Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2019

R. Carney
J. Snitker
GoDaddy Inc.
October 11, 2018

Validate Mapping for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-validate-04

Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for the validation of contact and eligibility data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes a Validate mapping for version 1.0 of the Extensible Provisioning Protocol (EPP) [RFC5730]. This EPP mapping specifies a flexible schema by which EPP clients and servers can reliably validate contact and eligibility data.

With the increased number of restrictions on contacts and required data points (license, ids, etc.) to register a domain name, a way to validate the data points prior to issuing a transform command is becoming more important.

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.

"validate" is used as an abbreviation for "urn:ietf:params:xml:ns:validate-0.2". The XML namespace prefix "validate" 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.

(Note to RFC Editor: remove the following paragraph before publication as an RFC.)

The XML namespace prefix above contains a version number, specifically "0.2". This version number will increment with successive versions of this document, and will reach 1.0 if and when this document is published as an RFC. This permits clients to distinguish which version of the extension a server has implemented.

2. Object Attributes

A EPP validation object has attributes and associated values that can be viewed by the client. This section describes each attribute type in detail.

2.1. Key Value

Key Value provides a flexible mechanism to share data between the client and the server. The <validate:kv> element defines the data, with two required simple attributes, key and value, and an optional contactType attribute for specificity in the response, more details below.

o An example <validate:kv key="VATID" value="0123456789"/>.

o An example <validate:kv contactType="Admin" key="contact:cc" value="Invalid country code for admin contact, must be MX."/>.

2.2. Validate Id

The <validate:id> element is used in two different scenarios.

First if the <validate:id> is passed by itself with no other elements (e.g. >validate:postalInfo>) then the client intent is that this is an already existing contact and the server should handle the request by looking up the associated data in its system and using that data to validate against.

Second scenario would be if the request includes additional elements then the server should treat the <validate:id> as a temporary contact handle and should not perform a look on the contact but use the data that is passed in the request to validate against.

2.3. Validate PostalInfo, Voice, Fax, Email, AuthInfo

These elements are intended to mirror the definitions in [RFC5733].

3. EPP Command Mapping

A detailed description of the EPP syntax and semantics can be found in [RFC5730]. The command mappings described here are specifically for the Validate Extension.

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

The EPP <check> command is used to validate a list of contact information. The <check> command MUST contain a <validate:check> element that identifies the validate namespace. The <validate:check> element contains the following child elements:

o one or more <validate:contact> element(s) for each contact that is to be validated that contains the contact type of the contact to be validated.

The <validate:contact> element has two required attributes: contactType, which describes the role (registrant, admin, tech,

billing, etc.) of the contact that the contact should be validated for; and tld, which provides the top level domain to be validated against. The <validate:contact> element contains the following child elements:

- o one <validate:id> element (as described in section 2.2).
- o an OPTIONAL <validate:postalInfo> element (as described in section 2.3).
- o an OPTIONAL <validate:voice> element (as described in section 2.3).
- o an OPTIONAL <validate:fax> element (as described in section 2.3).
- o an OPTIONAL <validate:email> element (as described in section 2.3).
- o an OPTIONAL <validate:authInfo> element (as described in section 2.3).
- o zero or more <validate:kv> elements (as described in section 2.1).

The following is an example <check> command where "sh8013" and "sh8014" are both "new" contacts and "sh8012" is an existing contact on the server.

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
C:  xmlns:validate="urn:ietf:params:xml:ns:validate-0.2"
C:  xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:  <command>
C:    <check>
C:      <validate:check>
C:        <validate:contact contactType="registrant" tld="COM">
C:          <validate:id>sh8013</validate:id>
C:          <validate:postalInfo type="int">
C:            <contact:name>John Doe</contact:name>
C:            <contact:org>Example Inc.</contact:org>
C:            <contact:addr>
C:              <contact:street>123 Example Dr.</contact:street>
C:              <contact:street>Suite 100</contact:street>
C:              <contact:city>Dulles</contact:city>
C:              <contact:sp>VA</contact:sp>
C:              <contact:pc>20166-6503</contact:pc>
C:              <contact:cc>US</contact:cc>
C:            </contact:addr>
C:          </validate:postalInfo>
C:          <validate:voice>+1.7035555555</validate:voice>
C:          <validate:fax>+1.7035555556</validate:fax>
C:          <validate:email>jdoe@example.com</validate:email>
C:          <validate:authInfo>
C:            <contact:pw>2fooBAR</contact:pw>
C:          </validate:authInfo>
```

