

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.

Copyright Notice

Copyright (c) 2019 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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Conventions Used in This Document	3
2.	Object Attributes	4
2.1.	Verification Code	4
2.1.1.	Signed Code	4
2.1.2.	Encoded Signed Code	7
2.2.	Verification Profile	11
3.	EPP Command Mapping	12
3.1.	EPP Query Commands	12
3.1.1.	EPP <check> Command	12
3.1.2.	EPP <info> Command	12
3.1.3.	EPP <transfer> Command	24
3.2.	EPP Transform Commands	25
3.2.1.	EPP <create> Command	25
3.2.2.	EPP <delete> Command	27
3.2.3.	EPP <renew> Command	28
3.2.4.	EPP <transfer> Command	28
3.2.5.	EPP <update> Command	28
4.	Formal Syntax	28
4.1.	Verification Code Extension Schema	28
5.	IANA Considerations	32
5.1.	XML Namespace	32
5.2.	EPP Extension Registry	32
6.	Implementation Status	33
6.1.	Verisign EPP SDK	33
6.2.	Net::DRI	34
7.	Security Considerations	34
8.	References	35
8.1.	Normative References	35
8.2.	Informative References	36
Appendix A.	Acknowledgements	36
Appendix B.	Change History	36
B.1.	Change from 00 to 01	36
B.2.	Change from 01 to 02	36
B.3.	Change from 02 to 03	36
B.4.	Change from 03 to 04	36
B.5.	Change from 04 to REGEXT 00	37
B.6.	Change from REGEXT 00 to REGEXT 01	37
B.7.	Change from REGEXT 01 to REGEXT 02	37
B.8.	Change from REGEXT 02 to REGEXT 03	37
B.9.	Change from REGEXT 03 to REGEXT 04	37

Gould

Expires July 14, 2019

[Page 2]

[B.10](#). Change from REGEXT 04 to REGEXT 05 [37](#)
[B.11](#). Change from REGEXT 05 to REGEXT 06 [37](#)
 Author's Address [38](#)

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-xmlsig-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-xmlsig-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-xmlsig-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-xmlldsig-core2-20120124](#)] for the `<verificationCode:signedCode>`. Use of a namespace prefix, like "dsig", is recommended for the XML Signature [[W3C.CR-xmlldsig-core2-20120124](#)] elements.

Example of a "domain" typed signed code using the `<verificationCode:signedCode>` element and XML Signature [[W3C.CR-xmlldsig-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/xmlldsig#">
    <SignedInfo>
      <CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <SignatureMethod
Algorithm="http://www.w3.org/2001/04/xmlldsig-more#rsa-sha256" />
      <Reference URI="#signedCode">
        <Transforms>
          <Transform
Algorithm="http://www.w3.org/2000/09/xmlldsig#enveloped-signature" />
```


Gould

Expires July 14, 2019

[Page 5]

