base="token">
        <enumeration value="udrp"/>
        <enumeration value="urs"/>
        <enumeration value="custom"/>
      </restriction>
    </simpleType>

    <!--
     Who type
    -->
    <simpleType name="whoType">
      <restriction base="normalizedString">
        <minLength value="1"/>
        <maxLength value="255"/>
      </restriction>
    </simpleType>

   <!--
   End of schema.
   -->
   </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]. The following URI assignment is requested of IANA:

Registration request for the changePoll namespace:

- URI: urn:ietf:params:xml:ns:changePoll-1.0
- Registrant Contact: IESG
- XML: None. Namespace URIs do not represent an XML specification.

Registration request for the changePoll XML schema:

URI: urn:ietf:params:xml:ns:changePoll-1.0
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: "Change Poll Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

## 6. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 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-ietf-regext-change-poll.

Level of maturity: Production

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

### 6.2. Verisign Consolidated Top Level Domain (CTLD) SRS

Organization: Verisign Inc.

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

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

Level of maturity: Production

Coverage: The "after" state poll message for an "update" transform operation of a domain name due to server policy.

Licensing: Proprietary

Contact: jgould@verisign.com

### 6.3. Verisign .COM / .NET SRS

Organization: Verisign Inc.

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

Description: The Verisign Shared Registry System (SRS) for .COM and .NET implements the server-side of draft-ietf-regext-change-poll.

Level of maturity: Production

Coverage: The "after" state poll message for an "update" transform operation of a domain name due to server policy.

Licensing: Proprietary

Contact: jgould@verisign.com

### 6.4. Neustar EPP SDK

Organisation: Neustar Inc.

Name: Neustar EPP SDK

Description: The Neustar EPP SDK includes a full client implementation of draft-ietf-regext-change-poll.

Level of maturity: Production

Coverage: All client side aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: quoc-anh.np@team.neustar

## 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 acknowledge the original concept for this draft and the efforts in the initial versions of this draft by Trung Tran and Sharon Wodjenski.

Special suggestions that have been incorporated into this document were provided by Scott Hollenbeck, Michael Holloway, and Patrick Mevzek.

## 9. References

### 9.1. Normative References

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[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>.

[RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <https://www.rfc-editor.org/info/rfc3915>.

[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>.

[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>.

[W3C.REC-xmlschema-2-20041028] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

### 9.2. Informative References

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

## Appendix A. Change History

### A.1. Change from 00 to 01

1. Added an optional caseId element that defines the case identifier from UDRP, URS, or custom case, based on feedback from Michael Holloway.

### A.2. Change from 01 to 02

1. Amended XML Namespace section of IANA Considerations, added EPP Extension Registry section.
2. Moved Change History to the back section as an Appendix.

### A.3. Change from 02 to 03

1. Fixed "before" state example to use the "before" state value based on feedback from Patrick Mevzek.

### A.4. Change from 03 to 04

1. Updated the authors for the draft.

### A.5. Change from 04 to 05

1. Ping update.

### A.6. Change from 05 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-change-poll to draft-ietf-regext-change-poll.

### A.7. Change from REGEXT 00 to REGEXT 01

1. Ping update.

### A.8. Change from REGEXT 01 to REGEXT 02

1. Added the Implementation Status section.

### A.9. Change from REGEXT 02 to REGEXT 03

1. Changed Neustar author to Kal Feher.

### A.10. Change from REGEXT 03 to REGEXT 04

1. Added Neustar implementation to the Implementation Status section.

### A.11. Change from REGEXT 04 to REGEXT 05

1. Updates based on feedback from Patrick Mevzek, that include:

    1. Added a missing comma to "Using this extension, clients" in the Introduction section.
    2. Modified the description of the "transfer", "restore", and "custom" operations to include "MUST set the "op" attribute" language.
    3. Rephrased the first sentence of the Who section.
    4. Added references to the <changePoll:who> element in the Who section.
    5. Revise the sentence that describes how the extension extends the info response in the EPP <info> Command section.
    6. Refer to EPP Object Mapping as EPP object mapping throughout the document.
    7. Add a Dates and Times section to the Object Attributes section.

### A.12. Change from REGEXT 05 to REGEXT 06

1. Added the "State" sub-section to the "Object Attributes" section to describe the expected behavior for the "before" and "after" states, based on feedback from Patrick Mevzek.
2. Added a colon suffix to each hangText entry to provide better separation.

### A.13. Change from REGEXT 06 to REGEXT 07

1. Updates based on feedback from Scott Hollenbeck, that include:

    1. Changed MAY to may in the Abstract.
    2. Revised the "IANA Considerations" section to include the registration of the XML schema.

3. Revised the description of the <changePoll:caseId> "name" attribute and the "changePoll:operation> "op" attribute as containing 7-bit US-ASCII identifiers for the case type or the operation type, respectively.

A.14. Change from REGEXT 07 to REGEXT 08

1. Updated obsoleted RFC 6982 to RFC 7942.
2. Moved RFC 7451 to an informational reference based on a check done by the Idnits Tool.
3. Changed Kal Feher's contact e-mail address.
4. Changed Neustar's Implementation Status contact e-mail address.

A.15. Change from REGEXT 08 to REGEXT 09

1. Fixed Section 1.1 (Conventions) to contain the updated language (e.g. "NOT RECOMMENDED", RFC 8174, BCP 14), based on feedback from the Document Shepherd.

A.16. Change from REGEXT 09 to REGEXT 10

1. Updates based on the AD review by Adam Roach, that include:

    1. Fix the "purge" and "autoPurge" examples to use the normative "before" state instead of the default "after" state.
    2. Added the sentences "The extension only extends the EPP <poll> response in [RFC5730] and does not extend the EPP <poll> command. Please refer to [RFC5730] for information and examples of the EPP <poll> command." in the "Introduction" to clarify what is extended and reference [RFC5730] for the EPP <poll> command.
    3. Added missing hyphens to "client-sponsored" and "court-directed".
    4. Removed "changePoll-1.0" is used as an abbreviation for "urn:ietf:params:xml:ns:changePoll-1.0" and replaced the paragraph based on what was done in draft-ietf-regext-allocation-token.
    5. Changed normative "SHOULD" to non-normative "should" in "An operation consists of any transform operation that impacts objects that the client sponsers and should be notified of."
    6. Added normative reference to [RFC0020] to define "7-bit US-ASCII".
    7. Added the sentence "The custom operations supported is up to server policy." to the description of the "custom" operation.

8. Broke up the "This extension adds operation detail..." sentence into two seperate sentences to address the "does" and the "is" seperately.
9. Removed the commas from "Any transform operation to an object..." sentence.
10. Changed to use an IPv6 address from the documentation-only prefix "2001:DB8::/32" in RFC 3849. The IPv6 address 2001:db8:0:0:1:0:0:1 was used.

## A.17. Change from REGEXT 10 to REGEXT 11

1. Updates based on the review by Benjamin Kaduk, that include:

    1. Change references of "The enumerated list ... include:" to "The enumerated list ... is:".
    2. In section 2.2, explicitly state what the message is inserted into, with the change of "... MUST be inserted prior to ..." to "... MUST be inserted into the message queue prior to ...".

## A.18. Change from REGEXT 11 to REGEXT 12

1. Added clarification for the <changePoll:who> element based on the feedback from Benjamin Kaduk.

## Authors' Addresses

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

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

Kal Feher
Neustar
lvl 8/10 Queens Road
Melbourne, VIC 3004
AU

Email: ietf@feherfamily.org
URI: http://www.neustar.biz

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2020

R. Carney
GoDaddy Inc.
G. Brown
CentralNic Group plc
J. Frakes

October 21, 2019

# Registry Fee Extension for the Extensible Provisioning Protocol (EPP) draft-ietf-regext-epp-fees-20

## Abstract

Given the expansion of the DNS namespace, and the proliferation of novel business models, it is desirable to provide a method for Extensible Provisioning Protocol (EPP) clients to query EPP servers for the fees and credits and provide expected fees and credits for certain commands and objects. This document describes an EPP extension mapping for registry fees.

## 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 https://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 23, 2020.

## Copyright Notice

Table of Contents

## 1. Introduction

Historically, domain name registries have applied a simple fee structure for billable transactions, namely a basic unit price applied to domain <create>, <renew>, <transfer> and RGP [RFC3915] restore commands. Given the relatively small number of EPP servers to which EPP clients have been required to connect, it has generally been the case that client operators have been able to obtain details of these fees out-of-band by contacting the server operators.

Given the expansion of the DNS namespace, and the proliferation of novel business models, it is desirable to provide a method for EPP clients to query EPP servers for the fees and credits associated with certain commands and specific objects.

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping provides a mechanism by which EPP clients may query the fees and credits associated with various billable transactions, and obtain their current account balance.

### 1.1. Conventions Used in This Document

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

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

XML 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.

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

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.

## 2. Migrating to Newer Versions of This Extension

Servers which implement this extension SHOULD provide a way for clients to progressively update their implementations when a new version of the extension is deployed.

Servers SHOULD (for a temporary migration period) provide support for older versions of the extension in parallel to the newest version, and allow clients to select their preferred version via the <svcExtension> element of the <login> command.

If a client requests multiple versions of the extension at login, then, when preparing responses to commands which do not include extension elements, the server SHOULD only include extension elements in the namespace of the newest version of the extension requested by the client.

When preparing responses to commands which do include extension elements, the server SHOULD only include extension elements for the extension versions present in the command.

## 3. Extension Elements

### 3.1. Client Commands

The <fee:command> element is used in the EPP <check> command to determine the fee that is applicable to the given command.

The use of the <fee:command> keys off the use of the "name" attribute to define which transform fees the client is requesting information about. Here is the list of possible values for the "name" attribute:

- "create" indicating a <create> command as defined in [RFC5730];
- "delete" indicating a <delete> command as defined in [RFC5730];
- "renew" indicating a <renew> command as defined in [RFC5730];
- "update" indicating a <update> command as defined in [RFC5730];
- "transfer" indicating a <transfer> command as defined in [RFC5730];
- If the server supports the Registry Grace Period Mapping [RFC3915], then the server MUST also support the "restore" value as defined in [RFC3915];
- "custom" indicating a custom command that MUST set the "customName" attribute with custom command name. The possible set of custom command name values is up to server policy.

The <fee:command> element MAY have an OPTIONAL "phase" attribute specifying a launch phase as described in [RFC8334]. It may also contain an OPTIONAL "subphase" attribute identifying the custom or sub-phase as described in [RFC8334].

## 3.2. Currency Codes

The <fee:currency> element is used to indicate which currency fees are charged in. This value of this element MUST be a three-character currency code from [ISO4217:2015].

Note that ISO 4217:2015 provides the special "XXX" code, which MAY be used if the server uses a non-currency based system for assessing fees, such as a system of credits.

The use of <fee:currency> elements in client commands is OPTIONAL: if a <fee:currency> element is not present in a command, the server MUST determine the currency based on the server default currency or based on the client's account settings which are agreed to by the client and server via an out-of-band channel. However, the <fee:currency> element MUST be present in responses.

Servers SHOULD NOT perform a currency conversion if a client uses an incorrect currency code. Servers SHOULD return a 2004 "Parameter value range" error instead.

## 3.3. Validity Periods

When querying for fee information using the <check> command, the <fee:period> element is used to indicate the period measured in years or months, with the appropriate units specified using the "unit"

attribute to be added to the registration period of objects by the <create>, <renew> and <transfer> commands. This element is derived from the <domain:period> element described in [RFC5731].

The <fee:period> element is OPTIONAL in <check> commands, if omitted, the server MUST determine the fee(s) using the server default period. The <fee:period> element MUST be present in <check> responses.

### 3.4. Fees and Credits

Servers which implement this extension will include elements in responses which provide information about the fees and/or credits associated with a given billable transaction. A fee will result in subtracting from the Account Balance (described in Section 3.5) and a credit will result in adding to the Account Balance (described in Section 3.5).

The <fee:fee> and <fee:credit> elements are used to provide this information. The presence of a <fee:fee> element in a response indicates a debit against the client's account balance; a <fee:credit> element indicates a credit. A <fee:fee> element MUST have a zero or greater (non-negative) value. A <fee:credit> element MUST have a negative value.

A server MAY respond with multiple <fee:fee> and <fee:credit> elements in the same response. In such cases, the net fee or credit applicable to the transaction is the arithmetic sum of the values of each of the <fee:fee> and/or <fee:credit> elements. This amount applies to the total additional validity period applied to the object (where applicable).

The following attributes are defined for the <fee:fee> element. These are described in detail below:

description: an OPTIONAL attribute which provides a human-readable description of the fee. Servers should provide documentation on the possible values of this attribute, and their meanings. An OPTIONAL "lang" attribute MAY be present, per the language structure in [RFC5646], to identify the language of the returned text and has a default value of "en" (English). If the "description" attribute is not present, the "lang" attribute can be ignored.

refundable: an OPTIONAL boolean attribute indicating whether the fee is refundable if the object is deleted.

grace-period: an OPTIONAL attribute which provides the time period during which the fee is refundable.

applied: an OPTIONAL attribute indicating when the fee will be deducted from the client's account.

The <fee:credit> element can take a "description" attribute as described above. An OPTIONAL "lang" attribute MAY be present to identify the language of the returned text and has a default value of "en" (English).

#### 3.4.1. Refunds

<fee:fee> elements MAY have an OPTIONAL "refundable" attribute which takes a boolean value. Fees may be refunded under certain circumstances, such as when a domain application is rejected (as described in [RFC8334]) or when an object is deleted during the relevant Grace Period (see below).

If the "refundable" attribute is omitted, then clients SHOULD NOT make any assumption about the refundability of the fee.

#### 3.4.2. Grace Periods

[RFC3915] describes a system of "grace periods", which are time periods following a billable transaction during which, if an object is deleted, the client receives a refund.

The "grace-period" attribute MAY be used to indicate the relevant grace period for a fee. If a server implements the Registry Grace Period extension [RFC3915], it MUST specify the grace period for all relevant transactions.

If the "grace-period" attribute is omitted, then clients SHOULD NOT make any assumption about the grace period of the fee.

#### 3.4.3. Correlation between Refundability and Grace Periods

If a <fee:fee> element has a "grace-period" attribute then it MUST also be refundable and the "refundable" attribute MUST be true. If the "refundable" attribute of a <fee:fee> element is false then it MUST NOT have a "grace-period" attribute.

#### 3.4.4. Applicability

Fees may be applied immediately upon receipt of a command from a client, or may only be applied once an out-of-band process (such as the processing of applications at the end of a launch phase) has taken place.

The "applied" attribute of the <fee:fee> element allows servers to indicate whether a fee will be applied immediately, or whether it will be applied at some point in the future. This attribute takes two possible values: "immediate" or "delayed".

### 3.5. Account Balance

The <fee:balance> element is an OPTIONAL element which MAY be included in server responses to transform commands. If present, it can be used by the client to determine the remaining credit at the server.

Whether or not the <fee:balance> is included in responses is a matter of server policy. However, if a server chooses to offer support for this element, it MUST be included in responses to all "transform" or billable commands (e.g. <create>, <renew>, <update>, <delete>, <transfer op="request">).

The value of the <fee:balance> MAY be negative. A negative balance indicates that the server has extended a line of credit to the client (see below).

If a server includes a <fee:balance> element in response to transform commands, the value of the element MUST reflect the client's account balance after any fees or credits associated with that command have been applied. If the "applied" attribute of the <fee:fee> element is "delayed", then the <fee:balance> MUST reflect the client's account balance without any fees or credits associated with that command.

### 3.6. Credit Limit

As described above, if a server returns a response containing a <fee:balance> with a negative value, then the server has extended a line of credit to the client. A server MAY also include a <fee:creditLimit> element in responses that indicates the maximum credit available to a client. A server MAY reject certain transactions if the absolute value of the <fee:balance> is equal to or exceeds the value of the <fee:creditLimit> element.

Whether or not the <fee:creditLimit> is included in responses is a matter of server policy. However, if a server chooses to offer support for this element, it MUST be included in responses to all "transform" commands (e.g. <create>, <renew>, <update>, <delete>, <transfer op="request">).

### 3.7. Classification of Objects

Objects may be assigned to a particular class, category, or tier, each of which has a particular fee or set of fees associated with it. The <fee:class> element, which MAY appear in <check> and transform responses, is used to indicate the classification of an object.

If a server makes use of this element, it should provide clients with a list of all the values that the element may take via an out-of-band channel. Servers MUST NOT use values which do not appear on this list.

Servers that make use of this element MUST use a <fee:class> element with the value "standard" for all objects that are subject to the standard or default fee.

### 3.8. Phase and Subphase Attributes

The <fee:command> element has two attributes, phase and subphase, that provide additional information related to a specific launch phase as described in [RFC8334]. These attributes are used as filters that should refine the server processing.

If the client <fee:command> contains a server supported combination of phase/subphase the server MUST return fee data (including the phase/subphase attribute(s)) for the specific combination.

If the client <fee:command> contains no phase/subphase attributes and the server has only one active phase/subphase combination the server MUST return data (including the phase/subphase attribute(s)) of the currently active phase/subphase.

If the client <fee:command> contains no phase/subphase attributes and the server has more than one active phase/subphase combination the server MUST respond with a 2003 "Required parameter missing" error.

If the client <fee:command> contains no phase/subphase attributes and the server is currently in a "quiet period" (e.g. not accepting registrations or applications) the server MUST return data consistent with the default general availability phase (e.g. "open" or "claims") including the appropriate phase/subphase attribute(s).

If the client <fee:command> contains a phase attribute with no subphase and the server has only one active subphase (or no subphase) of this phase, the server MUST return data (including the phase/subphase attribute(s)) of the provided phase and currently active subphase.

If the client <fee:command> contains a phase attribute with no subphase and the server has more than one active subphase combination of this phase, the server MUST respond with a 2003 "Required parameter missing" error.

If the client <fee:command> contains a subphase with no phase attribute the server MUST respond with a 2003 "Required parameter missing" error.

If the client <fee:command> contains a phase attribute not defined in [RFC8334] or not supported by server the server MUST respond with a 2004 "Parameter value range" error.

If the client <fee:command> contains a subphase attribute (or phase/subphase combination) not supported by server the server MUST respond with a 2004 "Parameter value range" error.

### 3.9. Reason

The <fee:reason> element is used to provide server specific text in an effort to better explain why a <check> command did not complete as the client expected. An OPTIONAL "lang" attribute MAY be present to identify the language, per the language structure in [RFC5646], of the returned text and has a default value of "en" (English).

The <fee:reason> element can be used within the server response <fee:command> element or within the <fee:cd> element. See section 5.1.1 for details on the <fee:cd> "check data" element.

If the server cannot calculate the relevant fees, because the object, command, currency, period, class or some combination is invalid per server policy, the server has two ways of handling error processing of <fee:command> element(s):

1. Fast-fail - The server, upon error identification, MAY stop processing <fee:command> elements and return to the client a <fee:cd> containing the <fee:objID> and a <fee:reason> element detailing the reason for failure.

```
S: <fee:cd avail="0">
S:   <fee:objID>example.xyz</fee:objID>
S:   <fee:reason>Only 1 year registration periods are
S:   valid.</fee:reason>
S: </fee:cd>
```

2. Partial-fail - The server, upon error identification, MAY continue processing <fee:command> elements and return to the client a <fee:cd> containing successfully processed <fee:command>

elements and failed <fee:command> elements. All returned failed <fee:command> elements MUST have a <fee:reason> element detailing the reason for failure, and the server MAY additionally include a <fee:reason> element at the <fee:cd> level.

```
S: <fee:cd avail="0">
S:   <fee:objID>example.xyz</fee:objID>
S:   <fee:command name="create">
S:     <fee:period unit="y">2</fee:period>
S:     <fee:reason>Only 1 year registration periods are
S:     valid.</fee:reason>
S:   </fee:command>
S: </fee:cd>
```

In either failure scenario the server MUST set the <fee:cd> avail attribute to false (0) and the server MUST process all objects in the client request.

## 4. Server Handling of Fee Information

Depending on server policy, a client MAY be required to include the extension elements described in this document for certain transform commands. Servers must provide clear documentation to clients about the circumstances in which this extension must be used.

The server MUST return avail="0" in its response to a <check> command for any object in the <check> command that does not include the <fee:check> extension for which the server would likewise fail a domain <create> command when no <fee> extension is provided for that same object.

If a server receives a <check> command from a client, which results in no possible fee combination, the server MUST set the "avail" attribute of the <fee:cd> element to false (0) and provide a <fee:reason>.

If a server receives a <check> command from a client, which results in an ambiguous result (i.e. multiple possible fee combinations) the server MUST reject the command with a 2003 "Required parameter missing" error.

If a server receives a command from a client, which does not include the fee extension data elements required by the server for that command, then the server MUST respond with a 2003 "Required parameter missing" error.

If the total fee provided by the client is less than the server's own calculation of the fee or the server determines the currency is inappropriate for that command, then the server MUST reject the command with a 2004 "Parameter value range" error.

## 5. EPP Command Mapping

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

### 5.1. EPP Query Commands

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

#### 5.1.1. EPP <check> Command

This extension defines a new command called the Fee Check Command that defines additional elements for the EPP <check> command to provide fee information along with the availability information of the EPP <check> command.

The command MAY contain an <extension> element which MAY contain a <fee:check> element. The <fee:check> element MAY contain one <fee:currency> element and MUST contain one or more <fee:command> elements.

The <fee:command> element(s) MUST contain(s) a "name" attribute (see Section 3.1), an OPTIONAL "phase" attribute, and an OPTIONAL "subphase" attribute (see Section 3.8). The <fee:command> element(s) MAY have the following child elements:

- An OPTIONAL <fee:period> element (as described in Section 3.3).

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:       <domain:check
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:name>example.net</domain:name>
C:         <domain:name>example.xyz</domain:name>
C:       </domain:check>
C:     </check>
C:     <extension>
C:       <fee:check xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:command name="create">
C:           <fee:period unit="y">2</fee:period>
C:         </fee:command>
C:         <fee:command name="renew"/>
C:         <fee:command name="transfer"/>
C:         <fee:command name="restore"/>
C:       </fee:check>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

When the server receives a <check> command that includes the extension elements described above, its response MUST contain an <extension> element, which MUST contain a child <fee:chkData> element. The <fee:chkData> element MUST contain a <fee:currency> element and a <fee:cd> element for each object referenced in the client <check> command.

Each <fee:cd> (check data) element MUST contain the following child elements:

- A <fee:objID> element, which MUST match an element referenced in the client <check> command.
- An OPTIONAL <fee:class> element (as described in Section 3.7).
- A <fee:command> element matching each <fee:command> (unless the "avail" attribute of the <fee:cd> if false) that appeared in the corresponding <fee:check> of the client command. This element MAY have the OPTIONAL "standard" attribute, with a default value of "0" (or "false"), which indicates whether the fee matches the fee of the "standard" classification (see section 3.7). This element MAY have the OPTIONAL "phase" and "subphase" attributes, which

will match the same attributes in the corresponding <fee:command> element of the client command if sent by the client.

The <fee:cd> element also has an OPTIONAL "avail" attribute which is a boolean. If the value of this attribute evaluates to false, this indicates that the server cannot calculate the relevant fees, because the object, command, currency, period, class or some combination is invalid per server policy. If "avail" is false then the <fee:cd> or the <fee:command> element MUST contain a <fee:reason> element (as described in Section 3.9) and the server MAY eliminate some or all of the <fee:command> element(s).

The <fee:command> element(s) MAY have the following child elements:

- An OPTIONAL <fee:period> element (as described in Section 3.3), which contains the same unit, if present, that appeared in the <fee:period> element of the command. If the value of the parent <fee:command> element is "restore", this element MUST NOT be included, otherwise it MUST be included. If no <fee:period> appeared in the client command (and the command is not "restore") then the server MUST return its default period value.
- Zero or more <fee:fee> elements (as described in Section 3.4).
- Zero or more <fee:credit> elements (as described in Section 3.4).
- An OPTIONAL <fee:reason> element (as described in Section 3.9).

If the "avail" attribute of the <fee:cd> element is true (1) and if no <fee:fee> elements are present in a <fee:command> element, this indicates that no fee will be assessed by the server for this command.

If the "avail" attribute of the <fee:cd> element is true (1), then the <fee:command> element MUST NOT contain a <fee:reason> element.

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:       <domain:chkData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:cd>
S:           <domain:name avail="1">example.com</domain:name>
S:         </domain:cd>
S:         <domain:cd>
```

```
S:           <domain:name avail="1">example.net</domain:name>
S:         </domain:cd>
S:         <domain:cd>
S:           <domain:name avail="1">example.xyz</domain:name>
S:         </domain:cd>
S:       </domain:chkData>
S:     </resData>
S:     <extension>
S:       <fee:chkData
S:           xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:cd avail="1">
S:           <fee:objID>example.com</fee:objID>
S:           <fee:class>Premium</fee:class>
S:           <fee:command name="create">
S:             <fee:period unit="y">2</fee:period>
S:             <fee:fee
S:               description="Registration Fee"
S:               refundable="1"
S:               grace-period="P5D">10.00</fee:fee>
S:           </fee:command>
S:           <fee:command name="renew">
S:             <fee:period unit="y">1</fee:period>
S:             <fee:fee
S:               description="Renewal Fee"
S:               refundable="1"
S:               grace-period="P5D">10.00</fee:fee>
S:           </fee:command>
S:           <fee:command name="transfer">
S:             <fee:period unit="y">1</fee:period>
S:             <fee:fee
S:               description="Transfer Fee"
S:               refundable="1"
S:               grace-period="P5D">10.00</fee:fee>
S:           </fee:command>
S:           <fee:command name="restore">
S:             <fee:fee
S:               description="Redemption Fee">15.00</fee:fee>
S:           </fee:command>
S:         </fee:cd>
S:         <fee:cd avail="1">
S:           <fee:objID>example.net</fee:objID>
S:           <fee:class>standard</fee:class>
S:           <fee:command name="create" standard="1">
S:             <fee:period unit="y">2</fee:period>
S:             <fee:fee
S:               description="Registration Fee"
S:               refundable="1"
```

```
S:               grace-period="P5D">5.00</fee:fee>
S:           </fee:command>
S:           <fee:command name="renew" standard="1">
S:             <fee:period unit="y">1</fee:period>
S:             <fee:fee
S:               description="Renewal Fee"
S:               refundable="1"
S:               grace-period="P5D">5.00</fee:fee>
S:           </fee:command>
S:           <fee:command name="transfer" standard="1">
S:             <fee:period unit="y">1</fee:period>
S:             <fee:fee
S:               description="Transfer Fee"
S:               refundable="1"
S:               grace-period="P5D">5.00</fee:fee>
S:           </fee:command>
S:           <fee:command name="restore" standard="1">
S:             <fee:fee
S:               description="Redemption Fee">5.00</fee:fee>
S:           </fee:command>
S:         </fee:cd>
S:         <fee:cd avail="0">
S:           <fee:objID>example.xyz</fee:objID>
S:           <fee:command name="create">
S:             <fee:period unit="y">2</fee:period>
S:             <fee:reason>Only 1 year registration periods are
S:               valid.</fee:reason>
S:           </fee:command>
S:         </fee:cd>
S:       </fee:chkData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

#### 5.1.2. EPP Transfer Query Command

This extension does not add any elements to the EPP <transfer> query command, but does include elements in the response, when the extension is included in the <login> command service extensions.