```
   C:            <validate:kv key="VAT" value="1234567890"/>
   C:          </validate:contact>
   C:          <validate:contact contactType="tech" tld="COM">
   C:            <validate:id>sh8012</validate:id>
   C:          </validate:contact>
   C:          <validate:contact contactType="admin" tld="COM">
   C:            <validate:id>sh8014</validate:id>
   C:            <validate:postalInfo type="int">
   C:              <contact:name>John Doe</contact:name>
   C:              <contact:org>Example Inc.</contact:org>
   C:              <contact:addr>
   C:                <contact:street>123 Example Dr.</contact:street>
   C:                <contact:street>Suite 100</contact:street>
   C:                <contact:city>Dulles</contact:city>
   C:                <contact:sp>VA</contact:sp>
   C:                <contact:pc>20166-6503</contact:pc>
   C:                <contact:cc>US</contact:cc>
   C:              </contact:addr>
   C:            </validate:postalInfo>
   C:            <validate:voice>+1.7035555555</validate:voice>
   C:            <validate:fax>+1.7035555556</validate:fax>
   C:            <validate:email>jdoe@example.com</validate:email>
   C:            <validate:authInfo>
   C:              <contact:pw>2fooBAR</contact:pw>
   C:            </validate:authInfo>
   C:          </validate:contact>
   C:          <validate:contact contactType="billing" tld="COM">
   C:            <validate:id>sh8014</validate:id>
   C:            <validate:postalInfo type="int">
   C:              <contact:name>John Doe</contact:name>
   C:              <contact:org>Example Inc.</contact:org>
   C:              <contact:addr>
   C:                <contact:street>123 Example Dr.</contact:street>
   C:                <contact:street>Suite 100</contact:street>
   C:                <contact:city>Dulles</contact:city>
   C:                <contact:sp>VA</contact:sp>
   C:                <contact:pc>20166-6503</contact:pc>
   C:                <contact:cc>US</contact:cc>
   C:              </contact:addr>
   C:            </validate:postalInfo>
   C:            <validate:voice>+1.7035555555</validate:voice>
   C:            <validate:fax>+1.7035555556</validate:fax>
   C:            <validate:email>jdoe@example.com</validate:email>
   C:            <validate:authInfo>
   C:              <contact:pw>2fooBAR</contact:pw>
   C:            </validate:authInfo>
   C:          </validate:contact>
   C:        </validate:check>
```

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

When the server receives a <check> command with a <validate:contact> element that contains only a <validate:id> element the server will process this as an existing contact. If the contact does not exist the server MUST return an EPP error response for that specific <validate:contact>.

When a <check> command has been processed succesfully, the EPP <resData> element MUST contain a child <validate:chkData> element that identifies the validate namespace. The <validate:chkData> element MUST contain a <validate:cd> element for each <validate:check> element contained in the <check> command. The <validate:cd> element contains the following child elements:

- one <validate:id> element.
- one <validate:response> element.
- zero or more <validate:kv> elements.

The following is an example of the <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:      <validate:chkData
S:        xmlns:validate="urn:ietf:params:xml:ns:validate-0.2">
S:        <validate:cd>
S:          <validate:id>sh8013</validate:id>
S:          <validate:response>1000</validate:response>
S:        </validate:cd>
S:        <validate:cd>
S:          <validate:id>sh8014</validate:id>
S:          <validate:response>2306</validate:response>
S:          <validate:kv key="contact:city" value="City not valid
S:            for state."/>
S:          <validate:kv contactType="Admin" key="contact:cc"
S:            value="Invalid country code for admin, must be mx."/>
S:          <validate:kv contactType="Billing" key="VAT" value="VAT
S:            required for Billing contact."/>
S:        </validate:cd>
S:      </validate:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-ZYX</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

3.1.2. EPP <info> Command

Info semantics do not apply to validate objects, so there is no mapping defined for the EPP <info> command.

3.1.3. EPP <transfer> Command

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

3.2. EPP Transform Commands

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

3.2.1. EPP <create> Command

Create semantics do not apply to validate objects, so there is no mapping defined for the EPP <create> command.

3.2.2. EPP <delete> Command

Delete semantics do not apply to validate objects, so there is no mapping defined for the EPP <delete> command.

3.2.3. EPP <renew> Command

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

3.2.4. EPP <transfer> Command

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

3.2.5. EPP <update> Command

Update semantics do not apply to validate objects, so there is no mapping defined for the EPP <update> command.

4. Formal Syntax

One schema is presented here that is the EPP Validate 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. Validate Schema