```

    </Transforms>
    <DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>wgyW3nZPoEfptlhRILKn0QnbdU6ArM7ShrAfHgDFg=
</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>
jMu4PfyQGjJBF0GWSEPFcJjmywCEqR2h4LD+ge6XQ+JnmKFFCuCZS/3SLKAX0L1w
QDF02e0Y69k2G7/LGE37X3v0flobFM1oGwja8+GMVraoto5xAd4/AF7eHukgAymD
o9toxo2h0yV4A4PmXzsU6S86XtCcUE+S/WM72nyn47zoUCzzPKHZBRyehVfQ+
jYRMIAMz57HHQA+6eaXefRvtPETgU04aVIVSugc40UAZZwbYcZrC6w0aQqqqAZi
30aPOBYbAvHMSmWSS+hFkbshomJfHxb97TD2grLYNrQIzqXk7WbHWy2SYdA+sI/Z
ipJsXNa6osTUw1CzA7jfwA==
    </SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate>
MIIESTCCAzGgAwIBAgIBAgIBANBgkqhkiG9w0BAQsFADBIMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJQ0FO
TiBUTUNIMRswGQYDVQQDExJJQ0FOTiBUTUNIIFRFU1QgQ0EwHhcNMTMwMjA4MDAw
MDAwWhcNMTMwMjA3MjM1OTU5WjBsmQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0Ex
FDASBgNVBACTC0xvcyBBbmdlbGVzMRcwFQYDVQQKEw5WYXpZGF0b3IgcVE1DSDEh
MB8GA1UEAxMYVmFsaWRhdG9yIFRNQ0ggVEVTVCBDRVJUMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAo/cwvXhbVYl0RDWwv0yeZpETVZVvcMCovUVNg/sw
WinuMgEWgVQFrz0xA04pEhXCFVv4evbUpekJ5buqU1gmQy0sCKQlh0HTdPjvkC5u
pDqa51Flk0TMaMkIqjs7aUKCmA4Rg4tTTGK/EjR1ix8/D0ghYVVRldy1YPrMP+ou7
5b0VnIos+HifrAtrIv4qEqwLL4FTZAUpaCa2BmgXfy2CSRQbxD50r1gcSa3vurh5
sPMCnxqaxmIXmQipS+DuEBqMM8tldaN7RYojUEKrgVsnk5i9y2/7sjn1zyyUPf7v
L4GgDYqhJYWV61DnXgx/Jd6CwXvsnDF6scscQzUTel+hywIDAQAB04H/MIH8MAwG
A1UdEwEB/wQCMAAwHQYDVR00BBYEFpZEcIQcD/Bj2IFz/LERuo2ADJviMIGMBgNV
HSMEgYQwgYGAFO0/7kEh3FuEKS+Q/kYHaD/W6wihoWakZDBIMQswCQYDVQQGEwJV
UzELMAkGA1UECBMCQ0ExFDASBgNVBACTC0xvcyBBbmdlbGVzMRMwEQYDVQQKEwpJ
Q0FOTiBUTUNIMRswGQYDVQQDExJJQ0FOTiBUTUNIIFRFU1QgQ0GCAQEWdgYDVR0P
AQH/BAQDAgeAMC4GA1UdHwQnMCUwI6AhoB+GHWh0dHA6Ly9jcmwuaWNhbm4ub3Jn
L3RtY2guY3JsMA0GCSqGSIb3DQEBwUAA4IBAQB2qSy7ui+43cebKUKwWPrzz9y/
IkrMeJGkjo40n+9uekaw3DJ5Eqi0f/qZ4pjBD++oR6BJCb6NQuQKwnoAz5lE4Ssu
y5+i93oT3HfyVc4gNMIOhm1PS19l7DBKrbwbzAea/0jKwVzrvMv7TBfjxD3AQo1R
bU5dBr6IjbdLFln05x0G0mrG7x50UPuurihyiURpFDpwh8KAH1wMcCpXGXFrtGKk
wydgyVYAty7otkl/z3bZkCVT34gPvF70sR6+QxUy8u0LzF5A/beYaZpxSYG31amL
AdXitTWfipaIGea9LEGFM0L9+Bg7XzNn4nVLXokyEB3bgS4scG6QznX23FGk
      </X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
</verificationCode:signedCode>

```

Gould

Expires July 14, 2019

[Page 6]

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>
ICAgICAgPHZlcmImaWNhdGlvbkNvZGU6c2lnbmVkb29kZQogICAgICAgIHhtbG5z
OnZlcmImaWNhdGlvbkNvZGU9CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
bDpuczp2ZXJpZmlyYXRpb25Db2RlLlTEuMCIKICAgICAgICAgICAgICAgICAgICAg
b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
ZmlyYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
PENhbm9uawNhbG6YXRpb25NZXR0b2QKIEFsZ29yaXR0b2R0iaHR0cDovL3d3dy53
My5vcmcvMjAwMS8xMC94bWwtZXhjLWwNG4jIi8+CiAgICAgICAgICAgICAgICAgICAg
aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDUvMDQveG1sZHNp
Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVkb2
9kZSI+CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
Q29kZSI+CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
aXR0b2R0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3B3
ZC1zaWduYXR1cmUiLz4KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
YyNzaGEyNTYiLz4KIDxEdEwlc3RwYXNzLz4KICAgICAgICAgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
UW5iZHRVnkFyTtTaHJkZkhkbnREZnR0b2R0LnZlcnR0LnZlcnR0LnZlcnR0LnZlcn
ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICAgICAgICAgICAgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
UGZ5U0dpSkJGMEduU0VQRkNKam15d0NfCvIyaDRMRctnZTZyUStKbm1LRkZDdUNA
Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNY9MR0UzN1gzdk9mbG9iRk0xb0d3
amE4K0dNvNjhb3RvNjB3ZDQvQUY3ZUhh1a2dBW1ECiBvOXRveG9hMmgweVY0QTRQ
bVh6c1U2UzgzWHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UETIwkJSEwVXZWhwRlEr
CiBqVWJNSUFnek01N0hIUUErNmVhWGVmUnZ0UEVUZ1VPNGFWSVZTdwdjNE9VQVpa
d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdUytoRmtic2hvbUpm
```