When the <transfer> query command has been processed successfully, if the client has included the extension in the <login> command service <svcExtension> element, and if the client is authorized by the server to view information about the transfer, then the server MAY include

in the <extension> section of the EPP response a <fee:trnData> element, which contains the following child elements:

- A <fee:currency> element (as described in Section 3.2).
- A <fee:period> element (as described in Section 3.3).
- Zero or more <fee:fee> elements (as described in Section 3.4) containing the fees that will be charged to the gaining client.
- Zero or more <fee:credit> elements (as described in Section 3.4) containing the credits that will be refunded to the losing client.

Servers SHOULD omit <fee:credit> when returning a response to the gaining client, and omit <fee:fee> elements when returning a response to the losing client.

If no <fee:trnData> element is included in the response, then no fee will be assessed by the server for the transfer.

Example <transfer> query 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="1001">
S:       <msg>Command completed successfully; action pending</msg>
S:     </result>
S:     <resData>
S:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2019-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2019-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2021-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:period unit="y">1</fee:period>
S:         <fee:fee>5.00</fee:fee>
S:       </fee:trnData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

## 5.2. EPP Transform Commands

### 5.2.1. EPP <create> Command

This extension adds elements to both the EPP <create> command and response, when the extension is included in the <login> command service extensions.

When submitting a <create> command to the server, the client MAY include in the <extension> element a <fee:create> element which includes the following child elements:

- An OPTIONAL <fee:currency> element (as described in Section 3.2);
- One or more <fee:fee> elements (as described in Section 3.4).

When the <create> command has been processed successfully, and the client included the extension in the <login> command service extensions, and a fee was assessed by the server for the transaction, the server MUST include in the <extension> section of the EPP response a <fee:creData> element, which contains the following child elements:

- A <fee:currency> element (as described in Section 3.2);
- Zero or more <fee:fee> elements (as described in Section 3.4);
- Zero or more <fee:credit> elements (as described in Section 3.4);
- An OPTIONAL <fee:balance> element (as described in Section 3.5);
- An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

Example <create> command:

```
C: <?xml version="1.0" encoding="utf-8" standalone="no"?>
C: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:   <command>
C:     <create>
C:       <domain:create
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:period unit="y">2</domain:period>
C:         <domain:ns>
C:           <domain:hostObj>ns1.example.net</domain:hostObj>
C:           <domain:hostObj>ns2.example.net</domain:hostObj>
C:         </domain:ns>
C:         <domain:registrant>jd1234</domain:registrant>
C:         <domain:contact type="admin">sh8013</domain:contact>
C:         <domain:contact type="tech">sh8013</domain:contact>
C:         <domain:authInfo>
C:           <domain:pw>2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:create>
C:     </create>
C:     <extension>
C:       <fee:create xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:create>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

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:       <domain:creData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:crDate>2019-04-03T22:00:00.0Z</domain:crDate>
S:         <domain:exDate>2021-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:creData>
S:     </resData>
S:     <extension>
S:       <fee:creData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           description="Registration Fee"
S:           lang="en"
S:           refundable="1"
S:           grace-period="P5D">5.00</fee:fee>
S:         <fee:balance>-5.00</fee:balance>
S:         <fee:creditLimit>1000.00</fee:creditLimit>
S:       </fee:creData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

#### 5.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command, but does include elements in the response, when the extension is included in the <login> command service extensions.

When the <delete> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the EPP response a <fee:delData> element, which contains the following child elements:

- A <fee:currency> element (as described in Section 3.2);

- Zero or more <fee:fee> elements (as described in Section 3.4);
- Zero or more <fee:credit> elements (as described in Section 3.4);
- An OPTIONAL <fee:balance> element (as described in Section 3.5);
- An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

Example <delete> response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <extension>
S:       <fee:delData
S:         xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:credit
S:           description="AGP Credit"
S:           lang="en">-5.00</fee:credit>
S:         <fee:balance>1005.00</fee:balance>
S:       </fee:delData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

#### 5.2.3. EPP <renew> Command

This extension adds elements to both the EPP <renew> command and response, when the extension is included in the <login> command service extensions.

When submitting a <renew> command to the server, the client MAY include in the <extension> element a <fee:renew> element which includes the following child elements:

- An OPTIONAL <fee:currency> element (as described in Section 3.2);
- One or more <fee:fee> elements (as described in Section 3.4).

When the <renew> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the

EPP response a <fee:renData> element, which contains the following child elements:

- A <fee:currency> element (as described in Section 3.2);
- Zero or more <fee:fee> elements (as described in Section 3.4);
- Zero or more <fee:credit> elements (as described in Section 3.4);
- An OPTIONAL <fee:balance> element (as described in Section 3.5);
- An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

Example <renew> 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:     <renew>
C:       <domain:renew
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:curExpDate>2019-04-03</domain:curExpDate>
C:         <domain:period unit="y">5</domain:period>
C:       </domain:renew>
C:     </renew>
C:     <extension>
C:       <fee:renew xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:renew>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

Example <renew> 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:       <domain:renData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:exDate>2024-04-03T22:00:00.0Z</domain:exDate>
S:       </domain:renData>
S:     </resData>
S:     <extension>
S:       <fee:renData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           refundable="1"
S:           grace-period="P5D">5.00</fee:fee>
S:         <fee:balance>1000.00</fee:balance>
S:       </fee:renData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

#### 5.2.4. EPP <transfer> Command

This extension adds elements to both the EPP <transfer> command and response, when the value of the "op" attribute of the <transfer> command element is "request", and the extension is included in the <login> command service extensions.

When submitting a <transfer> command to the server, the client MAY include in the <extension> element a <fee:transfer> element which includes the following child elements:

- An OPTIONAL <fee:currency> element (as described in Section 3.2);
- One or more <fee:fee> elements (as described in Section 3.4).

When the <transfer> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the

EPP response a <fee:trnData> element, which contains the following child elements:

- A <fee:currency> element (as described in Section 3.2);
- Zero or more <fee:fee> elements (as described in Section 3.4);
- Zero or more <fee:credit> elements (as described in Section 3.4);
- An OPTIONAL <fee:balance> element (as described in Section 3.5);
- An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

Example <transfer> 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:     <transfer op="request">
C:       <domain:transfer
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:period unit="y">1</domain:period>
C:         <domain:authInfo>
C:           <domain:pw roid="JD1234-REP">2fooBAR</domain:pw>
C:         </domain:authInfo>
C:       </domain:transfer>
C:     </transfer>
C:     <extension>
C:       <fee:transfer xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:transfer>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

Example <transfer> 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="1001">
S:       <msg>Command completed successfully; action pending</msg>
S:     </result>
S:     <resData>
S:       <domain:trnData
S:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:         <domain:name>example.com</domain:name>
S:         <domain:trStatus>pending</domain:trStatus>
S:         <domain:reID>ClientX</domain:reID>
S:         <domain:reDate>2019-06-08T22:00:00.0Z</domain:reDate>
S:         <domain:acID>ClientY</domain:acID>
S:         <domain:acDate>2019-06-13T22:00:00.0Z</domain:acDate>
S:         <domain:exDate>2021-09-08T22:00:00.0Z</domain:exDate>
S:       </domain:trnData>
S:     </resData>
S:     <extension>
S:       <fee:trnData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee
S:           refundable="1"
S:           grace-period="P5D">5.00</fee:fee>
S:       </fee:trnData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54322-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

#### 5.2.5. EPP <update> Command

This extension adds elements to both the EPP <update> command and response, when the extension is included in the <login> command service extensions.

When submitting a <update> command to the server, the client MAY include in the <extension> element a <fee:update> element which includes the following child elements:

- An OPTIONAL <fee:currency> element (as described in Section 3.2);
- One or more <fee:fee> elements (as described in Section 3.4).

When the <update> command has been processed successfully, and the client included the extension in the <login> command service extensions, the server MAY include in the <extension> section of the EPP response a <fee:updData> element, which contains the following child elements:

- A <fee:currency> element (as described in Section 3.2);
- Zero or more <fee:fee> elements (as described in Section 3.4);
- Zero or more <fee:credit> elements (as described in Section 3.4);
- An OPTIONAL <fee:balance> element (as described in Section 3.5);
- An OPTIONAL <fee:creditLimit> element (as described in Section 3.6).

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:       <domain:update
C:         xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:         <domain:name>example.com</domain:name>
C:         <domain:chg>
C:           <domain:registrant>sh8013</domain:registrant>
C:         </domain:chg>
C:       </domain:update>
C:     </update>
C:     <extension>
C:       <fee:update xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
C:         <fee:currency>USD</fee:currency>
C:         <fee:fee>5.00</fee:fee>
C:       </fee:update>
C:     </extension>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C: </epp>
```

Example <update> response:

```
S: <?xml version="1.0" encoding="utf-8" standalone="no"?>
S: <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <extension>
S:       <fee:updData xmlns:fee="urn:ietf:params:xml:ns:epp:fee-1.0">
S:         <fee:currency>USD</fee:currency>
S:         <fee:fee>5.00</fee:fee>
S:       </fee:updData>
S:     </extension>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S: </epp>
```

## 6. Formal Syntax

One schema is presented here that is the EPP Fee Extension 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.

### 6.1. Fee Extension 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.

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

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

    <annotation>
      <documentation>
        Extensible Provisioning Protocol v1.0 Fee Extension
      </documentation>
    </annotation>

    <!-- Child elements found in EPP commands and responses -->
    <element name="check" type="fee:checkType" />
    <element name="chkData" type="fee:chkDataType" />
    <element name="create" type="fee:transformCommandType" />
    <element name="creData" type="fee:transformResultType" />
    <element name="renew" type="fee:transformCommandType" />
    <element name="renData" type="fee:transformResultType" />
    <element name="transfer" type="fee:transformCommandType" />
    <element name="trnData" type="fee:transformResultType" />
    <element name="update" type="fee:transformCommandType" />
    <element name="updData" type="fee:transformResultType" />
    <element name="delData" type="fee:transformResultType" />

    <!-- client <check> command -->
    <complexType name="checkType">
      <sequence>
        <element name="currency" type="fee:currencyType"
          minOccurs="0" />
        <element name="command" type="fee:commandType"
          minOccurs="1" maxOccurs="unbounded" />
      </sequence>
    </complexType>

    <complexType name="objectIdentifierType">
      <simpleContent>
        <extension base="eppcom:labelType">
          <attribute name="element"
          type="NMTOKEN" default="name" />
        </extension>
      </simpleContent>
    </complexType>

    <!-- server <check> result -->
    <complexType name="chkDataType">
      <sequence>
        <element name="currency" type="fee:currencyType" />
        <element name="cd" type="fee:objectCDType"
          maxOccurs="unbounded" />
      </sequence>
```

```
    </complexType>

    <complexType name="objectCDType">
      <sequence>
        <element name="objID" type="fee:objectIdentifierType" />
        <element name="class" type="token" minOccurs="0" />
        <element name="command" type="fee:commandDataType"
          minOccurs="0" maxOccurs="unbounded" />
        <element name="reason" type="fee:reasonType" minOccurs="0" />
      </sequence>
      <attribute name="avail" type="boolean" default="1" />
    </complexType>

    <!-- general transform (create, renew, update, transfer) command -->
    <complexType name="transformCommandType">
      <sequence>
        <element name="currency" type="fee:currencyType"
          minOccurs="0" />
        <element name="fee" type="fee:feeType"
          maxOccurs="unbounded" />
        <element name="credit" type="fee:creditType"
          minOccurs="0" maxOccurs="unbounded" />
      </sequence>
    </complexType>

    <!-- general transform (create, renew, update) result -->
    <complexType name="transformResultType">
      <sequence>
        <element name="currency" type="fee:currencyType"
          minOccurs="0" />
        <element name="period" type="domain:periodType"
          minOccurs="0" />
        <element name="fee" type="fee:feeType"
          minOccurs="0" maxOccurs="unbounded" />
        <element name="credit" type="fee:creditType"
          minOccurs="0" maxOccurs="unbounded" />
        <element name="balance" type="fee:balanceType"
          minOccurs="0" />
        <element name="creditLimit" type="fee:creditLimitType"
          minOccurs="0" />
      </sequence>
    </complexType>

    <!-- common types -->
    <simpleType name="currencyType">
      <restriction base="string">
        <pattern value="[A-Z]{3}" />
      </restriction>
```

```
    </simpleType>

    <complexType name="commandType">
      <sequence>
        <element name="period" type="domain:periodType"
          minOccurs="0" maxOccurs="1" />
      </sequence>
      <attribute name="name" type="fee:commandEnum" use="required"/>
      <attribute name="customName" type="token"/>
      <attribute name="phase" type="token" />
      <attribute name="subphase" type="token" />
    </complexType>

    <complexType name="commandDataType">
      <complexContent>
        <extension base="fee:commandType">
          <sequence>
            <element name="fee" type="fee:feeType"
              minOccurs="0" maxOccurs="unbounded" />
            <element name="credit" type="fee:creditType"
              minOccurs="0" maxOccurs="unbounded" />
            <element name="reason" type="fee:reasonType"
              minOccurs="0" />
          </sequence>
          <attribute name="standard" type="boolean" default="0" />
        </extension>
      </complexContent>
    </complexType>

    <complexType name="reasonType">
      <simpleContent>
        <extension base="token">
          <attribute name="lang" type="language" default="en"/>
        </extension>
      </simpleContent>
    </complexType>

    <simpleType name="commandEnum">
      <restriction base="token">
        <enumeration value="create"/>
        <enumeration value="delete"/>
        <enumeration value="renew"/>
        <enumeration value="update"/>
        <enumeration value="transfer"/>
        <enumeration value="restore"/>
        <enumeration value="custom"/>
      </restriction>
    </simpleType>
```

```
    <simpleType name="nonNegativeDecimal">
      <restriction base="decimal">
        <minInclusive value="0" />
      </restriction>
    </simpleType>

    <simpleType name="negativeDecimal">
      <restriction base="decimal">
        <maxInclusive value="0" />
      </restriction>
    </simpleType>

    <complexType name="feeType">
      <simpleContent>
        <extension base="fee:nonNegativeDecimal">
          <attribute name="description"/>
          <attribute name="lang" type="language" default="en"/>
          <attribute name="refundable" type="boolean" />
          <attribute name="grace-period" type="duration" />
          <attribute name="applied">
            <simpleType>
              <restriction base="token">
                <enumeration value="immediate" />
                <enumeration value="delayed" />
              </restriction>
            </simpleType>
          </attribute>
        </extension>
      </simpleContent>
    </complexType>

    <complexType name="creditType">
      <simpleContent>
        <extension base="fee:negativeDecimal">
          <attribute name="description"/>
          <attribute name="lang" type="language" default="en"/>
        </extension>
      </simpleContent>
    </complexType>

    <simpleType name="balanceType">
      <restriction base="decimal" />
    </simpleType>

    <simpleType name="creditLimitType">
      <restriction base="decimal" />
    </simpleType>
```

```
</schema>
END
```

## 7. Security Considerations

The mapping extensions described in this document do not provide any security services beyond those described by EPP [RFC5730], the EPP domain name mapping [RFC5731], and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well. This extension passes financial information using the EPP protocol, so confidentiality and integrity protection must be provided by the transport mechanism. All transports compliant with [RFC5730] provide the needed level of confidentiality and integrity protections. The server will only provide information, including financial information, that is relevant to the authenticated client.

## 8. IANA Considerations

### 8.1. 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 fee namespace:

URI: urn:ietf:params:xml:ns:epp:fee-1.0

Registrant Contact: IESG

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

Registration request for the fee schema:

URI: urn:ietf:params:xml:schema:epp:fee-1.0

Registrant Contact: IESG

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

### 8.2. 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 Fee Extension for the Extensible Provisioning Protocol (EPP)

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

## 9. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

### 9.1. RegistryEngine EPP Service

Organization: CentralNic

Name: RegistryEngine EPP Service

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

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

Coverage: All aspects of the protocol are implemented.

Licensing: Proprietary In-House software

Contact: epp@centralnic.com

URL: https://www.centralnic.com

## 10. Acknowledgements

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

- James Gould of Verisign Inc
- Luis Munoz of ISC
- Michael Young of Architelos
- Ben Levac and Jeff Eckhaus of Demand Media
- Seth Goldman of Google
- Klaus Malorny and Michael Bauland of Knipp
- Jody Kolker, Joe Snitker and Kevin Allendorf of Go Daddy
- Michael Holloway of Com Laude
- Santosh Kalsangrah of Impetus Infotech
- Alex Mayrhofer of Nic.at
- Thomas Corte of Knipp Medien und Kommunikation GmbH

## 11. Change History

### 11.1. Change from 18 to 19

Added normative reference for XML Schema.

### 11.2. Change from 18 to 19

Updated per IESG review, all updates (except for one schema change) were just textual for clarity and correctness. The schema change was to require the name attribute of the commandType element.

### 11.3. Change from 17 to 18

Corrected erroneous edit left in place in previous revision (17), reverted text back to original text (revision 16) in section 3.4.

### 11.4. Change from 16 to 17

Updated per AD review, all updates were just textual for clarity and correctness.

### 11.5. Change from 15 to 16

Updated per AD review and list comments: several grammar corrections; clarification text added to section 3.4.3 and 3.5; and a schema update for consistency by providing a "lang" attribute to the <fee:fee> and <fee:credit> "description" attribute detailed in section 3.4.

### 11.6. Change from 14 to 15

Updated schema, moving the "standard" attribute of the "commandDataType" inside the <extension> block.

### 11.7. Change from 13 to 14

Moved RFC 7451 reference from Normative to Informative section.

### 11.8. Change from 12 to 13

Updated XML namespace and schema registration to be "epp" scoped - global replace of XML namespace from urn:ietf:params:xml:ns:fee-1.0 to urn:ietf:params:xml:ns:epp:fee-1.0 and the XML schema registration from urn:ietf:params:xml:schema:fee-1.0 to urn:ietf:params:xml:schema:epp:fee-1.0.

### 11.9. Change from 11 to 12

Updated references to current version of documents and moved the "standard" attribute from the check command (commandType) to the check response (commandDataType).

### 11.10. Change from 10 to 11

Updated document per Working Group Last Call comments. Made minor textual changes throughout for enhanced clarity per WGLC comments.

### 11.11. Change from 09 to 10

Updated document per Working Group Last Call comments. Updated schema to version 1.0 in anticipation of standardization, no changes were made to the latest, 0.25, schema. Made minor textual changes throughout for enhanced clarity per WGLC comments.

### 11.12. Change from 08 to 09

Updated scheme to version 0.25 to allow tighter checking on <fee:command> by splitting the client and server definitions, moved the class element from the command to the object level and added an optional standard attribute to the command element. Also updated section 3.1 for clarity on name attribute; updated section 3.9 for clarity on uses of <fee:reason>; removed second paragraph in section 5.2.1 as it was duplicative of second to last paragraph in 4.0; and updated section 5.1.1 to add section references.

### 11.13. Change from 07 to 08

Updated section 3.8 and 5.1.1 to provide clarity on server processing and response of various scenarios (i.e. "quiet" period processing).

### 11.14. Change from 06 to 07

Updated section 3.8 and 4.0 to provide clarity on server processing and response of various scenarios.

### 11.15. Change from 05 to 06

Updated scheme to version 0.23 to allow the return of no <fee:command> element(s) if an error situation occurs. Edited section 3.8 extensively after input from interim meeting and REGEXT F2F meeting at IETF-99. Added normative reference for draft-ietf-eppext-launchphase.

### 11.16. Change from 04 to 05

Updated scheme to version 0.21 to support the lang attribute for the reason element of the objectCDType and the commandType types as well as to add the update command to the commandEnum type. Updated section 3.1 to include language for the custom command. Added section 3.9 to provide a description of the <fee:reason> element. Fixed typos and added clarification text on when client fee is less than server fee in section 4. Additionally, I added description pointers to appropriate Section 3 definitions for element clarity throughout the document.

### 11.17. Change from 03 to 04

Updated scheme to version 0.19 to correct typos and to replace the commandTypeValue type with the commandEnum type and customName attribute for stricter validation. Updated various text for grammar and clarity. Added text to section 4 clarifying the <check> response

when the client provided no fee extension but the server was expecting the extension.

### 11.18. Change from 02 to 03

Updated scheme to version 0.17 to simplify the check command syntax. Moved fee avail to objectCDType to allow fast failing on error situations. Removed the objectCheckType as it was no longer being used. Updated examples to reflect these scheme changes. Added language for server failing a <create> if the <fee:fee> passed by the client is less than the server fee.

### 11.19. Change from 01 to 02

Updated scheme to version 0.15 to fix errors in CommandType, objectCDType, transformCommandType and transformResultType definitions.

### 11.20. Change from 00 to 01

Added Roger Carney as author to finish draft. Moved Formal Syntax section to main level numbering. Various grammar, typos, and administrative edits for clarity. Removed default value for the "applied" attribute of <fee:fee> so that it can truly be optional. Added support for the <delete> command to return a <fee:fee> element as well. Modified default response on the <check> command for the optional <fee:period> when it was not provided in the command, leaving it to the server to provide the default period value. Extensive edits were done to the <check> command, the <check> response and to the fee extension schema (checkType, objectCheckType, objectIdentifierType, objectCDType, commandType) to support requesting and returning multiple transformation fees in a single call. Added section on Phase/Subphase to provide more context on the uses.

### 11.21. Change from draft-brown-00 to draft-ietf-regext-fees-00

Updated to be REGEXT WG document.

## 12. References

### 12.1. Normative References

[ISO4217:2015]
International Organization for Standardization, "Codes for the representation of currencies", August 2015, <https://www.iso.org/standard/64758.html>.

[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>.

[RFC3915] Hollenbeck, S., "Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP)", RFC 3915, DOI 10.17487/RFC3915, September 2004, <https://www.rfc-editor.org/info/rfc3915>.

[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>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8334] Gould, J., Tan, W., and G. Brown, "Launch Phase Mapping for the Extensible Provisioning Protocol (EPP)", RFC 8334, DOI 10.17487/RFC8334, March 2018, <https://www.rfc-editor.org/info/rfc8334>.

[W3C.REC-xmlschema-1-20041028] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

### 12.2. Informative References

[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>.

## Authors' Addresses

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

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

Gavin Brown
CentralNic Group plc
35-39 Moorgate
London, England EC2R 6AR
GB

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

Jothan Frakes

Email: jothan@jothan.com
URI: http://jothan.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: June 3, 2019

L. Zhou
CNNIC
N. Kong
Consultant
G. Zhou
J. Yao
CNNIC
J. Gould
Verisign, Inc.

November 30, 2018

# Extensible Provisioning Protocol (EPP) Organization Mapping draft-ietf-regext-org-12

## Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for provisioning and management of organization objects stored in a shared central repository. Specified in Extensible Markup Language (XML), this extended mapping is applied to provide additional features required for the provisioning of organizations.

## 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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on June 3, 2019.

Table of Contents

## 1. Introduction

There are many entities, such as registrars, resellers, DNS service operators, or privacy proxies involved in the domain registration business. These kind of entities have not been formally defined as having an object in Extensible Provisioning Protocol (EPP). This document provides a way to specify them as "organization" entities.

This document describes an organization object mapping for version 1.0 of the EPP [RFC5730]. This mapping is specified using the XML 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

## 2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

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

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

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

## 3. Object Attributes

An EPP organization object has attributes and associated values that can 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.

### 3.1. Organization Identifier

All EPP organizations are identified by a server-unique identifier. Organization identifiers are character strings with a specified minimum length, a specified maximum length, and a specified format. Organization identifiers use the "clIDType" client identifier syntax described in [RFC5730]. Its corresponding element is <org:id>.

### 3.2. Organization Roles

The organization roles are used to represent the relationship an organization could have. Its corresponding element is <org:role>. An organization object MUST always have at least one associated role. Roles can be set only by the client that sponsors an organization object. A client can change the role of an organization object using the EPP <update> command.

#### 3.2.1. Role Type

An organization role MUST have a type field. This may have any of the values listed in Section 7.3. An organization could have multiple roles with different role types. Its corresponding element is <org:type>.

#### 3.2.2. Role Status

A role of an organization object MAY have its own statuses. Its corresponding element is <org:status>. The values of the role status are defined in Section 3.5.

#### 3.2.3. Role Identifier

A role MAY have a third-party-assigned identifier such as the IANA ID for registrars. Its corresponding element is <org:roleID>.

Example of organization role identifier:

