Verification Code Extension for the Extensible Provisioning Protocol (EPP)
draft-ietf-regext-verificationcode-00

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 April 17, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
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 data verified by the VSP
MUST be stored by the VSP along with the generated verification code
to address any compliance issues. 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", "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.

"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.

Example of a "domain" typed signed code using the <verificationCode:signedCode> element and XML Signature [W3C.CR-xmldsig-core2-20120124]:

```xml
<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>
      </Reference>
      <DigestMethod
       Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <DigestValue>wgyW3nZPoEfppt1hRILKnOQnbdtU6ArM7ShrAfHgDFg=
      </DigestValue>
    </SignedInfo>
    <SignatureValue/>
  </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:
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 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:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
  </command>
</epp>

Example <verificationCode:encodedSignedCode> element that contains two <verificationCode:code> elements ;.
2.2 Verification Profile

A Verification Profile defines the set of verification code types required, supported, or not supported, and the grace period by which the verification code types MUST be set. 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:

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

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <domain:info
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>domain.example</domain:name>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:info>
    </info>
    <extension>
      <verificationCode:info
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</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="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 assigned 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 to 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:

```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:       <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</domain:crDate>
S:         <domain:exDate>2015-04-03T22:00:00.0Z</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:     <verificationCode:status>compliant</verificationCode:status>
S:     <verificationCode:profile name="sample">
S:       <verificationCode:status>compliant</verificationCode:status>
S:     </verificationCode:profile>
S:   </verificationCode:infData>
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:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <domain:infData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>domain.example</domain:name>
        <domain:roid>DOMAIN-REP</domain:roid>
        <domain:status s="ok"/>
        <domain:clID>ClientX</domain:clID>
        <domain:crID>ClientY</domain:crID>
        <domain:crDate>2010-04-03T22:00:00.0Z</domain:crDate>
        <domain:exDate>2015-04-03T22:00:00.0Z</domain:exDate>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:infData>
      <extension>
        <verificationCode:infData
          xmlns:verificationCode="urn:ietf:params:xml:ns:verificationCode-1.0">
        </verificationCode:infData>
      </extension>
    </resData>
  </response>
</epp>
```
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</domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        <verificationCode:status>compliant</verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>compliant</verificationCode:status>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain" date="2010-04-03T22:00:00.0Z"/>
S:            <verificationCode:code type="registrant" date="2010-04-03T22:00:00.0Z"/>
S:          </verificationCode:set>
S:        </verificationCode:profile>
S:      </verificationCode:infData>
S:    </extension>
S:  </response>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54322-XYZ</svTRID>
S:  </trID>
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:</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: <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</domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        <verificationCode:status>pendingCompliance</verificationCode:status>
S:        <verificationCode:profile name="sample">
S:          <verificationCode:status>pendingCompliance</verificationCode:status>
S:          <verificationCode:missing>
S:            <verificationCode:code type="registrant"
due="2010-04-08T22:00:00.0Z"/>
S:          </verificationCode:missing>
S:          <verificationCode:set>
S:            <verificationCode:code type="domain"  
date="2010-04-03T22:00:00.0Z">1-abc333</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:

```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:      <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</domain:crDate>
S:        <domain:exDate>2015-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:infData>
S:    </resData>
S:    <extension>
S:      <verificationCode:infData
S:        <verificationCode:status>notApplicable</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:

```xml
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:    <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:      "urn:ietf:params:xml:ns:verificationCode-1.0">
C:      "urn:ietf:params:xml:ns:domain-1.0">
C:    </verificationCode:encodedSignedCode>
C:  </extension>
C:  <verificationCode:encodedSignedCode>
C:    xmlns:verificationCode="urn:ietf:params:xml:ns:verificationCode-1.0">
C:      "urn:ietf:params:xml:ns:domain-1.0">
C:    </verificationCode:encodedSignedCode>
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 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>

END
<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="abstractSignedCodeType">
<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>

<complexType name="encodedSignedCodeType">
<complexContent>
<extension base="verificationCode:encodedSignedCodeListType">
</complexContent>
</complexType>

<complexType name="encodedSignedCodeListType">
</complexType>

<complexType name="encodedSignedCodeType">
<complexContent>
<extension base="verificationCode:encodedSignedCodeListType">
</extension>
</complexContent>
</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"/>
<element name="set" type="verificationCode:codesType" minOccurs="0"/>
</sequence>
<attribute name="name" type="token"/>
</complexType>

<complexType name="statusEnum">
  <restriction base="token">
    <enumeration value="notApplicable"/>
    <enumeration value="nonCompliant"/>
    <enumeration value="pendingCompliance"/>
    <enumeration value="compliant"/>
  </restriction>
</complexType>

<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>

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: See the "Author’s Address" section of this document.
    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: See the "Author’s Address" section of this document.
    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
6. 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.

7. Normative References


Appendix A. Acknowledgements

Appendix B. Change History

B.1. Change from 00 to 01

1. Fixed pendingCompliance and complaint to pendingCompliance and compliant in text.
2. Fixed verification 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.
Author’s Address

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

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