```
   BEGIN
   <?xml version="1.0" encoding="UTF-8"?>
   <schema
      targetNamespace="urn:ietf:params:xml:ns:validate-0.2"
      xmlns:validate="urn:ietf:params:xml:ns:validate-0.2"
      xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
      xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
      xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
      xmlns="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified">

      <annotation>
        <documentation>
          Extensible Provisioning Protocol v1.0
          Validate Object
        </documentation>
      </annotation>

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

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

   <!--
   Child elements of the <check> command.
   -->
      <element name="check" type="validate:checkType"/>

      <complexType name="checkType">
        <sequence>
          <element name="contact"
            type="validate:validateContactType"
            maxOccurs="unbounded" />
        </sequence>
      </complexType>

      <complexType name="validateContactType">
        <sequence>
          <element name="id"
            type="eppcom:clIDType" />
          <element name="postalInfo"
            type="contact:postalInfoType"
            minOccurs="0" maxOccurs="2" />
          <element name="voice"
            type="contact:e164Type" minOccurs="0" />
          <element name="fax"
            type="contact:e164Type" minOccurs="0" />
          <element name="email"
            type="eppcom:minTokenType" minOccurs="0"/>
          <element name="authInfo"
            type="contact:authInfoType"
            minOccurs="0"/>
          <element name="kv"
            type="validate:kvType" minOccurs="0"
            maxOccurs="unbounded" />
        </sequence>
        <attribute name="contactType" type="eppcom:labelType"
          use="required"/>
        <attribute name="tld"
          type="eppcom:labelType" use="required"/>
      </complexType>

      <complexType name="kvType">
        <attribute name="contactType"
          type="eppcom:labelType" use="optional" />
        <attribute name="key"
```

```
          type="validate:keyType" use="required" />
       <attribute name="value"
          type="validate:valueType" use="required" />
      </complexType>

      <simpleType name="keyType">
        <restriction base="token">
          <minLength value="1" />
        </restriction>
      </simpleType>

      <simpleType name="valueType">
        <restriction base="token">
          <minLength value="0" />
        </restriction>
      </simpleType>

   <!--
   Child elements of the <check> response.
   -->
      <element name="chkData" type="validate:chkDataType" />

      <complexType name="chkDataType">
        <sequence>
          <element name="cd"
            type="validate:resCreateDataType" maxOccurs="unbounded" />
        </sequence>
      </complexType>

      <complexType name="resCreateDataType">
        <sequence>
          <element name="id"
            type="eppcom:clIDType" />
          <element name="response"
            type="epp:resultCodeType" />
          <element name="kv"
            type="validate:kvType"
            minOccurs="0" maxOccurs="unbounded" />
        </sequence>
      </complexType>

   </schema>
   END
```

5. Security Considerations

The mapping 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.

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].

Registration request for the validate namespace:

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

Registrant Contact: IESG

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

Registration request for the validate schema:

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

Registrant Contact: IESG

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

7. Implemntation 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".

7.1. To Be Filled In

Add implementation details once available.

8. Acknowledgements

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

- Kevin Allendorf of GoDaddy Inc.
- Jody Kolker of GoDaddy Inc.
- James Gould of Verisign Inc

9. Change History

9.1. Change from 03 to 04

Removed the <validate:cd> element from the <check> command, moving all sub-elements to the <validate:contact> element to simplify. Also removed the <disclose> element as it was not needed in this context. Also updated references to current versions of documents.

9.2. Change from 02 to 03

Corrected some formatting issues.

9.3. Change from 01 to 02

Corrected some formatting issues.

9.4. Change from 00 to 01

After review and broad feedback, extensive changes have been made transforming the original document from a standalone extension command to using the <check> command and response framework. Stubbed in an Implementation section for later documentation.

9.5. Change from carney-regext 01 to ietf-regext 00

Updated miscellaneous verbiage to reflect the change from an extension and changed to ietf naming as REGEXT WG will assume this work.

10. 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>.

[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>.

Authors' Addresses

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

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

Joseph Snitker
GoDaddy Inc.
14455 N. Hayden Rd. #219
Scottsdale, AZ 85260
US

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