```
<org:role>
  <org:type>registrar</org:type>
  <org:status>ok</org:status>
  <org:status>linked</org:status>
  <org:roleID>1362</org:roleID>
</org:role>
```

### 3.3. Contact and Client Identifiers

All EPP contacts are identified by server-unique identifiers. Contact identifiers are character strings with a specified minimum length, a specified maximum length, and a specified format. Contact identifiers use the "clIDType" client identifier syntax described in [RFC5730].

### 3.4. Organization Status Values

An organization object MUST always have at least one associated status value. Status values can be set only by the client that sponsors an organization object and by the server on which the object resides. A client can change the status of an organization object using the EPP <update> command. Each status value MAY be accompanied by a string of human-readable text that describes the rationale for the status applied to the object.

A client MUST NOT alter server status values set by the server. A server MAY alter or override status values set by a client, subject to local server policies. The status of an object MAY change as a result of either a client-initiated transform command or an action performed by a server operator.

Status values that can be added or removed by a client are prefixed with "client". Corresponding server status values that can be added or removed by a server are prefixed with "server". The "hold" and "terminated" status values are server-managed when the organization has no parent identifier [Section 3.6] and otherwise MAY be client-managed based on server policy. Other status values that do not begin with either "client" or "server" are server-managed.

Status Value Descriptions:

- ok: This is the normal status value for an object that has no operations pending or active prohibitions. This value is set and removed by the server as other status values are added or removed.

- hold: Organization transform commands and new links MUST be rejected.

- terminated: The organization which has been terminated MUST NOT be linked. Organization transform commands and new links MUST be rejected.

- linked: The organization object has at least one active association with another object. The "linked" status is not

explicitly set by the client. Servers should provide services to determine existing object associations.

- o clientLinkProhibited, serverLinkProhibited: Requests to add new links to the organization MUST be rejected.
- o clientUpdateProhibited, serverUpdateProhibited: Requests to update the object (other than to remove this status) MUST be rejected.
- o clientDeleteProhibited, serverDeleteProhibited: Requests to delete the object MUST be rejected.
- o pendingCreate, pendingUpdate, pendingDelete: A transform command has been processed for the object, but the action has not been completed by the server. Server operators can delay action completion for a variety of reasons, such as to allow for human review or third-party action. A transform command that is processed, but whose requested action is pending, is noted with response code 1001.

"pendingCreate", "ok", "hold", and "terminated" are mutually exclusive statuses. Organization MUST have exactly one of these statuses set.

"ok" status MAY only be combined with "linked" status.

A client or server MAY combine "linked" with either "clientLinkProhibited" or "serverLinkProhibited" if new links must be prohibited.

"pendingDelete" status MUST NOT be combined with either "clientDeleteProhibited" or "serverDeleteProhibited" status.

The pendingCreate, pendingDelete, and pendingUpdate status values MUST NOT be combined with each other.

If "clientUpdateProhibited" or "serverUpdateProhibited" is set, the client will not be able to update the object. For "clientUpdateProhibited", the client will first need to remove "clientUpdateProhibited" prior to attempting to update the object. The server can modify the object at any time.

### 3.5. Role Status Values

A role SHOULD have at least one associated status value. Valid values include "ok", "linked", "clientLinkProhibited", and "serverLinkProhibited".

Status Value Descriptions:

- ok: This is the normal status value for an role that has no operations pending or active prohibitions. This value is set and removed by the server as other status values are added or removed.

- linked: The role of an organization object has at least one active association with another object. The "linked" status is not explicitly set by the client. Servers SHOULD provide services to determine existing object associations.

- clientLinkProhibited, serverLinkProhibited: Requests to add new links to the role MUST be rejected.

### 3.6. Parent Identifier

There can be more than one layer of organizations, such as a reseller. The parent identifier, as defined with the <org:parentId> element, represents the parent organization identifier in a child organization.

The case of reseller organizations provides an example. The parent identifier is not defined for the top level reseller, namely the registrar of the registry. An N-tier reseller has a parent reseller and at least one child reseller. A reseller customer has a parent reseller and no child resellers.

Loops MUST be prohibited. For example: if organization A has B as its parent identifier, organization B cannot have organization A as its parent identifier. The same is true for larger loops involving three or more organizations.

### 3.7. URL

The URL represents the organization web home page, as defined with the <org:url> element.

### 3.8. 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 [W3C.REC-xmlschema-2-20041028] 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.

# 4. 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 organization information via EPP.

## 4.1. EPP Query Commands

EPP provides two commands to retrieve organization information: <check> to determine if an organization object can be provisioned within a repository, and <info> to retrieve detailed information associated with an organization object. This document does not define a mapping for the EPP <transfer> command to retrieve organization-object transfer status information.

### 4.1.1. EPP <check> Command

The EPP <check> command is used to determine if an object can be provisioned within a repository. It provides a hint that allows a client to anticipate the success or failure of provisioning an object using the <create> command, as object-provisioning requirements are ultimately a matter of server policy.

In addition to the standard EPP command elements, the <check> command MUST contain an <org:check> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:check> element contains the following child elements:

- One or more <org:id> elements that contain the server-unique identifier of the organization 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:      <org:check
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:        <org:id>res1523</org:id>
C:        <org:id>re1523</org:id>
C:        <org:id>1523res</org:id>
C:      </org: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 <org:chkData> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:chkData> element contains one or more <org:cd> elements that contain the following child elements:

- An <org:id> element that identifies the queried object. This element MUST contain an "avail" attribute whose value indicates object availability (can it be provisioned or not) at the moment the <check> command was completed. A value of "1" or "true" means that the object can be provisioned. A value of "0" or "false" means that the object cannot be provisioned.

- An OPTIONAL <org:reason> element that may be provided when an object cannot be provisioned. If present, this element contains server-specific text to help explain why the object cannot be provisioned. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute as defined in [RFC5646] may be present to identify the language if the negotiated value is something other than the default value of "en"(English).

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 lang="en">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:chkData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:cd>
S:          <org:id avail="1">res1523</org:id>
S:        </org:cd>
S:        <org:cd>
S:          <org:id avail="0">re1523</org:id>
S:          <org:reason lang="en">In use</org:reason>
S:        </org:cd>
S:        <org:cd>
S:          <org:id avail="1">1523res</org:id>
S:        </org:cd>
S:      </org: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.

#### 4.1.2. EPP <info> Command

The EPP <info> command is used to retrieve information associated with an organization object. In addition to the standard EPP command elements, the <info> command MUST contain a <org:info> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:info> element contains the following child elements:

- An <org:id> element that contains the server-unique identifier of the organization object to be queried.

Example <info> 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:    <info>
C:      <org:info
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:        <org:id>res1523</org:id>
C:      </org: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 <org:infData> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:infData> element contains the following child elements:

- An <org:id> element that contains the server-unique identifier of the organization object, as defined in Section 3.1.
- An <org:roid> element that contains the Repository Object IDentifier assigned to the organization object when the object was created.
- One or more <org:role> elements that contain the role type, role statuses and optional role id of the organization.
  - An <org:type> element that contains the type of the organization, as defined in Section 3.2.
  - One or more <org:status> elements that contain the role statuses. The values of the role status are defined in Section 3.5.
  - An OPTIONAL <org:roleID> element that contains a third-party-assigned identifier, such as IANA ID for registrars, as defined in Section 3.2.3.
- One or more <org:status> elements that contain the operational status of the organization, as defined in Section 3.4.
- An OPTIONAL <org:parentId> element that contains the identifier of the parent object, as defined in Section 3.6.
- Zero to two <org:postalInfo> elements that contain postal-address information. Two elements are provided so that address

information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of Unicode in the range U+0020 - U+007E. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The <org:postalInfo> element contains the following child elements:

* An <org:name> element that contains the name of the organization.

* An OPTIONAL <org:addr> element that contains address information associated with the organization. A <org:addr> element contains the following child elements:

  + One, two, or three <org:street> elements that contain the organization's street address.

  + An <org:city> element that contains the organization's city.

  + An OPTIONAL <org:sp> element that contains the organization's state or province.

  + An OPTIONAL <org:pc> element that contains the organization's postal code.

  + An <org:cc> element that contains the alpha-2 organization's country code. The detailed format of this element is described in section 2.4.3 of [RFC5733].

- An OPTIONAL <org:voice> element that contains the organization's voice telephone number. The detailed format of this element is described in Section 2.5 of [RFC5733].

- An OPTIONAL <org:fax> element that contains the organization's facsimile telephone number.

- An OPTIONAL <org:email> element that contains the organization's email address. The detailed format of this element is described in section 2.6 of [RFC5733].

- An OPTIONAL <org:url> element that contains the URL to the website of the organization. The detailed format of this element is described in [RFC3986].

- Zero or more <org:contact> elements that contain identifiers for the contact objects to be associated with the organization object.

Contact object identifiers MUST be known to the server before the contact object can be associated with the organization object. The required "type" is used to represent contact types. The type values include "admin", "tech", "billing", "abuse", and "custom". The OPTIONAL "typeName" attribute is used to define the name of a "custom" type.

- An OPTIONAL <org:clID> element that contains the organization identifier of the sponsoring client. There is no <org:clID> element if the organization is managed by the registry.

- An <org:crID> element that contains the identifier of the client that created the organization object.

- An <org:crDate> element that contains the date and time of organization object creation.

- An <org:upID> element that contains the identifier of the client that last updated the organization object. This element MUST NOT be present if the organization has never been modified.

- An <org:upDate> element that contains the date and time of the most recent organization object modification. This element MUST NOT be present if the organization object has never been modified.

Example <info> response for "Example Registrar Inc." organization organization object with identifier "registrar1362":

```
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 lang="en">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:infData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>registrar1362</org:id>
S:        <org:roid>registrar1362-REP</org:roid>
S:        <org:role>
S:          <org:type>registrar</org:type>
S:          <org:status>ok</org:status>
S:          <org:status>linked</org:status>
S:          <org:roleID>1362</org:roleID>
S:        </org:role>
S:        <org:status>ok</org:status>
S:        <org:postalInfo type="int">
```

```
S:          <org:name>Example Registrar Inc.</org:name>
S:          <org:addr>
S:            <org:street>123 Example Dr.</org:street>
S:            <org:street>Suite 100</org:street>
S:            <org:city>Dulles</org:city>
S:            <org:sp>VA</org:sp>
S:            <org:pc>20166-6503</org:pc>
S:            <org:cc>US</org:cc>
S:          </org:addr>
S:        </org:postalInfo>
S:        <org:voice x="1234">+1.7035555555</org:voice>
S:        <org:fax>+1.7035555556</org:fax>
S:        <org:email>contact@organization.example</org:email>
S:        <org:url>https://organization.example</org:url>
S:        <org:contact type="admin">sh8013</org:contact>
S:        <org:contact type="billing">sh8013</org:contact>
S:        <org:contact type="custom"
S:           typeName="legal">sh8013</org:contact>
S:        <org:crID>ClientX</org:crID>
S:        <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:        <org:upID>ClientX</org:upID>
S:        <org:upDate>1999-12-03T09:00:00.0Z</org:upDate>
S:      </org:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> response for "Example Reseller Inc." organization object of reseller type managed by identifier "registrar1362":

```
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 lang="en">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:infData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>reseller1523</org:id>
S:        <org:roid>reseller1523-REP</org:roid>
S:        <org:role>
S:          <org:type>reseller</org:type>
S:          <org:status>ok</org:status>
S:          <org:status>linked</org:status>
S:        </org:role>
S:        <org:status>ok</org:status>
S:        <org:parentId>registrar1362</org:parentId>
S:        <org:postalInfo type="int">
S:          <org:name>Example Reseller Inc.</org:name>
S:          <org:addr>
S:            <org:street>123 Example Dr.</org:street>
S:            <org:street>Suite 100</org:street>
S:            <org:city>Dulles</org:city>
S:            <org:sp>VA</org:sp>
S:            <org:pc>20166-6503</org:pc>
S:            <org:cc>US</org:cc>
S:          </org:addr>
S:        </org:postalInfo>
S:        <org:fax>+1.7035555556</org:fax>
S:        <org:url>https://organization.example</org:url>
S:        <org:contact type="admin">sh8013</org:contact>
S:        <org:clID>1362</org:clID>
S:        <org:crID>ClientX</org:crID>
S:        <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:        <org:upID>ClientX</org:upID>
S:        <org:upDate>1999-12-03T09:00:00.0Z</org:upDate>
S:      </org: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.

#### 4.1.3. EPP <transfer> Query Command

The transfer semantics does not apply to organization object. No EPP <transfer> query command is defined in this document.

### 4.2. EPP Transform Commands

This document provides three commands to transform organization object information: <create> to create an instance of an organization object, <delete> to delete an instance of an organization object, and <update> to change information associated with an organization object. This document does not define a mapping for the EPP <transfer> and <renew> command.

Transform commands are typically processed and completed in real time. Server operators MAY receive and process transform commands but defer completing the requested action if human or third-party review is required before the requested action can be completed. In such situations, the server MUST return a 1001 response code to the client to note that the command has been received and processed but that the requested action is pending. The server MUST also manage the status of the object that is the subject of the command to reflect the initiation and completion of the requested action. Once the action has been completed, the client MUST be notified using a service message that the action has been completed and that the status of the object has changed. Other notification methods MAY be used in addition to the required service message.

#### 4.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create an organization object. In addition to the standard EPP command elements, the <create> command MUST contain a <org:create> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:create> element contains the following child elements:

- An <org:id> element that contains the desired server-unique identifier for the organization to be created, as defined in Section 3.1.
- One or more <org:role> elements that contain the role type, role statuses and optional role id of the organization.
  - An <org:type> element that contains the type of the organization, as defined in Section 3.2.

* Zero or more <org:status> elements that contain the role statuses. The values of the role status are defined in Section 3.5.

* An OPTIONAL <org:roleID> element that contains a third-party-assigned identifier, such as IANA ID for registrars, as defined in Section 3.2.3.

- Zero or more <org:status> elements that contain the operational status of the organization, as defined in Section 3.4.

- An OPTIONAL <org:parentId> element that contains the identifier of the parent object, as defined in Section 3.6.

- Zero to two <org:postalInfo> elements that contain postal-address information. Two elements are provided so that address information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of Unicode in the range U+0020 - U+007E. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The <org:postalInfo> element contains the following child elements:

  * An <org:name> element that contains the name of the organization.

  * An OPTIONAL <org:addr> element that contains address information associated with the organization. A <org:addr> element contains the following child elements:

    + One, two, or three <org:street> elements that contain the organization’s street address.

    + An <org:city> element that contains the organization’s city.

    + An OPTIONAL <org:sp> element that contains the organization’s state or province.

    + An OPTIONAL <org:pc> element that contains the organization’s postal code.

    + An <org:cc> element that contains the alpha-2 organization’s country code. The detailed format of this element is described in section 2.4.3 of [RFC5733].

- An OPTIONAL <org:voice> element that contains the organization's voice telephone number. The detailed format of this element is described in Section 2.5 of [RFC5733]

- An OPTIONAL <org:fax> element that contains the organization's facsimile telephone number.

- An OPTIONAL <org:email> element that contains the organization's email address. The detailed format of this element is described in section 2.6 of [RFC5733].

- An OPTIONAL <org:url> element that contains the URL to the website of the organization. The detailed format of this element is described in [RFC3986].

- Zero or more <org:contact> elements that contain identifiers for the contact objects associated with the organization object.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <org:create
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:        <org:id>res1523</org:id>
C:        <org:role>
C:          <org:type>reseller</org:type>
C:        </org:role>
C:        <org:parentId>1523res</org:parentId>
C:        <org:postalInfo type="int">
C:          <org:name>Example Organization Inc.</org:name>
C:          <org:addr>
C:            <org:street>123 Example Dr.</org:street>
C:            <org:street>Suite 100</org:street>
C:            <org:city>Dulles</org:city>
C:            <org:sp>VA</org:sp>
C:            <org:pc>20166-6503</org:pc>
C:            <org:cc>US</org:cc>
C:          </org:addr>
C:        </org:postalInfo>
C:        <org:voice x="1234">+1.7035555555</org:voice>
C:        <org:fax>+1.7035555556</org:fax>
C:        <org:email>contact@organization.example</org:email>
C:        <org:url>https://organization.example</org:url>
C:        <org:contact type="admin">sh8013</org:contact>
C:        <org:contact type="billing">sh8013</org:contact>
C:      </org: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 <org:creData> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:creData> element contains the following child elements:

- An <org:id> element that contains the server-unique identifier for the created organization, as defined in Section 3.1.

- An <org:crDate> element that contains the date and time of organization-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 lang="en">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <org:creData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>res1523</org:id>
S:        <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:      </org: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 cannot be processed for any reason.

#### 4.2.2. EPP <delete> Command

The EPP <delete> command provides a transform operation that allows a client to delete an organization object. In addition to the standard EPP command elements, the <delete> command MUST contain an <org:delete> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:delete> element MUST contain the following child element:

- An <org:id> element that contains the server-unique identifier of the organization object to be deleted, as defined in Section 3.1.

An organization object MUST NOT be deleted if it is associated with other known objects. An associated organization MUST NOT be deleted until associations with other known objects have been broken. A server MUST notify clients that object relationships exist by sending a 2305 error response code when a <delete> command is attempted and fails due to existing object relationships.

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:      <org:delete
C:       xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:        <org:id>res1523</org:id>
C:      </org:delete>
C:    </delete>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <delete> command 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 lang="en">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 cannot be processed for any reason.

#### 4.2.3. EPP <renew> Command

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

#### 4.2.4. EPP <transfer> Command

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

#### 4.2.5. EPP <update> Command

The EPP <update> command provides a transform operation that allows a client to modify the attributes of an organization object. In addition to the standard EPP command elements, the <update> command MUST contain a <org:update> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:update> element contains the following child elements:

- An <org:id> element that contains the server-unique identifier of the organization object to be updated, as defined in Section 3.1.
- An OPTIONAL <org:add> element that contains attribute values to be added to the object.
- An OPTIONAL <org:rem> element that contains attribute values to be removed from the object.
- An OPTIONAL <org:chg> element that contains attribute values to be changed.

At least one <org:add>, <org:rem> or <org:chg> element MUST be provided if the command is not being extended. All of these elements MAY be omitted if an <update> extension is present. The OPTIONAL <org:add> and <org:rem> elements contain the following child elements:

- Zero or more <org:contact> elements that contain the identifiers for contact objects to be associated with or removed from the organization object. Contact object identifiers MUST be known to the server before the contact object can be associated with the organization object.
- Zero or more <org:role> elements that contain the role type, role statuses and optional role id of the organization.
  - An <org:type> element that contains the role type of the organization, as defined in Section 3.2. The role type uniquely identifies the role to update.
  - Zero or more <org:status> elements that contain the role statuses. The values of the role status are defined in Section 3.5.
  - An OPTIONAL <org:roleID> element that contains a third-party-assigned identifier, such as IANA ID for registrars, as defined in Section 3.2.3.

- Zero or more <org:status> elements that contain the operational status of the organization.

An OPTIONAL <org:chg> element contains the following child elements, where at least one child element MUST be present:

- An OPTIONAL <org:parentId> element that contains the identifier of the parent object.

- Zero to two <org:postalInfo> elements that contain postal-address information. Two elements are provided so that address information can be provided in both internationalized and localized forms; a "type" attribute is used to identify the two forms. If an internationalized form (type="int") is provided, element content MUST be represented in a subset of Unicode in the range U+0020 - U+007E. If a localized form (type="loc") is provided, element content MAY be represented in unrestricted UTF-8. The change of the postal info is defined as a replacement of that postal info element with the contents of the sub-elements included in the update command. An empty <org:postalInfo> element is supported to allow a type of postal info to be removed. The <org:postalInfo> element contains the following child elements:

  * An <org:name> element that contains the name of the organization.

  * An OPTIONAL <org:addr> element that contains address information associated with the organization. A <org:addr> element contains the following child elements:

    + One, two, or three <org:street> elements that contain the organization's street address.

    + An <org:city> element that contains the organization's city.

    + An OPTIONAL <org:sp> element that contains the organization's state or province.

    + An OPTIONAL <org:pc> element that contains the organization's postal code.

    + An <org:cc> element that contains the alpha-2 organization's country code. The detailed format of this element is described in section 2.4.3 of [RFC5733].

- An OPTIONAL <org:voice> element that contains the organization's voice telephone number. The detailed format of this element is described in Section 2.5 of [RFC5733]

- An OPTIONAL <org:fax> element that contains the organization's facsimile telephone number.

- An OPTIONAL <org:email> element that contains the organization's email address. The detailed format of this element is described in section 2.6 of [RFC5733].

- An OPTIONAL <org:url> element that contains the URL to the website of the organization. The detailed format of this element is described in [RFC3986]

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:      <org:update
C:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
C:        <org:id>res1523</org:id>
C:        <org:add>
C:          <org:contact type="tech">sh8013</org:contact>
C:          <org:role>
C:            <org:type>privacyproxy</org:type>
C:            <org:status>clientLinkProhibited</org:status>
C:          </org:role>
C:          <org:status>clientLinkProhibited</org:status>
C:        </org:add>
C:        <org:rem>
C:          <org:contact type="billing">sh8014</org:contact>
C:          <org:role>
C:            <org:type>reseller</org:type>
C:          </org:role>
C:        </org:rem>
C:        <org:chg>
C:          <org:postalInfo type="int">
C:            <org:addr>
C:              <org:street>124 Example Dr.</org:street>
C:              <org:street>Suite 200</org:street>
C:              <org:city>Dulles</org:city>
C:              <org:sp>VA</org:sp>
C:              <org:pc>20166-6503</org:pc>
C:              <org:cc>US</org:cc>
C:            </org:addr>
C:          </org:postalInfo>
C:          <org:voice>+1.7034444444</org:voice>
C:          <org:fax/>
C:        </org:chg>
C:      </org: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> 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 lang="en">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 cannot be processed for any reason.

### 4.3. Offline Review of Requested Actions

Commands are processed by a server in the order they are received from a client. Though an immediate response confirming receipt and processing of the command is produced by the server, a server operator MAY perform an offline review of requested transform commands before completing the requested action. In such situations, the response from the server MUST clearly note that the transform command has been received and processed, but the requested action is pending. The status in the response of the corresponding object MUST clearly reflect processing of the pending action. The server MUST notify the client when offline processing of the action has been completed.

Examples describing a <create> command that requires offline review are included here. Note the result code and message returned in response to the <create> 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="1001">
S:      <msg lang="en">Command completed successfully;
S:        action pending</msg>
S:    </result>
S:    <resData>
S:      <org:creData
S:        xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id>res1523</org:id>
S:        <org:crDate>1999-04-03T22:00:00.0Z</org:crDate>
S:      </org:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The status of the organization object after returning this response MUST include "pendingCreate". The server operator reviews the request offline, and informs the client of the outcome of the review by queuing a service message for retrieval via the <poll> command; it MAY additionally use an out-of-band mechanism to inform the client of the outcome.

The service message MUST contain text that describes the notification in the child <msg> element of the response <msgQ> element. In addition, the EPP <resData> element MUST contain a child <org:panData> element. This element or its ancestor element MUST identify the organization namespace "urn:ietf:params:xml:ns:epp:org-1.0". The <org:panData> element contains the following child elements:

- An <org:id> element that contains the server-unique identifier of the organization object. The <org:id> element contains a REQUIRED "paResult" attribute. A positive boolean value indicates that the request has been approved and completed. A negative boolean value indicates that the request has been denied and the requested action has not been taken.

- An <org:paTRID> element that contains the client transaction identifier and server transaction identifier returned with the original response to process the command. The client transaction

identifier is OPTIONAL and will only be returned if the client provided an identifier with the original <create> command.

- An <org:paDate> element that contains the date and time describing when review of the requested action was completed.

Example "review completed" service message:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg lang="en">Command completed successfully;
S:        ack to dequeue</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>1999-04-04T22:01:00.0Z</qDate>
S:      <msg>Pending action completed successfully.</msg>
S:    </msgQ>
S:    <resData>
S:      <org:panData
S:       xmlns:org="urn:ietf:params:xml:ns:epp:org-1.0">
S:        <org:id paResult="1">res1523</org:id>
S:        <org:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
S:        </org:paTRID>
S:        <org:paDate>1999-04-04T22:00:00.0Z</org:paDate>
S:      </org:panData>
S:    </resData>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

## 5. Formal Syntax

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

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

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

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

 <annotation>
   <documentation>
     Extensible Provisioning Protocol v1.0
     organization provisioning schema.
   </documentation>
 </annotation>

<!--
Child elements found in EPP commands.
-->
 <element name="create" type="org:createType"/>
 <element name="delete" type="org:sIDType"/>
 <element name="update" type="org:updateType"/>
 <element name="check" type="org:mIDType"/>
 <element name="info" type="org:infoType"/>
 <element name="panData" type="org:panDataType"/>

<!--
Utility types.
-->
 <simpleType name="statusType">
   <restriction base="token">
     <enumeration value="ok"/>
     <enumeration value="hold"/>
     <enumeration value="terminated"/>
     <enumeration value="clientDeleteProhibited"/>
     <enumeration value="clientUpdateProhibited"/>
     <enumeration value="clientLinkProhibited"/>
     <enumeration value="linked"/>
     <enumeration value="pendingCreate"/>
     <enumeration value="pendingUpdate"/>
     <enumeration value="pendingDelete"/>
```

```
        <enumeration value="serverDeleteProhibited"/>
        <enumeration value="serverUpdateProhibited"/>
        <enumeration value="serverLinkProhibited"/>
      </restriction>
    </simpleType>

    <simpleType name="roleStatusType">
      <restriction base="token">
        <enumeration value="ok"/>
        <enumeration value="clientLinkProhibited"/>
        <enumeration value="linked"/>
        <enumeration value="serverLinkProhibited"/>
      </restriction>
    </simpleType>

    <complexType name="roleType">
      <sequence>
        <element name="type" type="token"/>
        <element name="status" type="org:roleStatusType"
          minOccurs="0" maxOccurs="3"/>
        <element name="roleID" type="token" minOccurs="0"/>
      </sequence>
    </complexType>

    <complexType name="postalInfoType">
      <sequence>
        <element name="name"
          type="org:postalLineType"/>
        <element name="addr"
          type="org:addrType" minOccurs="0"/>
      </sequence>
      <attribute name="type"
        type="org:postalInfoEnumType"
        use="required"/>
    </complexType>

    <complexType name="contactType">
      <simpleContent>
        <extension base="eppcom:clIDType">
          <attribute name="type" type="org:contactAttrType"
            use="required"/>
          <attribute name="typeName" type="token"/>
        </extension>
      </simpleContent>
    </complexType>

    <simpleType name="contactAttrType">
      <restriction base="token">
```

```
          <enumeration value="admin"/>
          <enumeration value="billing"/>
          <enumeration value="tech"/>
          <enumeration value="abuse"/>
          <enumeration value="custom"/>
        </restriction>
      </simpleType>

      <complexType name="e164Type">
        <simpleContent>
          <extension base="org:e164StringType">
            <attribute name="x" type="token" />
          </extension>
        </simpleContent>
      </complexType>

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

      <simpleType name="postalLineType">
        <restriction base="normalizedString">
          <minLength value="1" />
          <maxLength value="255" />
        </restriction>
      </simpleType>

      <simpleType name="optPostalLineType">
        <restriction base="normalizedString">
          <maxLength value="255" />
        </restriction>
      </simpleType>

      <simpleType name="pcType">
        <restriction base="token">
          <maxLength value="16" />
        </restriction>
      </simpleType>

      <simpleType name="ccType">
        <restriction base="token">
          <length value="2" />
        </restriction>
      </simpleType>
```

```
      <complexType name="addrType">
        <sequence>
          <element name="street" type="org:optPostalLineType"
            minOccurs="0" maxOccurs="3" />
          <element name="city" type="org:postalLineType" />
          <element name="sp" type="org:optPostalLineType"
            minOccurs="0" />
          <element name="pc" type="org:pcType"
            minOccurs="0" />
          <element name="cc" type="org:ccType" />
        </sequence>
      </complexType>

      <simpleType name="postalInfoEnumType">
        <restriction base="token">
          <enumeration value="loc" />
          <enumeration value="int" />
        </restriction>
      </simpleType>

     <!--
     Child element of commands that require only an identifier.
     -->
      <complexType name="sIDType">
        <sequence>
          <element name="id" type="eppcom:clIDType"/>
        </sequence>
      </complexType>

     <!--
     Child element of commands that accept multiple identifiers.
     -->
      <complexType name="mIDType">
        <sequence>
          <element name="id"
            type="eppcom:clIDType" maxOccurs="unbounded"/>
        </sequence>
      </complexType>

     <!--
     Pending action notification response elements.
     -->
      <complexType name="panDataType">
        <sequence>
          <element name="id" type="org:paCLIDType"/>
          <element name="paTRID" type="epp:trIDType"/>
          <element name="paDate" type="dateTime"/>
        </sequence>
```

```
      </complexType>

      <complexType name="paCLIDType">
        <simpleContent>
          <extension base="eppcom:clIDType">
            <attribute name="paResult" type="boolean"
             use="required"/>
          </extension>
        </simpleContent>
      </complexType>

     <!--
     Child elements of the <info> commands.
     -->
      <complexType name="infoType">
        <sequence>
          <element name="id"
            type="eppcom:clIDType"/>
        </sequence>
      </complexType>

     <!--
     Child elements of the <create> command.
     -->
      <complexType name="createType">
        <sequence>
          <element name="id"
            type="eppcom:clIDType"/>
          <element name="role"
            type="org:roleType" maxOccurs="unbounded"/>
          <element name="status"
            type="org:statusType" minOccurs="0" maxOccurs="4"/>
          <element name="parentId"
            type="eppcom:clIDType" minOccurs="0"/>
          <element name="postalInfo"
            type="org:postalInfoType" minOccurs="0" maxOccurs="2"/>
          <element name="voice"
            type="org:e164Type" minOccurs="0"/>
          <element name="fax"
            type="org:e164Type" minOccurs="0"/>
          <element name="email"
            type="eppcom:minTokenType" minOccurs="0"/>
          <element name="url"
            type="anyURI" minOccurs="0"/>
          <element name="contact"
            type="org:contactType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
```

```
    </complexType>

  <!--
  Child elements of the <update> command.
  -->
    <complexType name="updateType">
      <sequence>
        <element name="id"
          type="eppcom:clIDType"/>
        <element name="add"
          type="org:addRemType" minOccurs="0"/>
        <element name="rem"
          type="org:addRemType" minOccurs="0"/>
        <element name="chg"
          type="org:chgType" minOccurs="0"/>
      </sequence>
    </complexType>

  <!--
  Data elements that can be added or removed.
  -->
    <complexType name="addRemType">
      <sequence>
        <element name="contact"
          type="org:contactType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="role" type="org:roleType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="status" type="org:statusType"
          minOccurs="0" maxOccurs="9"/>
      </sequence>
    </complexType>

  <!--
  Data elements that can be changed.
  -->
    <complexType name="chgType">
      <sequence>
        <element name="parentId"
          type="eppcom:clIDType" minOccurs="0"/>
        <element name="postalInfo"
          type="org:chgPostalInfoType"
          minOccurs="0" maxOccurs="2"/>
        <element name="voice"
          type="org:e164Type" minOccurs="0"/>
        <element name="fax"
          type="org:e164Type" minOccurs="0"/>
        <element name="email"
          type="eppcom:minTokenType" minOccurs="0"/>
```

```
          <element name="url"
            type="anyURI" minOccurs="0"/>
        </sequence>
       </complexType>

       <complexType name="chgPostalInfoType">
         <sequence>
           <element name="name"
             type="org:postalLineType" minOccurs="0"/>
           <element name="addr"
             type="org:addrType" minOccurs="0"/>
         </sequence>
         <attribute name="type"
           type="org:postalInfoEnumType" use="required"/>
       </complexType>

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

      <!--
      <check> response elements.
      -->
       <complexType name="chkDataType">
         <sequence>
           <element name="cd" type="org:checkType"
             maxOccurs="unbounded" />
         </sequence>
       </complexType>

       <complexType name="checkType">
         <sequence>
           <element name="id" type="org:checkIDType" />
           <element name="reason" type="eppcom:reasonType"
             minOccurs="0" />
         </sequence>
       </complexType>

       <complexType name="checkIDType">
         <simpleContent>
           <extension base="eppcom:clIDType">
             <attribute name="avail" type="boolean"
               use="required" />
           </extension>
         </simpleContent>
```

```
      </complexType>

     <!--
     <info> response elements.
     -->
      <complexType name="infDataType">
        <sequence>
          <element name="id"
            type="eppcom:clIDType"/>
          <element name="roid"
            type="eppcom:roidType"/>
          <element name="role"
            type="org:roleType" maxOccurs="unbounded"/>
          <element name="status"
            type="org:statusType" maxOccurs="9"/>
          <element name="parentId"
            type="eppcom:clIDType" minOccurs="0"/>
          <element name="postalInfo"
            type="org:postalInfoType" minOccurs="0" maxOccurs="2"/>
          <element name="voice"
            type="org:e164Type" minOccurs="0"/>
          <element name="fax"
            type="org:e164Type" minOccurs="0"/>
          <element name="email"
            type="eppcom:minTokenType" minOccurs="0"/>
          <element name="url"
            type="anyURI" minOccurs="0"/>
          <element name="contact"
            type="org:contactType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="clID"
            type="eppcom:clIDType" minOccurs="0"/>
          <element name="crID"
            type="eppcom:clIDType"/>
          <element name="crDate"
            type="dateTime"/>
          <element name="upID"
            type="eppcom:clIDType" minOccurs="0"/>
          <element name="upDate"
            type="dateTime" minOccurs="0"/>
        </sequence>
      </complexType>
     <!--
     <create> response elements.
     -->
      <complexType name="creDataType">
        <sequence>
          <element name="id" type="eppcom:clIDType" />
          <element name="crDate" type="dateTime" />
```

```
      </sequence>
    </complexType>

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

## 6. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 [RFC3629] and UTF-16 [RFC2781]. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED.

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

## 7. IANA Considerations

### 7.1. XML Namespace

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

Registration request for the organization namespace:

URI: urn:ietf:params:xml:ns:epp:org-1.0

Registrant Contact: IESG

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

Registration request for the organization XML schema:

URI: urn:ietf:params:xml:schema:epp:org-1.0

Registrant Contact: IESG

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

### 7.2. 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: Extensible Provisioning Protocol (EPP) Organization Mapping

Document status: Standards Track

Reference: RFCXXXX (please replace "XXXX" with the RFC number for this document after a number is assigned by the RFC Editor)

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

### 7.3. Role Type Values Registry

IANA has created a new category of protocol registry for values of the organization roles. The name of this registry is "EPP Organization Role Values". The registration policy for this registry is "Expert Review" [RFC8126].

#### 7.3.1. Registration Template

Value: the string value being registered.

Description: Brief description of the organization role values.

Registrant Name: For IETF RFCs, state "IESG". For others, give the name of the responsible party.

Registrant Contact Information: an email address, postal address, or some other information to be used to contact the registrant.

#### 7.3.2. Initial Registry Contents

Followings are the initial registry contents:

Value: registrar

Description: The entity object instance represents the authority responsible for the registration in the registry.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: reseller

Description: The entity object instance represents a third party through which the registration was conducted (i.e., not the registry or registrar).

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: privacyproxy

Description: The entity object instance represents a third-party who could help to register a domain without exposing the registrants' private information.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

Value: dns-operator

Description: The entity object instance represents a third-party DNS operator that maintains the name servers and zone data on behalf of a registrant.

Registrant Name: IESG

Registrant Contact Information: iesg@ietf.org

## 8. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

### 8.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

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

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

### 8.2. CNNIC Implementation

Organization: CNNIC

Name: EPP Organization Mapping

Description: CNNIC is trying to update EPP organization mapping from previous reseller mapping according to this document.

Level of maturity: Development

Coverage: EPP organization mapping

Contact: zhouguiqing@cnnic.cn

## 9. Security Considerations

The organization object may have personally identifiable information, such as <org:contact>. This information is not a required element in this document which can be provided on a voluntary basis. If it is provided, both client and server MUST ensure that authorization information is stored and exchanged with high-grade encryption mechanisms to provide privacy services, which is specified in [RFC5733]. The security considerations described in [RFC5730] or those caused by the protocol layers used by EPP will apply to this specification as well.

## 10. Acknowledgment

The authors would like to thank Rik Ribbers, Marc Groeneweg, Patrick Mevzek, Antoin Verschuren and Scott Hollenbeck for their careful review and valuable comments.

## 11. References

### 11.1. Normative References

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[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>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, ""Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204", February 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

[W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, ""XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028", October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

[W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, ""XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028", October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

### 11.2. Informative References

[RFC2781] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, DOI 10.17487/RFC2781, February 2000, <https://www.rfc-editor.org/info/rfc2781>.

[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>.

## Appendix A. Change Log

Initial -00: Individual document submitted.

-01:

* Updated abstract text.
* Added sentences to avoid loop of parent identifiers in section 3.4.
* Revised typos in section 3.6.
* Added explanation of contact type attribute in section 4.1.2.
* Updated <info> responses.
* Deleted description of <transfer> command in section 4.1 and 4.2.
* Deleted whoisInfo disclose type in XML schema.
* Deleted maxOccurs of addRemType.
* Deleted extra "OPTIONAL" in section 4.2.5.
* Updated typos in <update> response.

-02:

* Changed author information.
* Updated url definition.
* Updated XML schema.

-03:

* Changed author information.
* Updated section 3.1.
* Refactoried the XSD file. Added <chgPostalInfoType> element.
* Added acknowledgment.

WG document-00: WG document submitted

WG document-01: Keep document alive for further discussion. Reseller object or entity object with multiple roles?

Organization WG document-00: Change to a generic organization object mapping.

Organization WG document-01: Added "Implementation Status" section.

Organization WG document-02: Accepted some of the feedbacks on the mailing list.

Organization WG document-03:

* Updated section 3.2, changed the structure of organization role.

* Updated section 4.2.5 for the "add", "rem" and "chg" example.

* Updated section 5 of formal syntax.

* Updated section 7.2 for the registration template and initial values.

* Updated section 8 of implementation status.

Organization WG document-04:

* Updated section 3.2, changed the structure of organization role.

* Updated references.

* Updated section 8 of implementation status.

Organization WG document-05:

* Updated the description of <org:status> of a role.

* Removed the third paragraph of "Implementation Status".

* Remove the Informative Reference to draft-ietf-regext-reseller from the draft.

Organization WG document-06:

* Updated typos.

* Added "Query" for "<Transfer> Query Command".

* Change "Registrant Contact" to IESG in section 7.1.

* Modified section 7.2.

Organization WG document-07:

* Updated typos.

* Added dns-operator in section 7.1.

* Added "OPTIONAL" for <org:addr>

Organization WG document-08:

* Updated "Offline Review of Requested Actions".

Organization WG document-09:

* Updated "This element or its ancestor element MUST identify the organization namespace." in section 4.1.1 and other parts of this document.

* Updated text in section 2 match RFC 8174.

* Modified "roleid" to "roleID".

* Updated text about loops in section 3.6.

* Referred section 2.5 of RFC5733 for voice format.

* Updated XML schema for the maxOccurs value of "reason" element.

* Updated section 7.3.

* Replaced "http" with "https" in the examples.

* Updated writing typos.

* Modified XML namespace and schema.

Organization WG document-10:

* Modified XML namespace and schema.

* Removed the maxOccurs value of "reason" element.

Organization WG document-11:

* Typo of RFC2781 and moved this reference in "Informative References".

* "Loops MUST be prohibited." in section 3.6.

Organization WG document-12:

* Removed "OPTIONAL" when "zero or more" or "zero to two" appears.

* Updated the "Organization Status Values" text.

* Updated the full xml namespace.

* Updated the text in "Offline review".

* Updated the text in "Security Considerations".

* Added "Document satus" and "Reference" in section "EPP Extension Registry".

* Added references of RFC3688,RFC3986 and RFC5646.

Authors' Addresses

Linlin Zhou
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Email: zhoulinlin@cnnic.cn

Ning Kong
Consultant

Email: ietfing@gmail.com

Guiqing Zhou
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Email: zhouguiqing@cnnic.cn

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

Email: yaojk@cnnic.cn

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

Email: jgould@verisign.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: June 8, 2019

L. Zhou
CNNIC
N. Kong
Consultant
J. Wei
J. Yao
CNNIC
J. Gould
Verisign, Inc.

December 5, 2018

# Organization Extension for the Extensible Provisioning Protocol (EPP) draft-ietf-regext-org-ext-11

## Abstract

This document describes an extension to Extensible Provisioning Protocol (EPP) object mappings, which is designed to support assigning an organization to any existing object (domain, host, contact) as well as any future objects.

## 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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on June 8, 2019.

## Copyright Notice

Table of Contents



## 1. Introduction

In the business model of domain registration, we usually have three roles of entities: a registrant, a registrar and a registry, as defined in section 9 of [ID.draft-ietf-dnsop-terminology-bis]. There may be other roles of entities involved in the domain registration process, such as resellers, DNS operators in section 9 of [ID.draft-ietf-dnsop-terminology-bis], privacy proxies, etc.

A domain reseller is an individual or a company that acts as an agent for accredited registrars. DNS operator is defined in section 9 of [ID.draft-ietf-dnsop-terminology-bis]. A privacy proxy is an entity used for domain registrations to protect the private information of the individuals and organizations. These kind of entities are defined as "organizations" with different role types in this document.

In order to facilitate provisioning and management of organization information in a shared central repository, this document proposes an organization extension mapping for any Extensible Provisioning Protocol (EPP) object like domain names in [RFC5731], hosts in [RFC5732] and contacts in [RFC5733]. The examples provided in this document are used for the domain object for illustration purpose. The host and contact object could be extended in the same way with the domain object.

Organization object identifiers defined in [ID.draft-ietf-regext-org] MUST be known to the server before the organization object can be associated with the EPP object.

This document is specified using the XML 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

## 2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

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

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case.

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

## 3. Object Attributes

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

### 3.1. Organization Identifier

Organization identifier provides the ID of an organization. Its corresponding element is <orgext:id> which refers to the <org:id> element defined in [ID.draft-ietf-regext-org]. All organization objects are identified by a server-unique identifier. A "role" attribute is used to represent the relationship that the organization has to the EPP object. Any given object MUST have at most one associated organization ID for any given role value.

## 4. 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 assigning organizations to EPP objects.

### 4.1. EPP Query Commands

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

#### 4.1.1. EPP <check> Command

This extension does not add any elements to the EPP <check> command or <check> response described in the EPP object mapping.

#### 4.1.2. EPP <info> Command

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

When an <info> command has been processed successfully, the EPP <resData> element MUST contain child elements as described in the EPP object extensions. In addition, the EPP <extension> element SHOULD contain a child <orgext:infData> element. This element or its ancestor element MUST identify the extension namespace "urn:ietf:params:xml:ns:epp:orgext-1.0" if the object has data

associated with this extension and based on server policy. The <orgext:infData> element contains the following child elements:

- Zero or more <orgext:id> elements are allowed that contain the identifier of the organization, as defined in Section 3.1. The "role" attribute is used to represent the relationship that the organization has to the object. See Section 7.3 in [ID.draft-ietf-regext-org] for a list of values.

Example <info> response for an authorized client with multiple organizations:

```
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 lang="en-US">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="billing">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:        </domain:ns>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2015-02-06T04:01:21.0Z</domain:crDate>
S:        <domain:exDate>2018-02-06T04:01:21.0Z</domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <orgext:infData
S:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
S:        <orgext:id role="reseller">reseller1523</orgext:id>
S:        <orgext:id role="privacyproxy">proxy2935</orgext:id>
S:      </orgext:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ngcl-IvJjzMZc</clTRID>
S:      <svTRID>test142AWQONJZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> response for an authorized client with no organization:

```
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 lang="en-US">Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="billing">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:        </domain:ns>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2015-02-06T04:01:21.0Z</domain:crDate>
S:        <domain:exDate>2018-02-06T04:01:21.0Z</domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <orgext:infData
S:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0"/>
S:    </extension>
S:    <trID>
S:      <clTRID>ngcl-IvJjzMZc</clTRID>
S:      <svTRID>test142AWQONJZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

#### 4.1.3. EPP <transfer> Query Command

This extension does not add any elements to the EPP <transfer> query command or <transfer> query response described in the EPP object mapping.

### 4.2. EPP Transform Commands

EPP provides five commands to transform EPP 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 the object sponsorship changes, and <update> to change information associated with an object.

#### 4.2.1. EPP <create> Command

This extension defines additional elements for the EPP <create> command described in the EPP object extensions. No additional elements are defined for the EPP <create> response.

The EPP <create> command provides a transform operation that allows a client to create an object. In addition to the EPP command elements described in the EPP object extensions, the command MUST contain an <extension> element, and the <extension> element MUST contain a child <orgext:create> element. This element or its ancestor element MUST identify the extension namespace "urn:ietf:params:xml:ns:epp:orgext-1.0" if the client wants to associate data defined in this extension to the object. The <orgext:create> element contains the following child elements:

- One or more <orgext:id> elements that contain the identifier of the organization, as defined in Section 3.1. The "role" attribute is used to represent the relationship that the organization has to the object. See Section 7.3 in [ID.draft-ietf-regext-org] for a list of values.

Example <create> Command with only one organization:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:period unit="y">3</domain:period>
C:        <domain:ns>
C:          <domain:hostObj>ns1.example.com</domain:hostObj>
C:        </domain:ns>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:contact type="billing">sh8013</domain:contact>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <orgext:create
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:id role="reseller">reseller1523</orgext:id>
C:      </orgext:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <create> Command with multiple organizations:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:period unit="y">3</domain:period>
C:        <domain:ns>
C:          <domain:hostObj>ns1.example.com</domain:hostObj>
C:        </domain:ns>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:contact type="billing">sh8013</domain:contact>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <orgext:create
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:id role="reseller">reseller1523</orgext:id>
C:        <orgext:id role="privacyproxy">proxy2935</orgext:id>
C:      </orgext:create>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <create> command has been processed successfully, the EPP response is as described in the EPP object extension.

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

#### 4.2.2. EPP <delete> Command

This extension does not add any elements to the EPP <delete> command or <delete> response described in the EPP object mapping.

#### 4.2.3. EPP <renew> Command

This extension does not add any elements to the EPP <renew> command or <renew> response described in the EPP object mapping.

#### 4.2.4. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> command or <transfer> response described in the EPP object mapping, but after a successful transfer of an object with an assigned organization, the handling of the assigned organization is dependent on the organization roles and server policy.

#### 4.2.5. EPP <update> Command

This extension defines additional elements for the EPP <update> command described in the EPP domain mapping [RFC5731], host mapping [RFC5732] and contact mapping [RFC5733]. No additional elements are defined for the EPP <update> response.

The EPP <update> command provides a transform operation that allows a client to modify the attributes of an object. In addition to the EPP <update> command elements, the command MUST contain an <extension> element, and the <extension> element MUST contain a child <orgext:update> element. This element or its ancestor element MUST identify the extension namespace "urn:ietf:params:xml:ns:epp:orgext-1.0" if the client wants to update the object with data defined in this extension. The <orgext:update> element contains the following child elements:

- An OPTIONAL <orgext:add> element that contains one or more <orgext:id> elements, as defined in Section 3.1, that add non-existent organization roles to the object. The <orgext:id> element MUST have a non-empty organization identifier value. The server SHOULD validate that the <orgext:id> element role does not exist.

- An OPTIONAL <orgext:rem> element that contains one or more <orgext:id> elements, as defined in Section 3.1, that remove organization roles from the object. The <orgext:id> element MAY have an empty organization identifier value. The server SHOULD validate the existence of the <orgext:id> element role and the organization identifier if provided.

- An OPTIONAL <orgext:chg> element that contains one or more <orgext:id> elements, as defined in Section 3.1, that change organization role identifiers for the object. The existing organization identifier value will be replaced for the defined role. The server SHOULD validate the existence of the <orgext:id> element role.

At least one <orgext:add>, <orgext:rem> or <orgext:chg> element MUST be provided.

Example <update> command, adding a reseller:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <orgext:update
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:add>
C:          <orgext:id role="reseller">reseller1523</orgext:id>
C:        </orgext:add>
C:      </orgext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <update> command, adding multiple organizations:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <orgext:update
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:add>
C:          <orgext:id role="reseller">reseller1523</orgext:id>
C:          <orgext:id role="privacyproxy">proxy2935</orgext:id>
C:        </orgext:add>
C:      </orgext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <update> command, removing a reseller:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <orgext:update
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:rem>
C:          <orgext:id role="reseller"/>
C:        </orgext:rem>
C:      </orgext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <update> command, removing multiple organizations:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <orgext:update
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:rem>
C:          <orgext:id role="reseller"/>
C:          <orgext:id role="privacyproxy"/>
C:        </orgext:rem>
C:      </orgext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <update> command, updating reseller identifier:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <orgext:update
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:chg>
C:          <orgext:id role="reseller">reseller1523</orgext:id>
C:        </orgext:chg>
C:      </orgext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <update> command, updating multiple organization identifiers:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:      </domain:update>
C:    </update>
C:    <extension>
C:      <orgext:update
C:        xmlns:orgext="urn:ietf:params:xml:ns:epp:orgext-1.0">
C:        <orgext:chg>
C:          <orgext:id role="reseller">reseller1523</orgext:id>
C:          <orgext:id role="privacyproxy">proxy2935</orgext:id>
C:        </orgext:chg>
C:     </orgext:update>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When an extended <update> command has been processed successfully, the EPP response is as described in the EPP object extension.

An EPP error response MUST be returned if an <update> command cannot be processed for any reason. An attempt to add one organization ID or multiple organization IDs with a particular role value when at least one of them already exists does not change the object at all. A server SHOULD notify clients that object relationships exist by sending a 2305 error response code. An attempt to remove an organization ID or multiple organization IDs with a particular role value when at least one of them does not exist does not change the object at all. A server SHOULD notify clients that object relationships does not exist by sending a 2305 error response code. An attempt to change an organization ID or multiple organization IDs with a particular role value when at least one of them does not exist does not change the object at all. A server SHOULD notify clients that object relationships does not exist by sending a 2305 error response code. Response format with error value elements is defined in Section 2.6 of [RFC5730].

## 5. Formal Syntax

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

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

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

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Organization Extension Schema v1.0
    </documentation>
  </annotation>

  <!-- Child elements found in EPP commands. -->
```

```
<element
  name="create"
  type="orgext:createType"/>
<element
  name="update"
  type="orgext:updateType"/>

<!--
  Organization identifier with required role
-->
<complexType name="orgIdType">
  <simpleContent>
    <extension base="token">
      <attribute
        name="role"
        type="token"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
  Child elements of the <orgext:create> command
  All elements must be present at time of creation
-->
<complexType name="createType">
  <sequence>
    <!-- agent identifier or the organization,
      e.g. registrar, reseller, privacy proxy, etc. -->
    <element
      name="id"
      type="orgext:orgIdType"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!--
  Child elements of <orgext:update> command
-->
<complexType name="updateType">
  <sequence>
    <element
      name="add"
      type="orgext:addRemChgType"
      minOccurs="0"
    />
    <element
      name="rem"
```

```
          type="orgext:addRemChgType"
          minOccurs="0"
        />
        <element
          name="chg"
          type="orgext:addRemChgType"
          minOccurs="0"
        />
      </sequence>
    </complexType>

    <complexType name="addRemChgType">
      <sequence>
        <!-- agent identifier of the organization,
          e.g. registrar, reseller, privacy proxy, etc. -->
        <element
          name="id"
          type="orgext:orgIdType"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <!-- Child response element -->
    <element
      name="infData"
      type="orgext:infDataType"/>

    <!-- <orgext:infData> response elements -->
    <complexType name="infDataType">
      <sequence>
        <!-- agent identifier the organization,
          e.g. registrar, reseller, privacy proxy, etc. -->
        <element
          name="id"
          type="orgext:orgIdType"
          minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </complexType>

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

## 6. Internationalization Considerations

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

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

## 7. IANA Considerations

### 7.1. XML Namespace

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

Registration request for the organization extension namespace:

URI: urn:ietf:params:xml:ns:epp:orgext-1.0

Registrant Contact: IESG

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

Registration request for the organization XML schema:

URI: urn:ietf:params:xml:schema:epp:orgext-1.0

Registrant Contact: IESG

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

### 7.2. 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: Organization Extension for the Extensible Provisioning Protocol (EPP)

Document status: Standards Track

Reference: RFCXXXX (please replace "XXXX" with the RFC number for this document after a number is assigned by the RFC Editor)

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

## 8. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

### 8.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

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

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

### 8.2. CNNIC Implementation

Organization: CNNIC

Name: Organization Extension for EPP

Description: CNNIC is trying to update organization extension from previous reseller extension according to this document.

Level of maturity: Development

Coverage: Organization extension for EPP

Contact: zhouguiqing@cnnic.cn

## 9. Security Considerations

The object mapping extension described in this document does not provide any other security services or introduce any additional considerations beyond those described by [RFC5730], [RFC5731], [RFC5732] and [RFC5733] or those caused by the protocol layers used by EPP.

## 10. Acknowledgment

The authors would like to thank Rik Ribbers, Marc Groeneweg, Patrick Mevzek, Antoin Verschuren and Scott Hollenbeck for their careful review and valuable comments.

## 11. References

### 11.1. Normative References

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

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

[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>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[W3C.REC-xml-20040204]
Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, ""Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204", February 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

[W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, ""XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028", October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

[W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, ""XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028", October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

### 11.2. Informative References

[ID.draft-ietf-dnsop-terminology-bis]
Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", September 2018, <http://tools.ietf.org/html/draft-ietf-dnsop-terminology-bis>.

[ID.draft-ietf-regext-org]
Zhou, L., Kong, N., Zhou, G., Yao, J., and J. Gould, "Extensible Provisioning Protocol (EPP) Reseller Mapping", November 2018, <http://tools.ietf.org/html/draft-ietf-regext-org>.

[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>.

## Appendix A. Change Log

Initial -00: Individual document submitted.

-01:

* Updated abstract and introduction.
* Revised typos in info response.
* Added explanations on how to process reseller extension after successful transfer operation.
* Modified <update> explanation.
* Deleted reseller name element in <create> and <update> commands.
* Removed some inaccurate comments from xml schema.
* Modified the element name of reseller id and reseller name.

-02:

* Changed author information.

* Updated xml typos <reseller:infData> to <resellerext:infData> in <info> response.

-03:

* Changed author information.

* Updated section 3.1.

* Removed reseller name element in <info> response.

* Added acknowledgment.

* Revised the typo "resellerr" to "resellerext".

WG document-00: WG document submitted

WG document-01: Keep document alive for further discussion. The requirement of reseller information is clear for both registrar and registry. What we should reach a consensus is whether the extension should support only a name or ID and name.

Organization WG document-00: Change to a generic organization object extension.

Organization WG document-01: Added "Implementation Status" section.

Organization WG document-02: Accepted some of the feedbacks on the mailing list. Modified the examples in the document.

Organization WG document-03:

* Updated typos.

* Changed some descriptions about <orgext:id> and role attribute.

* Modified the example of "domain with no organization".

* Updated section 8, adding implementation status of Verisign.

Organization WG document-04:

* Updated typos.

* Removed the example of <update> command, domain with no organization.

* Updated references.

* Updated section 8 of implementation status.

Organization WG document-05:

* Removed the minOccurs="0" from the addRemChgType type of the XML schema
* Removed the third paragraph of "Implementation Status".
* Remove the Informative Reference to draft-ietf-regext-reseller-ext from the draft.

Organization WG document-06:

* Updated "Abstraction".
* Added "Query" for "<Transfer> Query Command".
* Change "Registrant Contact" to IESG in section 7.1.
* Modified section 7.2.

Organization WG document-07:

* Updated "Abstraction".

Organization WG document-08:

* Updated error codes of <update> response.
* Modified XML namespace and schema.

Organization WG document-09:

* Modified XML namespace and schema.
* Changed "Exactly one" to "At least one" in section 4.2.5.

Organization WG document-10:

* Updated the reseller id and dns proxy id in the document.
* Updated the full xml namespace.
* Updated the text of EPP <orgext:add>, <orgext:rem> and <orgext:chg>.

* Added "Document satus" and "Reference" in section "EPP Extension Registry".

Organization WG document-11:

* Added the reference of draft-ietf-dnsop-terminology-bis.

## Authors' Addresses

Linlin Zhou
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Email: zhoulinlin@cnnic.cn

Ning Kong
Consultant

Email: ietfing@gmail.com

Junkai Wei
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing, Beijing 100190
China

Email: weijunkai@cnnic.cn

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

Email: yaojk@cnnic.cn

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

Email: jgould@verisign.com

Registration Protocols Extensions
Internet-Draft
Updates: 7484 (if approved)
Intended status: Best Current Practice
Expires: February 4, 2019

S. Hollenbeck
Verisign Labs
A. Newton
ARIN

August 3, 2018

# Registration Data Access Protocol (RDAP) Object Tagging
# draft-ietf-regext-rdap-object-tag-05

## Abstract

The Registration Data Access Protocol (RDAP) includes a method that can be used to identify the authoritative server for processing domain name, IP address, and autonomous system number queries. The method does not describe how to identify the authoritative server for processing other RDAP query types, such as entity queries. This limitation exists because the identifiers associated with these query types are typically unstructured. This document updates RFC 7484 by describing an operational practice that can be used to add structure to RDAP identifiers that makes it possible to identify the authoritative server for additional RDAP queries.

## 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 https://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 February 4, 2019.

## Copyright Notice

Table of Contents



## 1. Introduction

The Registration Data Access Protocol (RDAP) includes a method ([RFC7484]) that can be used to identify the authoritative server for processing domain name, IP address, and autonomous system number (ASN) queries. This method works because each of these data elements is structured in a way that facilitates automated parsing of the element and association of the data element with a particular RDAP service provider. For example, domain names include labels (such as "com", "net", and "org") that are associated with specific service providers.

As noted in Section 9 of RFC 7484 [RFC7484], the method does not describe how to identify the authoritative server for processing entity queries, name server queries, help queries, or queries using certain search patterns. This limitation exists because the identifiers bound to these queries are typically not structured in a way that makes it easy to associate an identifier with a specific service provider. This document describes an operational practice that can be used to add structure to RDAP identifiers that makes it

possible to identify the authoritative server for additional RDAP queries.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Object Naming Practice

Tagging object identifiers with a service provider tag makes it possible to identify the authoritative server for processing an RDAP query using the method described in RFC 7484 [RFC7484]. A service provider tag is constructed by prepending the Unicode HYPHEN-MINUS character "-" (U+002D, described as an "unreserved" character in RFC 3986 [RFC3986]) to an IANA-registered value that represents the service provider. For example, a tag for a service provider identified by the string value "ARIN" is represented as "-ARIN".

In combination with the rdapConformance attribute described in Section 4, service provider tags are concatenated to the end of RDAP query object identifiers to unambiguously identify the authoritative server for processing an RDAP query. Building on the example from Section 3.1.5 of RFC 7482 [RFC7482], an RDAP entity handle can be constructed that allows an RDAP client to bootstrap an entity query. The following identifier is used to find information for the entity associated with handle "XXXX" at service provider "ARIN":

XXXX-ARIN

Clients that wish to bootstrap an entity query can parse this identifier into distinct handle and service provider identifier elements. Handles can themselves contain HYPHEN-MINUS characters; the service provider identifier is found following the last HYPHEN-MINUS character in the tagged identifier. The service provider identifier is used to retrieve a base RDAP URL from an IANA registry. The base URL and entity handle are then used to form a complete RDAP query path segment. For example, if the base RDAP URL "https://example.com/rdap/" is associated with service provider "YYYY" in an IANA registry, an RDAP client will parse a tagged entity identifier "XXXX-YYYY" into distinct handle ("XXXX") and service provider ("YYYY") identifiers. The service provider identifier "YYYY" is used to query an IANA registry to retrieve the base RDAP URL "https://example.com/rdap/". The RDAP query URL is formed using the base RDAP URL and entity path segment described in Section 3.1.5 of RFC 7482 [RFC7482], using "XXXX-YYY" as the value of the handle

identifier. The complete RDAP query URL becomes "https://example.com/rdap/entity/XXXX-YYYY".

Implementation of this practice requires tagging of unstructured potential query identifiers in RDAP responses. Consider these elided examples ("..." is used to note elided response objects) from Section 5.3 of RFC 7483 [RFC7483] in which the handle identifiers have been tagged with service provider tags "RIR", "DNR", and "ABC" respectively:

```
{
  "objectClassName" : "domain",
  "handle" : "XXXX-RIR",
  "ldhName" : "0.2.192.in-addr.arpa",
  "nameservers" :
  [
    ...
  ],
  "secureDNS":
  {
    ...
  },
  "remarks" :
  [
    ...
  ],
  "links" :
  [
    ...
  ],
  "events" :
  [
    ...
  ],
  "entities" :
  [
    {
      "objectClassName" : "entity",
      "handle" : "XXXX-RIR",
      "vcardArray":
      [
        ...
      ],
      "roles" : [ "registrant" ],
      "remarks" :
      [
        ...
      ],
```

```
        "links" :
        [
          ...
        ],
        "events" :
        [
          ...
        ]
      }
    ],
    "network" :
    {
      "objectClassName" : "ip network",
      "handle" : "XXXX-RIR",
      "startAddress" : "192.0.2.0",
      "endAddress" : "192.0.2.255",
      "ipVersion" : "v4",
      "name": "NET-RTR-1",
      "type" : "DIRECT ALLOCATION",
      "country" : "AU",
      "parentHandle" : "YYYY-RIR",
      "status" : [ "active" ]
    }
  }
```

Figure 1

```
  {
    "objectClassName" : "domain",
    "handle" : "XXXX-YYY-DNR",
    "ldhName" : "xn--fo-5ja.example",
    "unicodeName" : "foo.example",
    "variants" :
    [
      ...
    ],
    "status" : [ "locked", "transfer prohibited" ],
    "publicIds":
    [
      ...
    ],
    "nameservers" :
    [
      {
        "objectClassName" : "nameserver",
        "handle" : "XXXX-DNR",
        "ldhName" : "ns1.example.com",
        "status" : [ "active" ],
```

```
        "ipAddresses" :
        {
          ...
        },
        "remarks" :
        [
          ...
        ],
        "links" :
        [
          ...
        ],
        "events" :
        [
          ...
        ]
      },
      {
        "objectClassName" : "nameserver",
        "handle" : "XXXX-DNR",
        "ldhName" : "ns2.example.com",
        "status" : [ "active" ],
        "ipAddresses" :
        {
          ...
        },
        "remarks" :
        [
          ...
        ],
        "links" :
        [
          ...
        ],
        "events" :
        [
          ...
        ]
      }
    ],
    "secureDNS":
    {
      ...
    },
    "remarks" :
    [
      ...
    ],
```

```
      "links" :
      [
        ...
      ],
      "port43" : "whois.example.net",
      "events" :
      [
        ...
      ],
      "entities" :
      [
        {
          "objectClassName" : "entity",
          "handle" : "XXXX-ABC",
          "vcardArray":
          [
            ...
          ],
          "status" : [ "validated", "locked" ],
          "roles" : [ "registrant" ],
          "remarks" :
          [
            ...
          ],
          "links" :
          [
            ...
          ],
          "events" :
          [
            ...
          ]
        }
      ]
   }
```

Figure 2

As described in Section 5 of RFC 7483 [RFC7483], RDAP responses can contain "self" links. Service provider tags and self references SHOULD be consistent. If they are inconsistent, the service provider tag is processed with higher priority when using these values to identify a service provider.

There is a risk of unpredictable processing behavior if the HYPHEN-MINUS character is used for naturally occurring, non-separator purposes in an entity handle. This could lead to a client mistakenly assuming that a HYPHEN-MINUS character represents a separator and the

text that follows HYPHEN-MINUS is a service provider identifier. A client that queries the IANA registry for what they assume is a valid service provider will likely receive an unexpected, invalid result. As a consequence, use of the HYPHEN-MINUS character as a service provider tag separator MUST be noted by adding an rdapConformance value to query responses as described in Section 4.

The HYPHEN-MINUS character was chosen as a separator for two reasons: 1) it is a familiar separator character in operational use, and 2) it avoids collision with URI-reserved characters. The list of unreserved characters specified in Section 2.3 of RFC 3986 [RFC3986] provided multiple options for consideration:

unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"

ALPHA and DIGIT characters were excluded because they are commonly used in entity handles for non-separator purposes. HYPHEN-MINUS is commonly used as a separator and recognition of this practice will reduce implementation requirements and operational risk. The remaining characters were excluded because they are not broadly used as separators in entity handles.

## 3. Bootstrap Service Registry for Provider Object Tags

The bootstrap service registry for the RDAP service provider space is represented using the structure specified in Section 3 of RFC 7484 [RFC7484]. The JSON output of this registry contains contact information for the registered service provider identifiers, alphanumeric identifiers that identify RDAP service providers, and base RDAP service URLs as shown in this example.

```
{
  "version": "1.0",
  "publication": "YYYY-MM-DDTHH:MM:SSZ",
  "description": "RDAP bootstrap file for service provider object tags",
  "services": [
    [
      ["contact@example.com"],
      ["YYYY"],
      [
        "https://example.com/rdap/"
      ]
    ],
    [
      ["contact@example.org"],
      ["ZZ54"],
      [
        "http://rdap.example.org/"
      ]
    ],
    [
      ["contact@example.net"],
      ["1754"],
      [
        "https://example.net/rdap/",
        "http://example.net/rdap/"
      ]
    ]
  ]
 }
```

Figure 3

Alphanumeric service provider identifiers conform to the suffix portion ("\w{1,8}") of the "roidType" syntax specified in Section 4.2 of RFC 5730 [RFC5730].

### 3.1. Registration Procedure

The service provider registry is populated using the "First Come First Served" policy defined in RFC 8126 [RFC8126]. Provider identifier values can be derived and assigned by IANA on request. Registration requests include an email address to be associated with the registered service provider identifier, the requested service provider identifier (or an indication that IANA should assign an identifier), and one or more base RDAP URLs to be associated with the service provider identifier.

## 4. RDAP Conformance

RDAP responses that contain values described in this document MUST indicate conformance with this specification by including an rdapConformance ([RFC7483]) value of "rdap_objectTag_level_0". The information needed to register this value in the RDAP Extensions Registry is described in Section 5.2.

Example rdapConformance structure with extension specified:

```
"rdapConformance" :
[
  "rdap_level_0",
  "rdap_objectTag_level_0"
]
```

Figure 4

## 5. IANA Considerations

IANA is requested to create the RDAP "Bootstrap Service Registry for Provider Object Tags" listed below and make it available as JSON objects. The contents of this registry is described in Section 3, with the formal syntax specified in Section 10 of RFC 7484 [RFC7484].

### 5.1. Bootstrap Service Registry Structure

Entries in this registry contain the following information:

- An email address that identifies a contact associated with the registered RDAP service provider value.
- An alphanumeric value that identifies the RDAP service provider being registered.
- One or more URLs that provide the RDAP service regarding this registration. The URLS are expected to supply the same data, but they can differ in scheme or other components as required by the service operator.

### 5.2. RDAP Extensions Registry

IANA is requested to register the following value in the RDAP Extensions Registry:

Extension identifier: rdap_objectTag
Registry operator: Any
Published specification: This document.
Contact: IESG <iesg@ietf.org>

Intended usage: This extension describes a best practice for structuring entity identifiers to enable query bootstrapping.

## 6. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

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, "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 Labs

Responsible Organization: Verisign Labs
Location: https://rdap.verisignlabs.com/
Description: This implementation includes support for domain registry RDAP queries using live data from the .cc and .tv country code top-level domains. Client authentication is required to receive entity information in query responses.
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Scott Hollenbeck, shollenbeck@verisign.com

### 6.2. OpenRDAP

Responsible Organization: OpenRDAP
Location: https://www.openrdap.org

Description: RDAP client implementing bootstrapping for entity handles with a service provider tag. A test Bootstrap Services Registry file is currently used in lieu of an official one.
Level of Maturity: Alpha
Coverage: Implements draft 04+, supports the HYPHEN-MINUS separator character only.
Contact Information: Tom Harwood, tfh@skip.org

## 7. Security Considerations

This practice uses IANA as a well-known, central trusted authority to allow users to get RDAP data from an authoritative source, reducing the risk of sending queries to non-authoritative sources and divulging query information to unintended parties. Using TLS [RFC5246] to protect the connection to IANA allows the server to authenticate itself as being operated by IANA and provides integrity protection for the resulting referral information, as well as providing privacy protection via data confidentiality. The subsequent RDAP connection is performed as usual, and retains the same security properties of the RDAP protocols themselves.

## 8. Acknowledgements

The author would like to acknowledge the following individuals for their contributions to the development of this document: Tom Harrison, Patrick Mevzek, and Marcos Sanz. In addition, the authors would like to recognize the Regional Internet Registry (RIR) operators (AFRINIC, APNIC, ARIN, LACNIC, and RIPE) that have been implementing and using the practice of tagging handle identifiers for several years. Their experience provided significant inspiration for the development of this document.

## 9. References

### 9.1. Normative References

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

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <https://www.rfc-editor.org/info/rfc7484>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

## 9.2. Informative References

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[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>.

# Appendix A. Change Log

00: Initial version.

01: Changed separator character from HYPHEN MINUS to COMMERCIAL AT. Added a recommendation to maintain consistency between service provider tags and "self" links (suggestion received from Tom Harrison). Fixed a spelling error, and corrected the network

example in Section 2 (editorial erratum reported for RFC 7483 by Marcos Sanz). Added acknowledgements.

02: Changed separator character from COMMERCIAL AT to TILDE. Clarity updates and fixed an example handle. Added text to describe the risk of separator characters appearing naturally in entity handles and being misinterpreted as separator characters.

03: Added Implementation Status section (Section 6).

04: Keepalive refresh.

05: Added OpenRDAP implementation information to Section 6.

00: Initial working group version.

01: Added text to describe why the TILDE character was chosen as the separator character.

02: Nit fixes. Added rdapConformance text, switched back to HYPHEN MINUS, and added IANA registration instructions per working group last call discussion. Updated suffix syntax reference from the IANA EPP ROID registry to RFC 5730 (which is what the IANA registry references).

03: Shepherd writeup review updates to explain examples in Section 2.

04: AD review update to clarify query path construction.

05: IESG review update: object naming practice, revised an example to include multiple separator HYPHEN-MINUS characters, revised security considerations, revised IANA considerations, revised IANA registry description and registration procedure to add email address contact information.

## Authors' Addresses

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

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

Andrew Lee Newton
American Registry for Internet Numbers
PO Box 232290
Centreville, VA 20120
US

Email: andy@arin.net
URI: http://www.arin.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 14, 2019

J. Gould
VeriSign, Inc.

January 10, 2019

# Verification Code Extension for the Extensible Provisioning Protocol (EPP)
## draft-ietf-regext-verificationcode-06

Abstract

This document describes an Extensible Provisioning Protocol (EPP) extension for including a verification code for marking the data for a transform command as being verified by a 3rd party, which is referred to as the Verification Service Provider (VSP). The verification code is digitally signed by the VSP using XML Signature and is "base64" encoded. The XML Signature includes the VSP signer certificate, so the server can verify that the verification code originated from the VSP.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 14, 2019.

Table of Contents

## 1. Introduction

This document describes an extension mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This mapping, an extension to EPP object mappings like the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733], can be used to pass a verification code to one of the EPP transform commands. The domain name object is used for examples in the document. The verification code is signed using XML Signature [W3C.CR-xmldsig-core2-20120124] and is "base64" encoded. The "base64" encoded text of the verification code MUST conform to [RFC2045]. The verification code demonstrates that verification was done by a Verification Service Provider (VSP).

The Verification Service Provider (VSP) is a certified party to verify that data is in compliance with the policies of a locality. A locality MAY require the client to have data verified in accordance with local regulations or laws utilizing data sources not available to the server. The VSP has access to the local data sources and is authorized to verify the data. Examples include verifying that the domain name is not prohibited and verifying that the domain name registrant is a valid individual, organization, or business in the locality. The data verified, and the objects and operations that require the verification code to be passed to the server, is up to the policies of the locality. The verification code represents a marker that the verification was completed. The signer certificate and the digital signature of the verification code MUST be verified by the server.

### 1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

XML 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.

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

## 2. Object Attributes

This extension adds additional elements to EPP object mappings like the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733]. Only those new elements are described here.

### 2.1. Verification Code

The Verification Code is a formatted token, referred to as the Verification Code Token, that is digitally signed by a Verification Service Provider (VSP) using XML Signature [W3C.CR-xmldsig-core2-20120124], using the process described in Section 2.1.1, and is then "base64" encoded, as defined in Section 2.1.2. The Verification Code Token syntax is specified using Augmented Backus-Naur Form (ABNF) grammar [RFC5234] as follows:

Verification Code Token ABNF

```
token     = vsp-id "-" verification-id ; Verification Code Token
vsp-id    = 1*DIGIT                    ; VSP Identifier
verification-id = 1*(DIGIT / ALPHA)    ; Verification Identifier
```

For a VSP given VSP Identifier "1" and with a Verification Identifier of "abc123", the resulting Verification Code Token is "1-abc123". The Verification Identifier MUST be unique within a VSP and the VSP Identifier MUST be unique across supporting VSP's, so the Verification Code Token MUST be unique to an individual verification. The VSP Identifiers MAY require registration within an IANA registry.

#### 2.1.1. Signed Code

The <verificationCode:signedCode> is the fragment of XML that is digitally signed using XML Signature [W3C.CR-xmldsig-core2-20120124]. The <verificationCode:signedCode> element includes a required "id" attribute of type XSD ID for use with an IDREF URI from the Signature element. The certificate of the issuer MUST be included with the Signature so it can be chained with the issuer's certificate by the validating client.

The <verificationCode:signedCode> element includes a REQUIRED "type" attribute for use in defining the type of the signed code. It is up to the VSP and the server to define the valid values for the "type" attribute. Examples of possible "type" attribute values include "domain" for verification of the domain name, "registrant" for verification of the registrant contact, or "domain-registrant" for verification of both the domain name and the registrant. The typed signed code is used to indicate the verifications that are done by the VSP. The "type" attribute values MAY require registration within an IANA registry.

A <verificationCode:signedCode> element substitutes for the <verificationCode:abstractSignedCode> abstract element to define a concrete definition of a signed code. The <verificationCode:abstractSignedCode> element can be replaced by other signed code definitions using the XML schema substitution groups feature.

The child elements of the <verificationCode:signedCode> element include:

<verificationCode:code> Contains the Verification Code Token as defined by the ABNF in Section 2.1.
<Signature> XML Signature [W3C.CR-xmldsig-core2-20120124] for the <verificationCode:signedCode>. Use of a namespace prefix, like "dsig", is recommended for the XML Signature [W3C.CR-xmldsig-core2-20120124] elements.

Example of a "domain" typed signed code using the <verificationCode:signedCode> element and XML Signature [W3C.CR-xmldsig-core2-20120124]:

```
<verificationCode:signedCode
  xmlns:verificationCode=
  "urn:ietf:params:xml:ns:verificationCode-1.0"
  id="signedCode">
  <verificationCode:code type="domain">1-abc111
  </verificationCode:code>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
    <CanonicalizationMethod
 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <SignatureMethod
 Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <Reference URI="#signedCode">
     <Transforms>
      <Transform
 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
```

```
        </Transforms>
        <DigestMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <DigestValue>wgyW3nZPoEfpptlhRILKnOQnbdtU6ArM7ShrAfHgDFg=
    </DigestValue>
       </Reference>
      </SignedInfo>
      <SignatureValue>
    jMu4PfyQGiJBF0GWSEPFCJjmywCEqR2h4LD+ge6XQ+JnmKFFCuCZS/3SLKAx0L1w
    QDFO2e0Y69k2G7/LGE37X3vOflobFM1oGwja8+GMVraoto5xAd4/AF7eHukgAymD
    o9toxoa2h0yV4A4PmXzsU6S86XtCcUE+S/WM72nyn47zoUCzzPKHZBRyeWehVFQ+
    jYRMIAMzM57HHQA+6eaXefRvtPETgUO4aVIVSugc4OUAZZwbYcZrC6wOaQqqqAZi
    30aPOBYbAvHMSmWSS+hFkbshomJfHxb97TD2grlYNrQIzqXk7WbHWy2SYdA+sI/Z
    ipJsXNa6osTUw1CzA7jfwA==
      </SignatureValue>
      <KeyInfo>
       <X509Data>
       <X509Certificate>
    MIIESTCCAzGgAwIBAgIBAjANBgkqhkiG9w0BAQsFADBiMQswCQYDVQQGEwJVUzEL
    MAkGA1UECBMCQ0ExFDASBgNVBAcTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJQ0FO
    TiBUTUNIMRswGQYDVQQDExJJQ0FOTiBUTUNIIFRFU1QgQ0EwHhcNMTMwMjA4MDAw
    MDAwWhcNMTgwMjA3MjM1OTU5WjBsMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0Ex
    FDASBgNVBAcTC0xvcyBBbmdlbGVzMRcwFQYDVQQKEw5WYWxpZGF0b3IgVE1DSDEh
    MB8GA1UEAxMYVmFsaWRhdG9yIFRNQ0ggVEVTVCBDRVJUMIIBIjANBgkqhkiG9w0B
    AQEFAAOCAQ8AMIIBCgKCAQEAo/cwvXhbVYl0RDWWvoyeZpETVZVVcMCovUVNg/sw
    WinuMgEWgVQFrz0xA04pEhXCFVv4evbUpekJ5buqU1gmQyOsCKQlhOHTdPjvkC5u
    pDqa51Flk0TMaMkIQjs7aUKCmA4RG4tTTGK/EjR1ix8/D0gHYVRldy1YPrMP+ou7
    5bOVnIos+HifrAtrIv4qEqwLL4FTZAUpaCa2BmgXfy2CSRQbxD5Or1gcSa3vurh5
    sPMCNxqaXmIXmQipS+DuEBqMM8tldaN7RYojUEKrGVsNk5i9y2/7sjn1zyyUPf7v
    L4GgDYqhJYWV61DnXgx/Jd6CWxvsnDF6scscQzUTEl+hywIDAQABo4H/MIH8MAwG
    A1UdEwEB/wQCMAAwHQYDVR0OBBYEFPZEcIQcD/Bj2IFz/LERuo2ADJviMIGMBgNV
    HSMEgYQwgYGAFO0/7kEh3FuEKS+Q/kYHaD/W6wihoWakZDBiMQswCQYDVQQGEwJV
    UzELMAkGA1UECBMCQ0ExFDASBgNVBAcTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJ
    Q0FOTiBUTUNIMRswGQYDVQQDExJJQ0FOTiBUTUNIIFRFU1QgQ0GCAQEwDgYDVR0P
    AQH/BAQDAgeAMC4GA1UdHwQnMCUwI6AhoB+GHWh0dHA6Ly9jcmwuaWNhbm4ub3Jn
    L3RtY2guY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQB2qSy7ui+43cebKUKwWPrzz9y/
    IkrMeJGKjo40n+9uekaw3DJ5EqiOf/qZ4pjBD++oR6BJCb6NQuQKwnoAz5lE4Ssu
    y5+i93oT3HfyVc4gNMIoHm1PS19l7DBKrbwbzAea/0jKWVzrvmV7TBfjxD3AQo1R
    bU5dBr6IjbdLFlnO5x0G0mrG7x5OUPuurihyiURpFDpwH8KAH1wMcCpXGXFRtGKk
    wydgyVYAty7otkl/z3bZkCVT34gPvF70sR6+QxUy8u0LzF5A/beYaZpxSYG31amL
    AdXitTWFipaIGea9lEGFM0L9+Bg7XzNn4nVLXokyEB3bgS4scG6QznX23FGk
      </X509Certificate>
      </X509Data>
      </KeyInfo>
     </Signature>
   </verificationCode:signedCode>
```

### 2.1.2. Encoded Signed Code

The <verificationCode:encodedSignedCode> element contains one or more encoded form of the digitally signed <verificationCode:signedCode> element, described in Section 2.1.1.

The child elements of the <verificationCode:encodedSignedCode> element include:

<verificationCode:code> One or more <verificationCode:code> elements that is an encoded form of the digitally signed <verificationCode:signedCode> element, described in Section 2.1.1, with the encoding defined by the "encoding" attribute with the default "encoding" value of "base64". The "base64" encoded text of the <verificationCode:code> element MUST conform to [RFC2045].

Example <verificationCode:encodedSignedCode> element that contains one "base64" encoded <verificationCode:signedCode> contained in the <verificationCode:code> element:

```
<verificationCode:encodedSignedCode
  xmlns:verificationCode=
    "urn:ietf:params:xml:ns:verificationCode-1.0">
  <verificationCode:code>
ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYW1zOnht
bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWRD
b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+CiAgICA8U2lnbmF0dXJlTWV0
aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93cudzMub3JnLzIwMDEvMDQveG1sZHNp
Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVk
Q29kZSI+CiAgICAgPFRyYW5zZm9ybXM+CiAgICAgIDxUcmFuc2Zvcm0KIEFsZ29y
aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
ZC1zaWduYXR1cmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CiAgICAgPERpZ2VzdE1l
dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
YyNzaGEyNTYiLz4KIDxEaWdlc3RWYWx1ZT53Z3lXM25aUG9FZnBwdGxoUklMS25P
UW5iZHRVNkFyTTdTaHJBZkhnREZnPTwvRGlnZXN0VmFsdWU+CiAgICA8L1JlZmVy
ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICA8U2lnbmF0dXJlVmFsdWU+CiBqTXU0
UGZ5UUdpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRCtnZTZYUStKbm1LRkZDdUNa
Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzN1gzdk9mbG9iRk0xb0d3
amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUh1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
bVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWhWRlEr
CiBqWVJNSUFNek01N0hIUUErNmVhWGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
```

```
SHhiOTdURDJncmxZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYWx1ZT4KICAgPEtleUluZm8+
CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTUlJRVNUQ0NB
ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVFzRkFEQmlNUXN3Q1FZRFZRUUdF
d0pWVXpFTAogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
bGJHVnpNUk13RVFZRFZRUUtFd3BKUTBGTwogVGlCVVRVTklNUnN3R1FZRFZRUURF
eEpKUTBGT1RpQlVUVU5JSUZSRlUxUWdRMEV3SGhjTk1UTXdNakE0TURBdwogTURB
d1doY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
UVlEVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
c2FXUmhkRzl5SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkJna3Foa2lHOXcw
QgogQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBby9jd3ZYaGJWWWwwUkRXV3ZveWVa
cEVUVlpWVmNNQ292VVZOZy9zdwogV2ludU1nRVdnVlFGcnoweEEwNHBFaFhDRlZ2
NGV2YlVwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
MFRNYU1rSVFqczdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5MVlQ
ck1QK291NwogNWJPVm5Jb3MrSGlmckF0ckl2NHFFcXdMTDRGVFpBVXBhQ2EyQm1n
WGZ5MkNTUlFieEQ1T3IxZ2NTYTN2dXJoNQogc1BNQ054cWFYbUlYbVFpcFMrRHVF
QnFNTTh0bGRhTjdSWW9qVUVLckdWc05rNWk5eTIvN3NqbjF6eXlVUGY3dgogTDRH
Z0RZcWhKWVdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0lEQVFB
Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkVj
SVFjRC9CajJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRWdZUXdnWUdBRk8w
LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdK
VgogVXpFTE1Ba0dBMVVFQ0JNQ1EwRXhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
R1Z6TVJNd0VRWURWUVFLRXdwSgogUTBGT1RpQlVUVU5JTVJzd0dRWURWUVFERXhK
SlEwRk9UaUJVVFVOSUlGUkZVMVFnUTBHQ0FRRXdEZ1lEVlIwUAogQVFIL0JBUURB
Z2VBTUM0R0ExVWRId1FuTUNVd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXTmhi
bTR1YjNKbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
MnFTeTd1aSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
REo1RXFpT2YvcVo0cGpCRCsrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFNzdQogeTUr
aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUzE5bDdEQktyYndiekFlYS8waktXVnpydm1W
N1RCZmp4RDNBUW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
aWh5aVVScEZEcHdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWWFacHhTWUczMWFt
TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCM2Jn
UzRzY0c2UXpuWDIzRkdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8L1g1MDlE
YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPC92ZXJpZmljYXRp
b25Db2RlOnNpZ25lZENvZGU+Cg==
  </verificationCode:code>
</verificationCode:encodedSignedCode>
```

Example <verificationCode:encodedSignedCode> element that contains two <verificationCode:code> elements ;.

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

```
          xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
           <domain:name>domain.example</domain:name>
           <domain:registrant>jd1234</domain:registrant>
           <domain:contact type="admin">sh8013</domain:contact>
           <domain:contact type="tech">sh8013</domain:contact>
           <domain:authInfo>
             <domain:pw>2fooBAR</domain:pw>
           </domain:authInfo>
         </domain:create>
       </create>
       <extension>
         <verificationCode:encodedSignedCode
           xmlns:verificationCode=
             "urn:ietf:params:xml:ns:verificationCode-1.0">
           <verificationCode:code>
   ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
   OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYW1zOnht
   bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWRD
   b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
   ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
   dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
   PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
   My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+CiAgICA8U2lnbmF0dXJlTWV0
   aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNp
   Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVk
   Q29kZSI+CiAgICAgPFRyYW5zZm9ybXM+CiAgICAgIDxUcmFuc2Zvcm0KIEFsZ29y
   aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
   ZC1zaWduYXR1cmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CiAgICAgPERpZ2VzdE1l
   dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
   YyNzaGEyNTYiLz4KIDxEaWdlc3RWYWx1ZT53Z3lXM25aUG9FZnBwdGxoUklMS25P
   UW5iZHRVNkFyTTdTaHJBZkhnREZnPTwvRGlnZXN0VmFsdWU+CiAgICA8L1JlZmVy
   ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICA8U2lnbmF0dXJlVmFsdWU+CiBqTXU0
   UGZ5UUdpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRCtnZTZYUStKbm1LRkZDdUNa
   Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzN1gzdk9mbG9iRk0xb0d3
   amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUh1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
   bVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWhWRlEr
   CiBqWVJNSUFNek01N0hIUUErNmVhWGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
   d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
   SHhiOTdURDJncmxZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
   VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYWx1ZT4KICAgPEtleUluZm8+
   CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTUlJRVNUQ0NB
   ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVFzRkFEQmlNUXN3Q1FZRFZRUUdF
   d0pWVXpFTAogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
   bGJHVnpNUk13RVFZRFZRUUtFd3BKUTBGTwogVGlCVVRVTklNUnN3R1FZRFZRUURF
   eEpKUTBGT1RpQlVUVU5JSUZSRlUxUWdRMEV3SGhjTk1UXdNakE0TURBdwogTURB
   d1doY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
   R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
   UVlEVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
```

```
c2FXUmhkRzl5SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkJna3Foa2lHOXcw
QgogQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBby9jd3ZYaGJWWWwwUkRXV3ZveWVa
cEVUVlpWVmNNQ292VVZOZy9zdwogV2ludU1nRVdnVlFGcnoweEEwNHBFaFhDRlZ2
NGV2YlVwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
MFRNYU1rSVFqczdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5MVlQ
ck1QK291NwogNWJPVm5Jb3MrSGlmckF0ckl2NHFFcXdMTDRGVFpBVXBhQ2EyQm1n
WGZ5MkNTUlFieEQ1T3IxZ2NTYTN2dXJoNQogc1BNQ054cWFYbUlYbVFpcFMrRHVF
QnFNTTh0bGRhTjdSWW9qVUVLckdWc05rNWk5eTIvN3NqbjF6eXlVUGY3dgogTDRH
Z0RZcWhKWVdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0lEQVFB
Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkVj
SVFjRC9CajJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRWdZUXdnWUdBRk8w
LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdK
VgogVXpFTE1Ba0dBMVVFQ0JNQ1EwRXhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
R1Z6TVJNd0VRWURWUVFLRXdwSgogUTBGT1RpQlVUVU5JTVJzd0dRWURWUVFERXhK
SlEwRk9UaUJVVFVOSUlGUkZVMVFnUTBHQ0FRRXdEZ1lEVlIwUAogQVFIL0JBUURB
Z2VBTUM0R0ExVWRId1FuTUNVd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXTmhi
bTR1YjNKbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
MnFTeTd1aSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
REo1RXFpT2YvcVo0cGpCRCsrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFNzdQogeTUr
aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUzE5bDdEQktyYndiekFlYS8waktXVnpydm1W
N1RCZmp4RDNBUW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
aWh5aVVScEZEcHdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWWFacHhTWUczMWFt
TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCM2Jn
UzRzY0c2UXpuWDIzRkdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8L1g1MDlE
YXRhPgogICA8L0tleUluZm8+CiAgPC9ZXJpZmljYXRp
b25Db2RlOnNpZ25lZENvZGU+Cg==
        </verificationCode:code>
        <verificationCode:code>
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48dmVyaWZpY2F0
aW9uQ29kZTpzaWduZWRDb2RlIHhtbG5zOnZlcmlmaWNhdGlvbkNvZGU9InVybjpp
ZXRmOnBhcmFtczp4bWw6bnM6dmVyaWZpY2F0aW9uQ29kZS0xLjAiIGlkPSJzaWdu
ZWRDb2RlIiB0eXBlPSJyZWdpc3RyYW50Ij48dmVyaWZpY2F0aW9uQ29kZTpjb2Rl
PjEtYWJjMjIyPC92ZXJpZmljYXRpb25Db2RlOmNvZGU+PGRzaWc6U2lnbmF0dXJl
IHhtbG5zOmRzaWc9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyMi
Pjxkc2lnOlNpZ25lZEluZm8+PGRzaWc6Q2Fub25pY2FsaXphdGlvbk1ldGhvZCBB
bGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnL1RSLzIwMDEvUkVDLXhtbC1jMTRu
LTIwMDEwMzE1I1dpdGhDb21tZW50cyIvPjxkc2lnOlNpZ25hdHVyZU1ldGhvZCBB
bGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNyc2Et
c2hhMSIvPjxkc2lnOlJlZmVyZW5jZSBVUkk9IiNzaWduZWRDb2RlIj48ZHNpZzpU
cmFuc2Zvcm1zPjxkc2lnOlRyYW5zZm9ybSBBbGdvcml0aG09Imh0dHA6Ly93d3cu
dzMub3JnLzIwMDAvMDkveG1sZHNpZyNlbnZlbG9wZWQtc2lnbmF0dXJlIi8+PC9k
c2lnOlRyYW5zZm9ybXM+PGRzaWc6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0
cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii8+PGRzaWc6RGln
ZXN0VmFsdWU+SFg2TU1WUWdnSStzNG9tT3haYjBGTW1VSlBRdk15WmUybDVEdEhh
QlZMND08L2RzaWc6RGlnZXN0VmFsdWU+PC9kc2lnOlJlZmVyZW5jZT48L2RzaWc6
U2lnbmVkSW5mbz48ZHNpZzpTaWduYXR1cmVWYWx1ZT5VOUhPNVlYVWE0ZUsyYXRz
U1RuQk1DU3dXM0dWUzZnUEtkaDBZTlZicERud1d4b1BtYlR2YkVsNDE4NFlKZ3Uw
```

```
   WXB3RkROMmZLY3JVCk1YV0hncE56K0oycTh6MWpTcVJMUEw0UmpnRWw0eGhiOXl5
   cExOZC8xQXJXRVlhWWZEdUc1S3FYV05MRG5YVzJoQkEzK0R5Wk82MFQKcTVPd0R5
   ZVFSVlNPVWNXVE9FOTJsSlZ4M014Q1V6d1hoL0ZOSTlPbGtXK0ZPNVZNNTZlTmZq
   UEhkUlJVdjdzQzRmM0NnWmFaSWFXNQp2RmJnTmJodFJVa0hsSVhnYVNGWDgvcFdV
   RXFIY0dLTUxnRU1nbHBnQ3RtOFlIcXVqb0tXUk0yUDNiK2h3ZTRsU0hSWVRjK0pB
   eEluClU4RDc1WnliWThnSWFuZUprS2dwVTk2T0tJTGQ5L0l0UVhaeHZnPT08L2Rz
   aWc6U2lnbmF0dXJlVmFsdWU+PGRzaWc6S2V5SW5mbz48ZHNpZzpYNTA5RGF0YT48
   ZHNpZzpYNTA5Q2VydGlmaWNhdGU+TUlJRGlUQ0NBbkdnQXdJQkFnSUVmcXE2SFRB
   TkJna3Foa2lHOXcwQkFRc0ZBREIxTVJBd0RnWURWUVFHRXdkVmJtdHViM2R1TVJB
   dwpEZ1lEVlFRSUV3ZFZibXR1YjNkdU1SQXdEZ1lEVlFRSEV3ZFZibXR1YjNkdU1S
   QXdEZ1lEVlFRS0V3ZFZibXR1YjNkdU1SQXdEZ1lEClZRUUxFd2RWYm10dWIzZHVN
   Umt3RndZRFZRUURFeEIyWlhKcFptbGpZWFJwYjI1RGIyUmxNQjRYRFRFMU1EWXhO
   VEl4TURBeU1sb1gKRFRNMU1EWXhNREl4TURBeU1sb3dkVEVRTUE0R0ExVUVCaE1I
   Vlc1cmJtOTNiakVRTUE0R0ExVUVDQk1IVlc1cmJtOTNiakVRTUE0RwpBMVVFQnhN
   SFZXNXJibTkzYmpFUU1BNEdBMVVFQ2hNSFZXNXJibTkzYmpFUU1BNEdBMVVFQ3hN
   SFZXNXJibTkzYmpFWk1CY0dBMVVFCkF4TVFkbVZ5YVdacFkyRjBhVzl1UTI5a1pU
   Q0NBU0l3RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0NnZ0VCQUpjY2pY
   cmsKUWFJL2lHUEZ3WmVITjFnRFVhcTltVnJmQis2eWR5Qmdoc2FHVFZoaERIOFNO
   TmtpamxIMkxCQ3J3TjhjVjhQZ1BPOXRwbG9rR2F5UwpxNktFaHZtTk03b1dsZk5L
   SkdSdGNidGMzTnJuYzhiUUJacU1xcFo0UlNRTmh5QWh6Ri85UmErd3RFc0JWeGF3
   VDc1L2J0SDZ1YytmClJOdE5FcmhJdVlJUmN0WTZIRmRaR3BlS3cxYnlYK0RsNkJP
   L3ZLdnQ4NDllY1R3aEZIcDUwWGh2NFVTL0Z5aWVLaGs3dDdHRnJGRlQKL2NCTGsy
   WmxFa1lLcFlEU2dlc2lseFg2QkpTZVdCbXZLQzlTL2pBZDhNWmRHVUg2aHNHRXBl
   U1BmZkZQV3FWcXl6V0p5bG91OXF4ZQpnUTZjOFo2SVpXZkUzakxSOUVySDhzOTFD
   Mm1pTFZrQ0F3RUFBYU1oTUI4d0hRWURWUjBPQkJZRUZIY0JLdk03dmk3dUZNTUx5
   ZE43CmVGVXF2YzVVTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFBVjB2cmlrSWRB
   d2l4THZ0NUx5eXpTNFdTU1d0dVlWL2JQMVg3NzVMRmYKSWh3a2xoMENidk5rYXlK
   Tms2Tnp0eDlSc1AwNWZndkxrZER1N0V5cnRzY3I1ZVdETG1WMGtKMWE1N1Z4bnJh
   aEdLTnM2Wit1Ui9pSApMaTJXb3liWEpFT2N0NWtJSjFzL05CeUUrdkdGdjFoTmJz
   dVVVUEVCYWVtaWpYUFROOWxxZE9uM1FIbktobXhsa1czYS9KbmhtT20vCkRWYTE0
   NDJXTVVUSlUyVFlWVldtdUs2NFkwQXFrN2FldzkvVzIzZEcrT2xhOW9VYnBrSXJr
   dDRDN3hRa0d5SXN2eUo3bi91OFhBRDIKbno1T1cvek5GWnlrZDAzT2N3M240NkZx
   c1IwVDlBbFBEWHQxUjlmMjZMd1lxdjk3dWtVNEcrMVRJNHorV0F2TCtVRk9FVnNu
   PC9kc2lnOlg1MDlDZXJ0aWZpY2F0ZT48L2RzaWc6WDUwOURhdGE+PC9kc2lnOktl
   eUluZm8+PC9kc2lnOlNpZ25hdHVyZT48L3ZlcmlmaWNhdGlvbkNvZGU6c2lnbmVk
   Q29kZT4=
         </verificationCode:code>
      </verificationCode:encodedSignedCode>
      </extension>
      <clTRID>ABC-12345</clTRID>
    </command>
   </epp>
```

## 2.2. Verification Profile

A Verification Profile defines the set of verification code types, the commands that the verification code types are required, supported, or not supported, and the grace period by which the

verification code types MUST be set. It is up to server policy what action to take if the verification code type is not set by the grace period. A server MAY support many verification profiles, each with a unique name and a unique verification policy that is implemented by the server. Each client MAY have zero or more server assigned verification profiles that will enforce the required verification policies. Most likely a client will be assigned zero or one server assigned verification profile, but overlapping profiles is possible. Overlapping verification profiles MUST be treated as a logical "and" of the policies by the server. If no verification profile is assigned to the client, no additional verification is required by the client.

## 3. EPP Command Mapping

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

### 3.1. EPP Query Commands

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

#### 3.1.1. EPP <check> Command

This extension does not add any elements to the EPP <check> command or <check> response described in the [RFC5730].

#### 3.1.2. EPP <info> Command

This extension defines additional elements to extend the EPP <info> command of an object mapping like the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733].

The EPP <info> command is used to retrieve the verification information. The verification information is based on the verification profile, as defined in Section 2.2, set in the server for the client. The <verificationCode:info> element is an empty element that indicates that the client requests the verification information. The OPTIONAL "profile" attribute can be used by the client to explicitly specify a verification profile, as defined in Section 2.2, to base the verification information on. It is up to server policy on the set of verification profiles that the client is allowed to explicitly specify, and if the client is not allowed, the server MUST return the 2201 error response.

Example <info> domain command with the <verificationCode:info> extension to retrieve the verification information for the domain "domain.example", using the profiles associated with the client:

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

Example <info> domain command with the <verificationCode:info> extension to retrieve the verification information for the domain "domain.example", using the profiles associated with the client and with the authorization information to retrieve the verification codes from the non-sponsoring client:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <domain:info
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:info>
C:    </info>
C:    <extension>
C:      <verificationCode:info
C:        xmlns:verificationCode=
C:          "urn:ietf:params:xml:ns:verificationCode-1.0"/>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> domain command with the <verificationCode:info> extension to retrieve the verification information for the domain "domain.example", using the the "sample" profile:

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

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

<verificationCode:status> The status of the verification for the object, using all of the verification profiles assigned to the client. There are four possible values for the status:

notApplicable The status is not applicable to the client since there is no assigned verification profile.

nonCompliant The object is non-compliant according to the verification profiles. If at least one of the profiles is "nonCompliant", the object is "nonCompliant".

pendingCompliance The object is not in compliance with the verification profiles, but has a grace period to set the required set of verification codes, as reflected by the due date of the verification code type. If at least one of the profiles is "pendingCompliance" and none of the profiles is "nonCompliant", the object is "pendingCompliance".

compliant The object is compliant with the verification profiles. If All of the profiles for the object are "compliant" or if the object has no assignd profiles, the object is "compliant".

<verificationCode:profile> Zero or more OPTIONAL <verificationCode:profile> elements that defines the verification status of the object based on the profile. The required "name" attribute defines the name of the profile. The <verificationCode:profile> element contains the following child elements:

<verificationCode:status> The status of the verification for the object and the profile. There are four possible values for the status:

notApplicable The profile status is not applicable to the client based on the assigned verification profiles or the profile specified.

nonCompliant The object is non-compliant according to the verification profile.

pendingCompliance The object is not in compliance with the verification profile, but has a grace period to set the required set of verification codes, as reflected by the due date of the verification code type.

compliant The object is compliant with the verification profile.

<verificationCode:missing> OPTIONAL list of missing verification code types. The <verificationCode:missing> element is returned only if there is at least one missing verification code type and based on server policy. The <verificationCode:missing> element contains the following child elements:

<verificationCode:code> One or more <verificationCode:code> elements that is empty with the REQUIRED "type" attribute that indicates the verification code type and the REQUIRED "due" attribute that indicates when the verification code type was or is due. Past due verification code types will result in the <verificationCode:status> element being set to "nonCompliant".

<verificationCode:set> OPTIONAL list of set verification codes. The <verificationCode:set> element is returned only if there is at least one set verification code. The <verificationCode:set> element contains the following child elements:

<verificationCode:code> One or more <verificationCode:code> elements containing the verification code with a REQUIRED "type" attribute that indicates the code type and a REQUIRED "date" attribute that indicates when the verification code was set. The inclusion of the code value is up server policy, so if the server determines that the code value cannot be exposed to a non-sponsoring client, the <verificationCode:code> element MUST be empty.

Example <info> domain response using the <verificationCode:infData> extension for a compliant domain using the "sample" profile, and with the two verification codes, from the sponsoring or authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>compliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>compliant
S:          </verificationCode:status>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
```

```
S:              date="2010-04-03T22:00:00.0Z">1-abc333
S:            </verificationCode:code>
S:            <verificationCode:code type="registrant"
S:              date="2010-04-03T22:00:00.0Z">1-abc444
S:            </verificationCode:code>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData> extension for a compliant domain using the "sample" profile, and with the two verification codes, from the sponsoring or authorized client that also includes codes set for the "sample2" profile:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
```

```
S:        <verificationCode:status>compliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>compliant
S:          </verificationCode:status>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z">1-abc333
S:            </verificationCode:code>
S:            <verificationCode:code type="registrant"
S:              date="2010-04-03T22:00:00.0Z">1-abc444
S:            </verificationCode:code>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:        <verificationCode:profile name="sample2">
S:          <verificationCode:status>notApplicable
S:          </verificationCode:status>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z">2-abc555
S:            </verificationCode:code>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData> extension for a compliant domain using the "sample" profile, and with the two verification code types, from the non-sponsoring client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>compliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>compliant
S:          </verificationCode:status>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z"/>
S:            <verificationCode:code type="registrant"
S:              date="2010-04-03T22:00:00.0Z"/>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData> extension for a non-compliant domain using the "sample" profile, and with the verification code types missing along with their due dates:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="serverHold"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>nonCompliant
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>nonCompliant
S:          </verificationCode:status>
S:          <verificationCode:missing>
S:            <verificationCode:code
S:              type="domain"
S:              due="2010-04-03T22:00:00.0Z"/>
S:            <verificationCode:code
S:              type="registrant"
S:              due="2010-04-08T22:00:00.0Z"/>
S:          </verificationCode:missing>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData>

extension for a pending compliance domain using the "sample" profile, with the verification code type missing along with the due date, and with set verification code:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>pendingCompliance
S:        </verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>pendingCompliance
S:          </verificationCode:status>
S:          <verificationCode:missing>
S:            <verificationCode:code
S:              type="registrant"
S:              due="2010-04-08T22:00:00.0Z"/>
S:          </verificationCode:missing>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"
S:              date="2010-04-03T22:00:00.0Z">1-abc333
S:            </verificationCode:code>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> domain response using the <verificationCode:infData> extension for a client that does not have a verification profile assigned:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>domain.example</domain:name>
S:        <domain:roid>DOMAIN-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2010-04-03T22:00:00.0Z
S:        </domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z
S:        </domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        xmlns:verificationCode=
S:        "urn:ietf:params:xml:ns:verificationCode-1.0">
S:        <verificationCode:status>notApplicable
S:        </verificationCode:status>
S:      </verificationCode:infData>
S:    </extension>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

### 3.1.3. EPP <transfer> Command

This extension does not add any elements to the EPP <transfer> query command or <transfer> response described in the [RFC5730].

### 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

This extension defines additional elements to extend the EPP <create> command of an object mapping like the EPP domain name mapping [RFC5731], EPP host mapping [RFC5732], and EPP contact mapping [RFC5733].

The EPP <create> command provides a transform operation that allows a client to create an object. In addition to the EPP command elements described in an object mapping like [RFC5731], the command MAY contain a child <verificationCode:encodedSignedCode> element, as defined in Section 2.1.2, that identifies the extension namespace for the client to provide proof of verification by a Verification Service Provider (VSP). The server MAY support multiple policies for the passing of the <verificationCode:encodedSignedCode> element based on the client profile, which include:

required The client MUST pass a valid <verificationCode:encodedSignedCode> element containing the required set of verification codes. If a <verificationCode:encodedSignedCode> element is not passed or the required set of verification codes is not included, the server MUST return an EPP error result code of 2306. If an invalid <verificationCode:encodedSignedCode> element is passed, the server MUST return an EPP error result code of 2005.

optional The client MAY pass a valid <verificationCode:encodedSignedCode> element. If an invalid <verificationCode:encodedSignedCode> element is passed, the server MUST return an EPP error result code of 2005.

not supported The client MUST NOT pass a <verificationCode:encodedSignedCode> element. If a <verificationCode:encodedSignedCode> element is passed, the server MUST return an EPP error result code of 2102.

Example <create> command to create a domain object with a verification code:

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

```
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>domain.example</domain:name>
C:        <domain:registrant>jd1234</domain:registrant>
C:        <domain:contact type="admin">sh8013</domain:contact>
C:        <domain:contact type="tech">sh8013</domain:contact>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <extension>
C:      <verificationCode:encodedSignedCode
C:        xmlns:verificationCode=
C:          "urn:ietf:params:xml:ns:verificationCode-1.0">
C:        <verificationCode:code>
C:ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
C:OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYW1zOnht
C:bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWRD
C:b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
C:ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
C:dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
C:PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
C:My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+CiAgICA8U2lnbmF0dXJlTWV0
C:aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNp
C:Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVk
C:Q29kZSI+CiAgICAgPFRyYW5zZm9ybXM+CiAgICAgIDxUcmFuc2Zvcm0KIEFsZ29y
C:aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
C:ZC1zaWduYXR1cmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CiAgICAgPERpZ2VzdE1l
C:dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
C:YyNzaGEyNTYiLz4KIDxEaWdlc3RWYWx1ZT53Z3lXM25aUG9FZnBwdGxoUklMS25P
C:UW5iZHRVNkFyTTdTaHJBZkhnREZnPTwvRGlnZXN0VmFsdWU+CiAgICA8L1JlZmVy
C:ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICA8U2lnbmF0dXJlVmFsdWU+CiBqTXU0
C:UGZ5UUdpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRCtnZTZYUStKbm1LRkZDdUNa
C:Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzN1gzdk9mbG9iRk0xb0d3
C:amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUh1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
C:bVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWhWRlEr
C:CiBqWVJNSUFNek01N0hIUUErNmVhWGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
C:d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
C:SHhiOTdURDJncmxZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
C:VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYWx1ZT4KICAgPEtleUluZm8+
C:CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTUlJRVNUQ0NB
C:ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVFzRkFEQmlNUXN3Q1FZRFZRUUdF
C:d0pWVXpFTAogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
C:bGJHVnpNUk13RVFZRFZRUUtFd3BKUTBGTwogVGlCVVRVTklNUnN3R1FZRFZRUURF
C:eEpKUTBGT1RpQlVUVU5JSUZSRlUxUWdRMEV3SGhjTk1UTXdNakE0TURBdwogTURB
C:d1doY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
```

```
C:R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
C:UVlEVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
C:c2FXUmhkRzl5SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkJna3Foa2lHOXcw
C:QgogQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBby9jd3ZYaGJWWWwwUkRXV3ZveWVa
C:cEVUVlpWVmNNQ292VVZOZy9zdwogV2ludU1nRVdnVlFGcnoweEEwNHBFaFhDRlZ2
C:NGV2YlVwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
C:MFRNYU1rSVFqczdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5MVlQ
C:ck1QK291NwogNWJPVm5Jb3MrSGlmckF0ckl2NHFFcXdMTDRGVFpBVXBhQ2EyQm1n
C:WGZ5MkNTUlFieEQ1T3IxZ2NTYTN2dXJoNQogc1BNQ054cWFYbUlYbVFpcFMrRHVF
C:QnFNTTh0bGRhTjdSWW9qVUVLckdWc05rNWk5eTIvN3NqbjF6eXlVUGY3dgogTDRH
C:Z0RZcWhKWVdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0lEQVFB
C:Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkVj
C:SVFjRC9CajJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRWdZUXdnWUdBRk8w
C:LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdK
C:VgogVXpFTE1Ba0dBMVVFQ0JNQ1EwRXhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
C:R1Z6TVJNd0VRWURWUVFLRXdwSgogUTBGT1RpQlVUVU5JTVJzd0dRWURWUVFERXhK
C:SlEwRk9UaUJVVFVOSUlGUkZVMVFnUTBHQ0FRRXdEZ1lEVlIwUAogQVFIL0JBUURB
C:Z2VBTUM0R0ExVWRId1FuTUNVd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXTmhi
C:bTR1YjNKbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
C:MnFTeTd1aSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
C:REo1RXFpT2YvcVo0cGpCRCsrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFNzdQogeTUr
C:aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUzE5bDdEQktyYndiekFlYS8waktXVnpydm1W
C:N1RCZmp4RDNBUW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
C:aWh5aVVScEZEcHdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
C:L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWWFacHhTWUczMWFt
C:TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCM2Jn
C:UzRzY0c2UXpuWDIzRkdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8L1g1MDlE
C:YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPC92ZXJpZmljYXRp
C:b25Db2RlOnNpZ25lZENvZGU+Cg==
C:        </verificationCode:code>
C:      </verificationCode:encodedSignedCode>
C:    </extension>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

This extension does not add any elements to the EPP <create> response described in the [RFC5730].

### 3.2.2. EPP <delete> Command

This extension defines additional elements to extend the EPP <delete> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

### 3.2.3. EPP <renew> Command

This extension defines additional elements to extend the EPP <renew> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

### 3.2.4. EPP <transfer> Command

This extension defines additional elements to extend the EPP <transfer> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

### 3.2.5. EPP <update> Command

This extension defines additional elements to extend the EPP <update> command and response in the same fashion as defined for the EPP <create> Command (Section 3.2.1).

## 4. Formal Syntax

One schema is presented here that is the EPP Verification Code Extension 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. Verification Code Extension Schema

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

  <annotation>
    <documentation>
      Extensible Provisioning Protocol v1.0
      Verification Code Extension.
    </documentation>
  </annotation>
```

```
<import namespace="http://www.w3.org/2000/09/xmldsig#"
  schemaLocation="xmldsig-core-schema.xsd"/>

<!-- Abstract signed code for substitution -->
<element name="abstractSignedCode"
  type="verificationCode:abstractSignedCodeType"
  abstract="true"/>

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

<!-- Definition of concrete signed code -->
<element name="signedCode"
  type="verificationCode:signedCodeType"
  substitutionGroup="verificationCode:abstractSignedCode"/>

<complexType name="signedCodeType">
  <complexContent>
    <extension base="verificationCode:abstractSignedCodeType">
      <sequence>
        <element name="code"
          type="verificationCode:verificationCodeType"/>
        <element ref="dsig:Signature"/>
      </sequence>
      <attribute name="id" type="ID" use="required"/>
    </extension>
  </complexContent>
</complexType>

<simpleType name="verificationCodeValueType">
  <restriction base="token">
      <pattern value="\d+-[A-Za-z0-9]+"/>
  </restriction>
</simpleType>

<complexType name="verificationCodeType">
  <simpleContent>
    <extension base=
      "verificationCode:verificationCodeValueType">
      <attribute name="type" type="token"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!-- Definition of an encoded signed code -->
<element name="encodedSignedCode"
  type="verificationCode:encodedSignedCodeListType"/>
```

```
<complexType name="encodedSignedCodeListType">
  <sequence>
     <element name="code"
        type="verificationCode:encodedSignedCodeType"
        minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
</complexType>

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

<!-- info command extension elements -->
<element name="info" type="verificationCode:infoType"/>

<complexType name="infoType">
  <simpleContent>
    <extension base="token">
       <attribute name="profile" type="token"/>
    </extension>
  </simpleContent>
</complexType>


<!-- info response extension elements -->
<element name="infData" type="verificationCode:infDataType"/>

<complexType name="infDataType">
  <sequence>
    <element name="status"
      type="verificationCode:statusEnum"/>
    <element name="profile"
      type="verificationCode:profileDataType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="profileDataType">
  <sequence>
    <element name="status"
      type="verificationCode:statusEnum"/>
    <element name="missing"
      type="verificationCode:missingCodes"
```

```
          minOccurs="0"/>
        <element name="set"
          type="verificationCode:codesType"
          minOccurs="0"/>
      </sequence>
      <attribute name="name" type="token"/>
    </complexType>

    <simpleType name="statusEnum">
      <restriction base="token">
        <enumeration value="notApplicable"/>
        <enumeration value="nonCompliant"/>
        <enumeration value="pendingCompliance"/>
        <enumeration value="compliant"/>
      </restriction>
    </simpleType>

    <complexType name="missingVerificationCode">
      <simpleContent>
        <extension base="token">
          <attribute name="type" type="token"
            use="required"/>
          <attribute name="due" type="dateTime"
            use="required"/>
        </extension>
      </simpleContent>
    </complexType>

    <complexType name="missingCodes">
      <sequence>
        <element name="code"
          type="verificationCode:missingVerificationCode"
          minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <complexType name="infoVerificationCodeType">
      <simpleContent>
        <extension base="token">
          <attribute name="type" type="token"
            use="required"/>
          <attribute name="date" type="dateTime"
            use="required"/>
        </extension>
      </simpleContent>
    </complexType>

    <complexType name="codesType">
```

```
      <sequence>
        <element name="code"
          type="verificationCode:infoVerificationCodeType"
          minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

  </schema>
  END
```

## 5. IANA Considerations

### 5.1. XML Namespace

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

Registration request for the verificationCode namespace:

URI: ietf:params:xml:ns:verificationCode-1.0
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the verificationCode XML schema:

URI: ietf:params:xml:ns:verificationCode-1.0
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: "Verification Code Extension for the Extensible Provisioning Protocol (EPP)"

Document status: Standards Track

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

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

TLDs: Any

IPR Disclosure: None

Status: Active

Notes: None

## 6. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 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-ietf-regext-verificationcode.

Level of maturity: Production

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

### 6.2. Net::DRI

Organization: Dot and Co

Name: Net::DRI

Description: Net::DRI implements the client-side of draft-ietf-regext-verificationcode.

Level of maturity: Production

Coverage: All client-side aspects of the protocol are implemented.

Licensing: GNU Lesser General Public License

Contact: netdri@dotandco.com

## 7. Security Considerations

The mapping extension described in this document is based on the security services described by EPP [RFC5730] and protocol layers used by EPP. The security considerations described in these other specifications apply to this specification as well.

XML Signature [W3C.CR-xmldsig-core2-20120124] is used in this extension to verify that the Verification Code originated from a trusted Verification Service Provider (VSP) and that it wasn't tampered with in transit from the VSP to the client to the server. To support multiple VSP keys, the VSP certificate chain MUST be included in the <X509Certificate> elements of the Signed Code (Section 2.1.1) and MUST chain up and be verified by the server against a set of trusted certificates.

It is RECOMMENDED that signed codes do not include white-spaces between the XML elements in order to mitigate risks of invalidating the digital signature when transferring of signed codes between applications takes place.

Use of XML canonicalization SHOULD be used when generating the signed code. SHA256/RSA-SHA256 SHOULD be used for digesting and signing. The size of the RSA key SHOULD be at least 2048 bits.

# 8. References

## 8.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[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>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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>.

[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>.

[W3C.CR-xmldsig-core2-20120124]
Cantor, S., Roessler, T., Eastlake, D., Yiu, K., Reagle, J., Solo, D., Datta, P., and F. Hirsch, "XML Signature Syntax and Processing Version 2.0", World Wide Web Consortium CR CR-xmldsig-core2-20120124, January 2012, <http://www.w3.org/TR/2012/CR-xmldsig-core2-20120124>.

## 8.2. Informative References

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

# Appendix A. Acknowledgements

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

- Gurshabad Grover
- Rick Wilhelm
- John Levine

# Appendix B. Change History

## B.1. Change from 00 to 01

1. Fixed pendingComplaince and complaint to pendingCompliance and compliant in text.
2. Fixed verificaton to verification.

## B.2. Change from 01 to 02

1. Added support for the notApplicable status value.

## B.3. Change from 02 to 03

1. Added regular expression pattern for the format of the verification code token value in the XML schema.

## B.4. Change from 03 to 04

1. Ping update.

### B.5. Change from 04 to REGEXT 00

1. Changed to regext working group draft by changing draft-gould-eppext-verificationcode to draft-ietf-regext-verificationcode.

### B.6. Change from REGEXT 00 to REGEXT 01

1. Ping update.

### B.7. Change from REGEXT 01 to REGEXT 02

1. Ping update.

### B.8. Change from REGEXT 02 to REGEXT 03

1. Moved RFC 7451 to an informational reference based on a check done by the Idnits Tool.
2. Replaced the IANA Registrant Contact to be "IESG".

### B.9. Change from REGEXT 03 to REGEXT 04

1. Added the Implementation Status section.
2. Revised the sentence "The data verified by the VSP MUST be stored by the VSP along with the generated verification code to address any compliance issues." to "The VSP MUST store the proof of verification and the generated verification code; and MAY store the verified data.", and added text to the Security Considerations section associated with storing the verification data, based on feedback from Gurshabad Grover.

### B.10. Change from REGEXT 04 to REGEXT 05

1. Removed the "The Verification Service Provider (VSP) MUST store the verification data in compliance with the applicable privacy laws and regulations." sentence from the Security Considerations, based on feedback from Rick Wilhelm and agreement from Gurshabad Grover.
2. Added the sentence "It is up to server policy what action to take if the verification code type is not set by the grace period." to section 2.2 "Verification Profile", to clarify what happens when the verification code grace period expires. This is based on an issue raised by Gurshabad Grover at the IETF-103 REGEXT meeting.

### B.11. Change from REGEXT 05 to REGEXT 06

1. Removed the "The VSP MUST store the proof of verification and the generated verification code; and MAY store the verified data."

sentence from the Introduction, based on feedback from John Levine.

## Author's Address

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

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

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: August 4, 2019

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it

January 31, 2019

# Registration Data Access Protocol (RDAP) Reverse search capabilities
draft-loffredo-regext-rdap-reverse-search-04

## Abstract

The Registration Data Access Protocol (RDAP) does not include query capabilities to find the list of domains related to a set of entities matching a given search pattern. Even if such capabilities, commonly referred as reverse search, respond to some needs not yet readily fulfilled by the current Whois protocol, they have raised concerns from two perspectives: server processing impact and data privacy. Anyway, the impact of the reverse queries on RDAP servers processing is the same as the standard searches and it can be reduced by implementing policies to deal with large result sets, while data privacy risks can be prevented by RDAP access control functionalities. This document describes RDAP query extensions that allow clients to request a reverse search based on the domains-entities relationship.

## 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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on August 4, 2019.

## Copyright Notice

Table of Contents



## 1. Introduction

Reverse Whois is a service provided by many web applications that allow users to find domain names owned by an individual or a company starting from the owner details, such as name and email. Even if it has been considered useful for some legal purposes (e.g. uncovering trademark infringements, detecting cybercrime cases), its availability as a standardised Whois capability has been objected for two main reasons, which now don't seem to conflict with an RDAP implementation.

The first objection has been caused by the potential risks of privacy violation. However, TLDs community is considering a new generation of Registration Directory Services ([ICANN-RDS1],[ICANN-RDS2]), which provide access to sensitive data under some permissible purposes and according to adequate policies to enforce the requestor accreditation, authentication, authorization, and terms and conditions of data use. It is well known that such security policies are not implemented in Whois ([RFC3912]), while they are in RDAP

([RFC7481]). Therefore, RDAP permits a reverse search implementation complying with privacy protection principles.

Another objection to the implementation of a reverse search capability has been connected with its impact on server processing. Since RDAP supports search queries, the impact of both standard and reverse searches is equivalent and can be mitigated by servers adopting ad hoc strategies. Furthermore, reverse search is almost always performed by specifying an entity role (e.g. registrant, technical contact) and this can contribute to restricting the result set.

Reverse searches, such as finding the list of domain names associated with contacts, nameservers or DNSSEC keys, may be useful to registrars as well. Usually, registries adopt out-of-band mechanisms to provide results to registrars asking for reverse searches on their domains. Possible reasons of such requests are:

- the loss of synchronization between the registrar database and the registry database;

- the need of such data to perform massive EPP ([RFC5730]) updates (e.g. changing the contacts of a set of domains, etc.).

Currently, RDAP does not provide any way for a client to search for the collection of domains associated with an entity ([RFC7482]). A query (lookup or search) on domains can return the array of entities related to a domain with different roles (registrant, registrar, administrative, technical, reseller, etc.), but the reverse operation is not allowed. Only reverse searches to find the collection of domains related to a nameserver (ldhName or ip) can be requested. Since entities can be in relation with all RDAP objects ([RFC7483]), the availability of a reverse search can be common to all RDAP query paths.

The protocol described in this specification aims to extend the RDAP query capabilities to enable reverse search based on the domains-entities relationship (the classic Reverse Whois scenario). The extension is implemented by adding new path segments (i.e. search paths) and using a RESTful web service ([REST]). The service is implemented using the Hypertext Transfer Protocol (HTTP) ([RFC7230]) and the conventions described in RFC 7480 ([RFC7480]).

## 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 [RFC2119].

## 2. RDAP Path Segment Specification

The new search paths are OPTIONAL extensions of path segments defined in RFC 7482 ([RFC7482]). The search paths are:

Syntax: domains?entityHandle=<reverse search pattern>

Syntax: domains?entityFn=<reverse search pattern>

Syntax: domains?entityEmail=<reverse search pattern>

Syntax: domains?entityAddr=<reverse search pattern>

The reverse search pattern is a JSON ([RFC8259]) object including two members: "value" and "role". The "value" member represents the search pattern to be applied to the corresponding entity field and can be a JSON type primitive or object. The "role" member is a string whose possible values are those detailed in Section 10.2.4 of RFC 7483 ([RFC7483]). The former is REQUIRED while the latter is OPTIONAL to allow RDAP servers to provide reverse search capabilities without specifying any role.

The search patterns corresponding to the "value" in the first two cases (Figure 1) are the same as specified in paragraph Section 3.2.3 of RFC 7482 ([RFC7482]).

```
domains?entityHandle={"value":"CID-40*","role":"registrant"}

domains?entityFn={"value":"Bobby*","role":"registrant"}
```

Figure 1: Examples of RDAP queries to find all domains related to a registrant whose handle matches "CID-40*" and whose formatted name matches "Bobby*"

The last two reverse searches are considered by gTLD stakeholders very useful to improve RDS searchability ([ICANN-RDS1], [ICANN-RA]).

Searches for domains by related entity email are specified using this form:

```
domains?entityEmail={"value":"XXXX","role":"ZZZZ"}
```

where XXXX is a search pattern representing an email address as defined in RFC 5322 ([RFC5322]).

Searches for domains by related entity postal address are specified using this form:

domains?entityAddr={"value":YYYY,"role":"ZZZZ"}

where YYYY is a JSON object containing the information described in Section 2.4 of RFC 5733 ([RFC5733]), respectively: "street", "city", "sp", "pc" and "cc" (Figure 2). All the members of the postal address object are OPTIONAL but at least one is REQUIRED. The constraints on the members are implicitly joined by AND.

```
domains?entityAddr={"value":{"cc":"CA","city":"Sydney"},"role":"registrant"}
```

Figure 2: Example of a RDAP query to find all domains related to a registrant whose postal address contains the country code equals to "CA" and the city equals to "Sydney"

## 3. Implementation Considerations

The implementation of the proposed extension is technically feasible. The search paths "handle" and "fn" are used as standard paths to search for entities. With regards to the last two reverse searches, both email and postal address information are usually required by the registries but, while the former is usually mapped onto a DBMS indexed field, the latter is mapped onto a combination of non-indexed fields. As a consequence while the former should not significantly decrease the performance, the latter might have an impact on server processing. Anyway, this impact is evaluated to be the same as other query capabilities already presented in RDAP (e.g. wildcard prefixed search pattern) so the risks to generate huge result sets are the same as those related to other standard searches and can be mitigated by adopting the same policies (e.g. restricting search functionalities, limiting the rate of search requests according to the user profile, truncating and paging the results, returning partial responses).

### 3.1. JSON in URLs

Many web services, including RDAP, rely on the HTTP GET method to take advantage from some of its features:

- GET requests can be cached;
- GET requests remain in the browser history;
- GET requests can be bookmarked.

Sometimes, it happens that such advantages should be combined with the requirement to pass objects and arrays in the query string. JSON is the best candidate as data interchange format, but it contains

some characters that are forbidden from appearing in a URL. Anyway, escaping the invalid characters is not an issue because, on the client side, modern browsers automatically encode URLs and, on the server side, several URL encoding/decoding libraries for all web development programming languages are available. The downside of URL encoding is that it can make a pretty long URL, which, depending on the initial length and the number of invalid characters, might exceed the practical limit of web browsers (i.e. 2,000 characters).

Other solutions to pass a JSON expression in a URL could be:

- converting JSON to Base64 ([RFC4648]), but binary data are unreadable;
- using a JSON variation that complies with URL specifications and maintains readability like Rison ([RISON]), URLON ([URLON]) or JSURL ([JSURL]).

The extensions proposed in this document rely on URL encoding because it is widely supported and the risk to exceed the maximum URL length is considered to be very unlikely in RDAP.

## 4. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

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

### 4.1. IIT-CNR/Registro.it

Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
Location: https://rdap.pubtest.nic.it/
Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD.
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

## 5. Privacy Considerations

The use of the capability described in this document SHOULD be compliant with the rules about privacy protection each RDAP provider is subject to. Sensitive registration data SHOULD be protected and accessible for permissible purposes only. Therefore, it is recommended that RDAP servers provide reverse search only to those requestors who are authorized according to a lawful basis. Some potential users of this capability include registrars searching for their own domains and operators in the exercise of an official authority or performing a specific task in the public interest that is set out in law. Another scenario consists of permitting reverse searches, which take into account only those entities that have previously given the explicit consent for publishing and processing their personal data.

## 6. Security Considerations

Security services required to provide controlled access to the operations specified in this document are described in RFC 7481 ([RFC7481]).

## 7. IANA Considerations

This document has no actions for IANA.

## 8. Acknowledgements

The authors would like to acknowledge Scott Hollenbeck, Francisco Arias, Gustavo Lozano and Eduardo Alvarez for their contribution to this document.

# 9. References

## 9.1. Normative References

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

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <https://www.rfc-editor.org/info/rfc3912>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[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>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

## 9.2. Informative References

[ICANN-RA]
Internet Corporation For Assigned Names and Numbers, "Registry Agreement", July 2017, <https://newgtlds.icann.org/sites/default/files/agreements/agreement-approved-31jul17-en.pdf>.

[ICANN-RDS1]
Internet Corporation For Assigned Names and Numbers, "Final Report from the Expert Working Group on gTLD Directory Services: A Next-Generation Registration Directory Service (RDS)", June 2014, <https://www.icann.org/en/system/files/files/final-report-06jun14-en.pdf>.

[ICANN-RDS2]
Internet Corporation For Assigned Names and Numbers, "Final Issue Report on a Next-Generation gTLD RDS to Replace WHOIS", October 2015, <http://whois.icann.org/sites/default/files/files/final-issue-report-next-generation-rds-07oct15-en.pdf>.

[JSURL] github.com, "JSURL", 2016, <https://github.com/Sage/jsurl>.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>.

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

[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>.

[RISON] github.com, "Rison - Compact Data in URIs", 2017, <https://github.com/Nanonid/rison>.

[URLON] github.com, "URL Object Notation", 2017, <https://github.com/cerebral/urlon>.

## Appendix A. Change Log

00: Initial version.
01: Revised some sentences and references.
02: Added "entityEmail" and "entityAddr" path segments. Removed "entityRole" path segment. Revised "Acknowledgements" section.
03: Added "JSON in URLs" section.
04: Revised some sentences in "Introduction" section. Added "Privacy Considerations" section.

## Authors' Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: mario.loffredo@iit.cnr.it
URI: http://www.iit.cnr.it

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: maurizio.martinelli@iit.cnr.it
URI: http://www.iit.cnr.it

Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: March 25, 2019

M. Loffredo
M. Martinelli
IIT-CNR/Registro.it
S. Hollenbeck
Verisign Labs

September 21, 2018

# Registration Data Access Protocol (RDAP) Query Parameters for Result Sorting and Paging
## draft-loffredo-regext-rdap-sorting-and-paging-05

## Abstract

The Registration Data Access Protocol (RDAP) does not include core functionality for clients to provide sorting and paging parameters for control of large result sets. This omission can lead to unpredictable server processing of queries and client processing of responses. This unpredictability can be greatly reduced if clients can provide servers with their preferences for managing response values. This document describes RDAP query extensions that allow clients to specify their preferences for sorting and paging result sets.

## 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 https://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 March 25, 2019.

## Copyright Notice

Table of Contents



## 1. Introduction

The availability of functionality for result sorting and paging provides benefits to both clients and servers in the implementation of RESTful services [REST]. These benefits include:

- reducing the server response bandwidth requirements;
- improving server response time;
- improving query precision and, consequently, obtaining more reliable results;
- decreasing server query processing load;
- reducing client response processing time.

Approaches to implementing features for result sorting and paging can be grouped into two main categories:

1. Sorting and paging are implemented through the introduction of additional parameters in the query string (i.e. ODATA protocol [OData-Part1]);

2. Information related to the number of results and the specific portion of the result set to be returned, in addition to a set of ready-made links for the result set scrolling, are inserted in the HTTP header of the request/response.

However, there are some drawbacks associated with use of the HTTP header. First, the header properties cannot be set directly from a web browser. Moreover, in an HTTP session, the information on the status (i.e. the session identifier) is usually inserted in the header or in the cookies, while the information on the resource identification or the search type is included in the query string. The second approach is therefore not compliant with the HTTP standard [RFC7230]. As a result, this document describes a specification based on use of query parameters.

Currently the RDAP protocol [RFC7482] defines two query types:

- lookup: the server returns only one object;
- search: the server returns a collection of objects.

While the lookup query does not raise issues in the management of large result sets, the search query can potentially generate a large result set that could be truncated according to the limits of the server. In addition, it is not possible to obtain the total number of the objects found that might be returned in a search query response [RFC7483]. Lastly, there is no way to specify sort criteria to return the most relevant objects at the beginning of the result set. Therefore, the client could traverse the whole result set to find the relevant objects or, due to truncation, could not find them at all.

The protocol described in this specification extends RDAP query capabilities to enable result sorting and paging, by adding new query parameters that can be applied to RDAP search path segments. The service is implemented using the Hypertext Transfer Protocol (HTTP) [RFC7230] and the conventions described in RFC 7480 [RFC7480].

The implementation of these parameters is technically feasible, as operators for counting, sorting and paging rows are currently supported by the major RDBMSs.

### 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 [RFC2119].

## 2. RDAP Query Parameter Specification

The new query parameters are OPTIONAL extensions of path segments defined in RFC 7482 [RFC7482]. They are as follows:

- "count": a boolean value that allows a client to request the total number of objects found (that due to truncation can be different from the number of returned objects);
- "sort": a string value that allows a client to request a specific sort order for the result set;
- "limit" and "offset": numeric values that allow a client to request a specific portion of the entire result set.

Augmented Backus-Naur Form (ABNF) [RFC5234] is used in the following sections to describe the formal syntax of these new parameters.

### 2.1. Sorting and Paging Metadata

According to most advanced principles in REST design, collectively known as HATEOAS (Hypermedia as the Engine of Application State) ([HATEOAS]), a client entering a REST application through an initial URI should use the server-provided links to dynamically discover available actions and access the resources it needs. In this way, the client is not requested to have prior knowledge of the service and, consequently, to hard code the URIs of different resources. This would allow the server to make URI changes as the API evolves without breaking the clients. Definitively, a REST service should be self-descriptive as much as possible.

Therefore, the implementation of the query parameters described in this specification recommends servers to provide additional information in their responses about both the available sorting criteria and the possible pagination. Such information is collected in two new data structures named, respectively, "sorting_metadata" and "paging_metadata".

Obviously, both the new data structures are OPTIONAL because their presence in the response not only depends on the implementation of sorting and paging query capabilities but also on some situations related to the results. For example, it is quite natural to expect

that the "paging_metadata" section will not be present at the last result page when the server implements only the forward pagination.

The "sorting_metadata" structure contains the following fields:

- "currentSort": the value of sort parameter as specified in the query string;
- "availableSorts": an array of objects each one describing an available sorting criterion:
  - "property": the name that can be used by the client to request the sorting criterion;
  - "jsonPath": the JSON Path of the RDAP field corresponding to the property;
  - "default": whether the sorting criterion is applied by default;
  - "links": an array of links as described in RFC 8288 [RFC8288] containing the query string that applies the sorting criterion.

Both "currentSort" and "availableSorts" are OPTIONAL fields of the "sorting_metadata" structure. In particular, the "currentSort" field is provided when the query string contains a valid value for sort parameter, while the "availableSorts" field SHOULD be provided when the sort parameter is missing in the query string or when it is present and the server implements more than a sorting criterion for the RDAP object. At least the "property" field is REQUIRED in each item of the "availableSorts" array while the other fields are RECOMMENDED.

The "paging_metadata" structure contains the following fields:

- "totalCount": a numeric value representing the total number of objects found;
- "pageCount": a numeric value representing the number of objects returned in the current page;
- "offset": a numeric value identifying the start of current page in the result set;
- "nextOffset": a numeric value identifying the start of the next page in the result set or null if the result set has been completely scrolled;
- "links": an array of links as described in RFC 8288 [RFC8288] containing the reference to next page.

Only the "pageCount" field is REQUIRED in the "paging_metadata" structure. The other fields appear when pagination occurs. In this specification, only the forward pagination is dealt because it is considered satisfactory in order to traverse the result set. If a server should also implement backward pagination, an appropriate field (e.g. "prevOffset") identifying the start of the previous page is RECOMMENDED. Finally, the "totalCount" field is provided if the query string contains the count parameter.

FOR DISCUSSION: Should the metadata described in this specification be part of a more general "metadata" section including other contents (e.g rate limits, information about the server, information about the response, metadata information related to other parameters)?

## 2.2. "count" Parameter

Currently the RDAP protocol does not allow a client to determine the total number of the results in a query response when the result set is truncated. This is rather inefficient because the user cannot evaluate the query precision and, at the same time, cannot receive information that could be relevant.

The count parameter provides additional functionality (Figure 1) that allows a client to request information from the server that specifies the total number of elements matching a particular search pattern.

```
https://example.com/rdap/domains?name=*nr.com&count=true
```

Figure 1: Example of RDAP query reporting the count parameter

The ABNF syntax is the following:

```
count = "count" EQ ( trueValue / falseValue )
trueValue = ("true" / "yes" / "1")
falseValue = ("false" / "no" / "0")
EQ = "="
```

A trueValue means that the server MUST provide the total number of the objects in the "totalCount" field of the "paging_metadata" section (Figure 2). A falseValue means that the server MUST NOT provide this number.

```
{
  "rdapConformance": [
        "rdap_level_0",
        "paging_level_0"
  ],
  ...
  "paging_metadata": {
    "totalCount": 73
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 2: Example of RDAP response with "paging_metadata" section containing the "totalCount" field

## 2.3. "sort" Parameter

The RDAP protocol does not provide any capability to specify results sort criteria. A server could implement a default sorting scheme according to the object class, but this feature is not mandatory and might not meet user requirements. Sorting can be addressed by the client, but this solution is rather inefficient. Sorting and paging features provided by the RDAP server could help avoid truncation of relevant results and allow for scrolling the result set using subsequent queries.

The sort parameter allows the client to ask the server to sort the results according to the values of one or more properties and according to the sort direction of each property. The ABNF syntax is the following:

```
sort = "sort" EQ sortItem *( "," sortItem )
sortItem = property-ref [":" ( "a" / "d" ) ]
```

"a" means that the ascending sort MUST be applied, "d" means that the descending sort MUST be applied. If the sort direction is absent, an ascending sort MUST be applied (Figure 3).

In the sort parameter ABNF syntax, property-ref represents a reference to a property of an RDAP object. Such a reference could be expressed by using a JSON Path. The JSON Path in a JSON document [RFC8259] is equivalent to the XPath [W3C.CR-xpath-31-20161213] in a XML document. For example, the JSON Path to select the value of the ASCII name inside an RDAP domain object is "$.ldhName", where $ identifies the root of the document (DOM). Another way to select a value inside a JSON document is the JSON Pointer [RFC6901]. While

JSON Path or JSON Pointer are both standard ways to select any value inside JSON data, neither is particularly easy to use (e.g. "$.events[?(@.eventAction='registration')].eventDate" is the JSON Path expression of the registration date in a RDAP domain object).

Therefore, this specification provides a definition of property-ref in terms of RDAP properties. However, not all the RDAP properties are suitable to be used in sort criteria, such as:

- properties providing service information (e.g. links, notices, remarks, etc.);
- multivalued properties (e.g. status, roles, variants, etc.);
- properties modeling relationships to other objects (e.g. entities).

On the contrary, some properties expressed as values of other properties (e.g. registration date) could be used in such a context.

In the following, a list of the proposed properties for sort criteria is presented. The properties are divided in two groups: object common properties and object specific properties.

- Object common properties. Object common properties are derived from the merge of the "eventAction" and the "eventDate" properties. The following values of the sort parameter are defined:
  - registrationDate
  - reregistrationDate
  - lastChangedDate
  - expirationDate
  - deletionDate
  - reinstantiationDate
  - transferDate
  - lockedDate
  - unlockedDate
- Object specific properties. With regard to the specific properties, some of them are already defined among the query paths. In the following the list of the proposed sorting properties, grouped by objects, is shown:
  - Domain: ldhName
  - Nameserver: ldhName, ipV4, ipV6.
  - Entity: fn, handle, org, email, voice, country, city.

In the following, the correspondence between the sorting properties and the RDAP fields is shown (Table 1):

| Object class | Sorting property | RDAP property | Reference in RFC 7483 | Reference in RFC 6350 |
|---|---|---|---|---|
| Searchable objects | Common properties | eventAction values suffixed by "Date" | 4.5. | |
| Domain | ldhName | ldhName | 5.3. | |
| Nameserver | ldhName | ldhName | 5.2. | |
| | ipV4 | v4 ipAddress | 5.2. | |
| | ipV6 | v6 ipAddress | 5.2. | |
| Entity | handle | handle | 5.1. | |
| | fn | vcard fn | 5.1. | 6.2.1 |
| | org | vcard org | 5.1. | 6.6.4 |
| | voice | vcard tel with type="voice" | 5.1. | 6.4.1 |
| | email | vcard email | 5.1. | 6.4.2 |
| | country | country name in vcard adr | 5.1. | 6.3.1 |
| | city | locality in vcard adr | 5.1. | 6.3.1 |

Table 1: Sorting properties definition

With regard to the definitions in Table 1, some further considerations must be made to disambiguate cases where the RDAP property is multivalued:

- Even if a nameserver can have multiple IPv4 and IPv6 addresses, the most common configuration includes one address for each IP version. Therefore, the assumption of having a single IPv4 and/or IPv6 value for a nameserver cannot be considered too stringent.

- With the exception of handle values, all the sorting properties defined for entity objects can be multivalued according to the definition of vCard as given in RFC6350 [RFC6350]. When more than a value is reported, sorting can be applied to the preferred value identified by the parameter pref="1".

Each RDAP provider MAY define other sorting properties than those shown in this document.

The "jsonPath" field in the "sorting_metadata" section is used to clarify the RDAP field the sorting property refers to. In the following, the mapping between the sorting properties and the JSON Paths of the RDAP fields is shown (Table 2). The JSON Paths are provided according to the Goessner v.0.8.0 specification ([GOESSNER-JSON-PATH]):

| Object class | Sorting property | JSON Path |
|---|---|---|
| Searchable objects | registrationDate | "$.domainSearchResults[*].events[?(@.eventAction=="registration")].eventDate |
| | reregistrationDate | "$.domainSearchResults[*].events[?(@.eventAction=="reregistration")].eventDate |
| | lastChangedDate | "$.domainSearchResults[*].events[?(@.eventAction=="lastChanged")].eventDate |
| | expirationDate | "$.domainSearchResults[*].events[?(@.eventAction=="expiration")].eventDate |
| | deletionDate | "$.domainSearchResults[*].events[?(@.eventAction=="deletion")].eventDate |
| | reinstantiationDate | "$.domainSearchResults[*].events[?(@.eventAction=="reinstantiation")].eventDate |
| | transferDate | "$.domainSearchResults[*].events[?(@.eventAction=="transfer")].eventDate |
| | lockedDate | "$.domainSearchResults[*].events[?(@.eventAction=="locked")].eventDate |
| | unlockedDate | "$.domainSearchResults[*].events[?(@.eventAction=="unlocked")].eventDate |
| Domain | ldhName | $.domainSearchResults[*].ldhName |
| Nameserver | ldhName | $.nameserverSearchResults[*].ldhName |
| | ipV4 | $.nameserverSearchResults[*].ipAddresses.v4[0] |
| | ipV6 | $.nameserverSearchResults[*].ipAddresses.v6[0] |
| Entity | handle | $.entitySearchResults[*].handle |
| | fn | $.entitySearchResults[*].vcardArray[1][?(@[0]="fn")][3] |
| | org | $.entitySearchResults[*].vcardArray[1][?(@[ |

| | | | |
|---|---|---|---|
| | | 0]="org")][3] | |
| | voice | $.entitySearchResults[*].vcardArray[1][?(@[ | |
| | | 0]=="tel" && @[1].type=="voice")][3] | |
| | email | $.entitySearchResults[*].vcardArray[1][?(@[ | |
| | | 0]=="email")][3] | |
| | country | $.entitySearchResults[*].vcardArray[1][?(@[ | |
| | | 0]=="adr")][3][6] | |
| | city | $.entitySearchResults[*].vcardArray[1][?(@[ | |
| | | 0]=="adr")][3][3] | |

Table 2: Sorting properties - JSON Path Mapping

If the sort parameter reports an allowed sorting property, it MUST be provided in the "currentSort" field of the "sorting_metadata" structure.

```
https://example.com/rdap/domains?name=*nr.com&sort=ldhName

https://example.com/rdap/domains?name=*nr.com&sort=registrationDate:d

https://example.com/rdap/domains?name=*nr.com&sort=lockedDate,ldhName
```

Figure 3: Examples of RDAP query reporting the sort parameter

### 2.3.1. Representing Sorting Links

An RDAP server MAY use the "links" array of the "sorting_metadata" section to provide ready-made references [RFC8288] to the available sort criteria (Figure 4). Each link represents a reference to an alternate view of the results.

```
{
  "rdapConformance": [
    "rdap_level_0",
    "sorting_level_0"
  ],
  ...
  "sorting_metadata": {
     "currentSort": "ldhName",
     "availableSorts": [
     {
     "property": "registrationDate",
     "jsonPath": "$.domainSearchResults[*].events[?(@.eventAction==\"registratio
n\")].eventDate",
     "default": false,
     "links": [
        {
        "value": "https://example.com/rdap/domains?name=*nr.com
                  &sort=ldhName",
        "rel": "alternate",
        "href": "https://example.com/rdap/domains?name=*nr.com
                 &sort=registrationDate",
        "title": "Result Ascending Sort Link",
        "type": "application/rdap+json"
        },
        {
        "value": "https://example.com/rdap/domains?name=*nr.com
                  &sort=ldhName",
        "rel": "alternate",
        "href": "https://example.com/rdap/domains?name=*nr.com
                 &sort=registrationDate:d",
        "title": "Result Descending Sort Link",
        "type": "application/rdap+json"
        }
     ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 4: Example of a "sorting_metadata" instance to implement result sorting

## 2.4. "limit" and "offset" Parameters

An RDAP query could return a response with hundreds of objects, especially when partial matching is used. For that reason, two parameters addressing result pagination are defined to make responses easier to handle:

- "limit": means that the server MUST return the first N objects of the result set in the response;
- "offset": means that the server MUST skip the first N objects and MUST return objects starting from position N+1.

The ABNF syntax is the following:

```
EQ = "="
limit = "limit" EQ positive-number
offset = "offset" EQ positive-number
positive-number = non-zero-digit *digit
non-zero-digit = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" /
"9"
digit = "0" / non-zero-digit
```

When limit and offset are used together, they allow implementation of result pagination. The following examples illustrate requests to return, respectively, the first 5 objects, the set of objects starting from position 6, and first 5 objects starting from position 11 of the result set (Figure 5).

```
https://example.com/rdap/domains?name=*nr.com&limit=5

https://example.com/rdap/domains?name=*nr.com&offset=5

https://example.com/rdap/domains?name=*nr.com&limit=5&offset=10
```

Figure 5: Examples of RDAP query reporting the limit and offset parameters

### 2.4.1. Representing Paging Links

An RDAP server MAY use the "links" array of the "paging_metadata" section to provide a ready-made reference [RFC8288] to the next page of the result set (Figure 6). Examples of additional "rel" values are "first", "last", "prev".

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
      "search results for domains are limited to 10"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageCount": 10,
    "offset": 10,
    "nextOffset": 20,
    "links": [
      {
      "value": "https://example.com/rdap/domains?name=*nr.com",
      "rel": "next",
      "href": "https://example.com/rdap/domains?name=*nr.com&limit=10
               &offset=10",
      "title": "Result Pagination Link",
      "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 6: Example of a "paging_metadata" instance to implement result pagination based on offset and limit

## 3. Negative Answers

The value constraints for the parameters are defined by their ABNF syntax. Therefore, each request providing an invalid value for a parameter SHOULD obtain an HTTP 400 (Bad Request) response code. The same response SHOULD be returned if the client provides an unsupported value for the sort parameter in both single and multi sort.

The server can provide a different response when it supports the limit and/or offset parameters and the client submits values that are out of the valid ranges. The possible cases are:

- If the client submits a value for the limit parameter that is greater than the number of objects to be processed, it is RECOMMENDED that server returns a response including only the processed objects.

- If the client submits a value for the offset parameter that is greater than the number of objects to be processed, it is RECOMMENDED that server returns an HTTP 404 (Not Found) response code.

Optionally, the response MAY include additional information regarding the negative answer in the HTTP entity body.

## 4. RDAP Conformance

Servers returning the "paging_metadata" section in their responses MUST include "paging_level_0" in the rdapConformance array as well as servers returning the "sorting_metadata" section MUST include "sorting_level_0".

## 5. Implementation Considerations

The implementation of the new parameters is technically feasible, as operators for counting, sorting and paging are currently supported by the major RDBMSs.

In the following, the match between the new defined parameters and the SQL operators is shown (Table 3):

| New query parameter | SQL operator |
|---|---|
| count | count(*) query without offset, limit and order by [MYSQL-COUNT],[POSTGRES-COUNT],[ORACLE-COUNT] |
| sort | order by [MYSQL-SORT],[POSTGRES-SORT],[ORACLE-SORT] |
| limit | limit n (in MySql [MYSQL-LIMIT] and Postgres [POSTGRES-LIMIT]) FETCH FIRST n ROWS ONLY (in Oracle [ORACLE-LIMIT]) |
| offset | offset m (in Postgres) OFFSET m ROWS (in Oracle) |
| limit + offset | limit n offset m (in MySql and Postgres) OFFSET m ROWS FETCH NEXT n ROWS ONLY (in Oracle) |

Table 3: New query parameters vs. SQL operators

With regard to Oracle, Table 3 reports only one of the three methods that can be used to implement limit and offset parameters. The others are described in [ORACLE-ROWNUM] and [ORACLE-ROW-NUMBER].

In addition, similar operators are completely or partially supported by the most known NoSQL databases (MongoDB, CouchDB, HBase, Cassandra, Hadoop) so the implementation of the new parameters seems to be practicable by servers working without the use of an RDBMS.

## 5.1. Considerations about Paging Implementation

The use of limit and offset operators represents the most common way to implement results pagination. However, when offset has a high value, scrolling the result set could take some time. In addition, offset pagination may return inconsistent pages when data are frequently updated (i.e. real-time data) but this is not the case of registration data. An alternative approach to offset pagination is the keyset pagination, a.k.a. seek-method [SEEK] or cursor based pagination. This method has been taken as the basis for the implementation of a cursor parameter [CURSOR] by some REST API providers (e.g. [CURSOR-API1],[CURSOR-API2]). The cursor parameter is an opaque URL-safe string representing a logical pointer to the first result of the next page (Figure 7).

```
{
  "rdapConformance": [
    "rdap_level_0",
    "paging_level_0"
  ],
  ...
  "notices": [
    {
      "title": "Search query limits",
      "type": "result set truncated due to excessive load",
      "description": [
      "search results for domains are limited to 10"
      ]
    }
  ],
  "paging_metadata": {
    "totalCount": 73,
    "pageCount": 10,
    "links": [
      {
      "value": "https://example.com/rdap/domains?name=*nr.com",
      "rel": "next",
      "href": "https://example.com/rdap/domains?name=*nr.com&limit=10
              &cursor=wJlCDLIl6KTWypN7T6vc6nWEmEYe99Hjf1XY1xmqV-M=",
      "title": "Result Pagination Link",
      "type": "application/rdap+json"
      }
    ]
  },
  "domainSearchResults": [
    ...
  ]
}
```

Figure 7: Example of a "paging_metadata" instance to implement keyset pagination

But keyset pagination raises some drawbacks with respect to offset pagination:

- it needs at least one key field;
- it does not allow to sort by any field and paginate the results because sorting has to be made on the key field;
- it does not allow to skip pages because they have to be scrolled in sequential order starting from the initial page;
- it makes very hard the navigation of the result set in both directions because all comparison and sort operations have to be reversed.

Furthermore, in the RDAP context, some additional considerations can be made:

- an RDAP object is a conceptual aggregation of information collected from more than one data structure (e.g. table) and this makes even harder for the developers the implementation of the seek-method that is already quite difficult. In fact, for example, the entity object can gather information from different data structures (registrars, registrants, contacts, resellers, and so on), each one with its own key field mapping the RDAP entity handle;

- depending on the number of the page results as well as the number and the complexity of the properties of each RDAP object in the response, the time required by offset pagination to skip the previous pages could be much faster than the processing time needed to build the current page. In fact, RDAP objects are usually formed by information belonging to multiple data structures and containing multivalued properties (e.g. arrays) and, therefore, data selection is a time consuming process. This situation occurs even though the data selection process makes use of indexes;

- depending on the access levels defined by each RDAP operator, the increase of complexity and the decrease of flexibility of keyset pagination with respect to the offset pagination could be considered impractical.

Finally, the keyset pagination is not fully compliant with the additional RDAP capabilities proposed by this document. In fact, the presence of a possible cursor parameter does not seem to be consistent with both the sorting capability and the possibility to implement additional ready-made links besides the classic "next page" link. But, while the provisioning of more paging links can be superfluous, dropping the sorting capability seems quite unreasonable.

If pagination is implemented by using a cursor, both "offset" and "nextOffset" fields MUST not be included in the "paging_metadata" section.

FOR DISCUSSION: Should RDAP specification reports both offset and cursor parameters and let operators to implement pagination according to their needs, the user access levels, the submitted queries?

## 6. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

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, "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. IIT-CNR/Registro.it

Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
Location: https://rdap.pubtest.nic.it/
Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD. The RDAP server does not implement any security policy because data returned by this server are only for experimental testing purposes. The RDAP server implements both offset and cursor based pagination (the latter only when sort and offset parameters are not present in the query string).
Level of Maturity: This is a "proof of concept" research implementation.
Coverage: This implementation includes all of the features described in this specification.
Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

### 6.2. Google Registry

Responsible Organization: Google Registry
Location: https://www.registry.google/rdap/

Description: This implementation includes support for RDAP queries for TLDs such as .GOOGLE, .HOW, .SOY, and .xn--q9jyb4c . The RDAP server implements cursor based pagination (the number of objects per page is fixed so the limit parameter is not available). The link used to request the next page is included in the notice section of the response.
Level of Maturity: Production.
Coverage: This implementation includes the cursor parameter described in this specification.
Contact Information: Brian Mountford, mountford@google.com

## 7. Security Considerations

Security services for the operations specified in this document are described in RFC 7481 [RFC7481].

Search query typically requires more server resources (such as memory, CPU cycles, and network bandwidth) when compared to lookup query. This increases the risk of server resource exhaustion and subsequent denial of service due to abuse. This risk can be mitigated by either restricting search functionality and limiting the rate of search requests. Servers can also reduce their load by truncating the results in the response. However, this last security policy can result in a higher inefficiency if the RDAP server does not provide any functionality to return the truncated results.

The new parameters presented in this document provide the RDAP operators with a way to implement a secure server without penalizing its efficiency. The "count" parameter gives the user a measure to evaluate the query precision and, at the same time, return a significant information. The sort parameter allows the user to obtain the most relevant information at the beginning of the result set. In both cases, the user doesn’t need to submit further unnecessary search requests. Finally, the limit and offset parameters enable the user to scroll the result set by submitting a sequence of sustainable queries according to the server limits.

## 8. IANA Considerations

This document has no actions for IANA.

## 9. Acknowledgements

The authors would like to acknowledge Brian Mountford for his contribution to the development of this document.

# 10. References

## 10.1. Normative References

[ISO.3166.1988]
International Organization for Standardization, "Codes for the representation of names of countries, 3rd edition", ISO Standard 3166, August 1988.

[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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

## 10.2. Informative References

[CURSOR] Nimesh, R., "Paginating Real-Time Data with Keyset Pagination", July 2014, <https://www.sitepoint.com/paginating-real-time-data-cursor-based-pagination/>.

[CURSOR-API1] facebook.com, "facebook for developers - Using the Graph API", July 2017, <https://developers.facebook.com/docs/graph-api/using-graph-api>.

[CURSOR-API2] twitter.com, "Pagination", 2017, <https://developer.twitter.com/en/docs/ads/general/guides/pagination.html>.

[GOESSNER-JSON-PATH] Goessner, S., "JSONPath - XPath for JSON", 2007, <http://goessner.net/articles/JsonPath/>.

[HATEOAS] Jedrzejewski, B., "HATEOAS - a simple explanation", 2018, <https://www.e4developer.com/2018/02/16/hateoas-simple-explanation/>.

[MYSQL-COUNT] mysql.com, "MySQL 5.7 Reference Manual, Counting Rows", October 2015, <https://dev.mysql.com/doc/refman/5.7/en/counting-rows.html>.

[MYSQL-LIMIT] mysql.com, "MySQL 5.7 Reference Manual, SELECT Syntax", October 2015, <https://dev.mysql.com/doc/refman/5.7/en/select.html>.

[MYSQL-SORT]
mysql.com, "MySQL 5.7 Reference Manual, Sorting Rows", October 2015, <https://dev.mysql.com/doc/refman/5.7/en/sorting-rows.html>.

[OData-Part1]
Pizzo, M., Handl, R., and M. Zurmuehl, "OData Version 4.0. Part 1: Protocol Plus Errata 03", June 2016, <http://docs.oasis-open.org/odata/odata/v4.0/errata03/os/complete/part1-protocol/odata-v4.0-errata03-os-part1-protocol-complete.pdf>.

[ORACLE-COUNT]
Oracle Corporation, "Database SQL Language Reference, COUNT", March 2016, <http://docs.oracle.com/database/122/SQLRF/COUNT.htm>.

[ORACLE-LIMIT]
Oracle Corporation, "Database SQL Language Reference, SELECT, Row limiting clause", March 2016, <http://docs.oracle.com/database/122/SQLRF/SELECT.htm>.

[ORACLE-ROW-NUMBER]
Oracle Corporation, "Database SQL Language Reference, SELECT, ROW_NUMBER", March 2016, <http://docs.oracle.com/database/122/SQLRF/ROW_NUMBER.htm#SQLRF06100>.

[ORACLE-ROWNUM]
Oracle Corporation, "Database SQL Language Reference, SELECT, ROWNUM Pseudocolumn", March 2016, <http://docs.oracle.com/database/122/SQLRF/ROWNUM-Pseudocolumn.htm#SQLRF00255>.

[ORACLE-SORT]
Oracle Corporation, "Database SQL Language Reference, SELECT, Order by clause", March 2016, <http://docs.oracle.com/database/122/SQLRF/SELECT.htm>.

[POSTGRES-COUNT]
postgresql.org, "PostgresSQL, Aggregate Functions", September 2016, <https://www.postgresql.org/docs/9.6/static/functions-aggregate.html>.

[POSTGRES-LIMIT]
postgresql.org, "PostgresSQL, LIMIT and OFFSET", September 2016, <https://www.postgresql.org/docs/9.6/static/queries-limit.html>.

[POSTGRES-SORT]
postgresql.org, "PostgresSQL, Sorting Rows", September 2016, <https://www.postgresql.org/docs/9.6/static/queries-order.html>.

[REST] Fredrich, T., "RESTful Service Best Practices, Recommendations for Creating Web Services", April 2012, <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

[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>.

[SEEK] EverSQL.com, "Faster Pagination in Mysql - Why Order By With Limit and Offset is Slow?", July 2017, <https://www.eversql.com/faster-pagination-in-mysql-why-order-by-with-limit-and-offset-is-slow/>.

[W3C.CR-xpath-31-20161213]
Robie, J., Dyck, M., and J. Spiegel, "XML Path Language (XPath) 3.1", World Wide Web Consortium CR CR-xpath-31-20161213, December 2016, <https://www.w3.org/TR/2016/CR-xpath-31-20161213>.

## Appendix A. Change Log

00: Initial version.
01: Added the paragraph "Considerations about Paging Implementation" to "Implementation Considerations" section. Added "Implementation Status" section. Added acknowledgements. Renamed the property reporting the paging links.
02: Corrected the value of "title" field in "paging_links" property. Updated references to RFC5988 (obsoleted by RFC 8288) and RFC7159 (obsoleted by RFC 8259). Revised some sentences.
03: Added the paragraph "Google Registry" to "Implementation Status" section.

04: Rearranged the information about pagination included in RDAP responses. Added the section "Paging Metadata". Replaced the wrong reference to RFC 5266 with the correct reference to RFC 5226.
05: Renamed "sortby" parameter in "sort". Removed "country" from the list of sorting properties. Added "sorting_level_0" into the "rdapConformance" array. Changed the title of section "Paging Metadata" in "Sorting and Paging Metadata". Changed the "IANA Considerations" section. Added "Representing Sorting Links" section. Changed the name of some sorting properties to be compliant with EPP.

## Authors' Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: mario.loffredo@iit.cnr.it
URI: http://www.iit.cnr.it

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
Pisa 56124
IT

Email: maurizio.martinelli@iit.cnr.it
URI: http://www.iit.cnr.it

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

Email: shollenbeck@verisign.com
URI: https://www.verisignlabs.com/

Internet Engineering Task Force
Internet-Draft
Intended status: Best Current Practice
Expires: May 24, 2019

N. McPherson
1&1 IONOS SE
T. Sattler, Editor

November 25, 2018

# Registry Reporting Repository
## draft-mcpherson-sattler-registry-reporting-repo-06

Abstract

This document describes a domain name registry reporting repository used to provide reports to accredited domain name registrars.

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on May 24, 2019.

Table of Contents



## 1. Introduction

Modern top-level domain registries provide a number of detailed reports and documents that their registrars require on a daily, weekly and monthly basis. These most commonly include transaction reports, as well as lists containing currently unavailable domains and current premium domains. These reports are critical for registrars' businesses and play an important role in accounting and operations processes as well as in sales and marketing activities. In the current set-up registrars must download these reports from each registry's intranet in a different manner according to each registry's own document management set up.

This document describes a domain registry reporting repository used to provide reports to accredited domain registrars.

## 2. Terminology and Definitions

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

### 2.1. Internationalized Domain Names

Top-level domains and domain names contained in a directory or file name MUST be written as A-LABEL according to [RFC5890].

### 2.2. Dates and Times

All dates and times attribute values MUST be expressed in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in ISO 8601 [RFC3339] MUST be used to represent date-time values.

One day is defined as one day in UTC+0. Therefore, months and years will also be calculated on this basis.

## 3. SFTP Server

Each domain name registry sets up and manages a SFTP (https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol) server. Every SFTP server MUST be reachable through a generic URI, such as sftp://registry.example, and MUST listen on port 22.

IP whitelisting to reach this SFTP server is RECOMMENDED.

## 4. SFTP Account

Each domain name registry MUST create one SFTP account for every accredited domain name registrar. If a registrar owns more than one registrar, then a registry SHOULD combine them in one account on request by the parent registrar or entity.

The authentication for a SFTP account should be done with an username and key instead of a password.

To avoid security risks, it is strongly RECOMMENDED to limit SFTP accounts to SFTP only and to exclude them from SSH functions, such as port forwarding, tunneling, SSH login or command execution.

## 5. SFTP Directory Structure

The home directory of a SFTP user MUST be its root. This can be achieved with e.g. chroot (https://en.wikipedia.org/wiki/Chroot) and prevents that a SFTP user can access other directories that are not owned by themselves. All directories MUST be lowercase.

Files in these directories MUST be stored in an appropriate subdirectory according to their creation date.

```
/YYYY-MM/example.csv.gz
/YYYY-MM/domains/example.csv.gz
/YYYY-MM/foo/bar/example.csv.gz
```

YYYY represents the year in which the file was created, format according to ISO 8601 [RFC3339]

MM represents the months in which the file was created, format according to ISO 8601 [RFC3339]

## 6. SFTP Checksum

All files stored on the SFTP server MUST have a sha256sum (https://en.wikipedia.org/wiki/Sha1sum) checksum.

The file name with the checksum MUST be the same as the file name with the content extended with the suffix .sha256.

File with content: example.csv.gz
File with sha256sum: example.csv.gz.sha256

## 7. SFTP Server Maintenance

Maintenance is important and necessary, especially to keep the SFTP server up to date and secure. It MUST be announced in advance. In this case the EPP registry maintenance specification [I-D.sattler-epp-registry-maintenance] is RECOMMENDED to use. The notification notice MUST be send at least 7 days in advance.

## 8. IANA Considerations

This document has no IANA actions.

## 9. Security Considerations

The registry reporting repository described in this document do not provide any security services beyond those described by SFTP. The security considerations described in these other specifications apply to this specification as well.

## 10. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

Add implementation details once available.

# 11. References

## 11.1. Normative References

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

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

## 11.2. Informative References

[I-D.sattler-epp-registry-maintenance] Sattler, T., Roger, C. and Kolker, J., "Registry Maintenance Notifications for the Extensible Provisioning Protocol (EPP)", <https://tools.ietf.org/html/draft-sattler-epp-registry-maintenance> (work in progress), July 2018

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

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

# Appendix A. Change History

## A.1. Change from 00 to 01

Added reference to IDN A-LABEL. Added clarification on date delimitation. Added checksum requirement.

## A.2. Change from 01 to 02

Updated Neal’s author details.

## A.3. Change from 02 to 03

Fixed formatting of the table of contents.

### A.4. Change from 03 to 04

Added editor flag to author.

### A.4. Change from 04 to 05

Minor formatting changes. Added security advice to SFTP.

### A.5. Change from 05 to 06

Changed examples to reflect compressed files.

## Appendix B. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions (sorted alphabetically by company):

- Anders Henke, 1&1 IONOS
- Thomas Keller, 1&1 IONOS
- James Galvin, Afilias
- Andreas Huber, united-domains

## Authors' Addresses

Neal McPherson
1&1 IONOS SE
Ernst-Frey-Str. 5
76135 Karlsruhe
DE

Email: neal.mcpherson@ionos.com
URI: https://www.ionos.com

Tobias Sattler

Email: tobias.sattler@me.com
URI: https://tobiassattler.com