Gould

Expires July 14, 2019

[Page 7]

SHhi0TdURDJncmXZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYX1ZT4KICAgPEtleUluZm8+
CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTUlJRvNUQ0NB
ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVFzRkFEQmlNUXN3Q1FZRFZRUUdF
d0pWVXpFTAogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
bGJHVnpNuk13RVFZRFZRUUtdFd3BKUTBGTWogVGLCVVRVTklNUnN3R1FZRFZRUURF
eEpKUTBGT1RpQlVUVU5JSUZSRlUxUWdRMEV3SGhjtK1UTXdNake0TURBdwogTURB
d1doY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVLEVlFRR0V3SLZVekVMTUFr
R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
UVLEVlFRS0V3NVdZV3hWkdGMGIzSwdWRTFEU0RfaAogTUI4R0ExVUVBeE1ZVm1G
c2FXUmhkRzL5SUZSTLEwZ2dWRVZUVkNCRFJWSLVNSUlcSwpBTKJna3Foa2LHOXcw
QgogQVFFRkFBT0NBUThtBUlJQkNnS0NBuUVBby9jd3ZYaGJWwWwUkRXV3ZveWVa
cEVUVlpWmNNQ292VZ0Zy9zdWogV2ludU1nRVdnVlFGcnoweEEwNHBFaFhDRlZ2
NGV2YlVwZwtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
MFRNYU1rSVFqcZdhVUtDbUE0Ukc0dFRUR0svRwpSMwL40C9EMGdIwVZSbGR5MVlQ
ck1QK291NwogNWJpVm5Jb3MrSGlmcKf0ckl2NHFFcXdMTDRGVfpBVXBhQ2EyQm1n
WGZ5MkNTUlfieEQ1T3IXz2NTYTN2dXJOnQogc1BNQ054cWfYbUlYbVfpcFMrRHVF
QnFNTTh0bGRhtjdSWw9qVUVLckdWc05rNwk5eTIvN3NqbjF6eXlVUGY3dgogTDRH
Z0RZCwhKwVdwnjFEbLhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0LEQVFB
Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFbd0hRWURWUjBPQkJZRUZQWkVj
SVFjRC9CajJJRnovTEVSdW8yQURKdmlNSUdNQmd0VgogSFNNRwdZUXdnWudBRk8w
LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUvFHRXdk
VgogVXpFTE1Ba0dBmVVFQ0JNQ1EwRXhGREFTQmd0VkJBY1RDMHh2Y3lCQmJtZGxi
R1Z6TVJNd0VRWURWUvFLRXdwSgogUTBGT1RpQlVUVU5JTVJzd0dRWURWUvFERXhK
SlEwRk9UaUJVVfVOSUlgUKZVMVFuUTBHQ0FRRXdEZ1LEVlIwJAogQVFIL0JBUURB
Z2VBTUM0R0ExVWRId1FuTUNvd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXThhi
bTR1YjNkbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIZRFFFQkN3VUFBNElCQVFC
MnFteTd1aSS0M2NlyktVS3dXUHJ6ejl5LwogSwtyTWVKR0tqbzQwbis5dWvrYXcz
REo1RXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZ0UXVRS3dub0F6NwxFNfnzdQogeTur
aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUZE5bDdEQktyYndiekFLYS8waktXVnpydm1W
N1RCZmp4RDNBuW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
aWh5aVvScEZEchdIOEtBSDF3TWNdCfHhWEZSdEdLawogd3lkZ3lWwUF0eTdvDgts
L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEX6RjVBL2JlWwFacHhTWUczMwFt
TAogQWRyaXRUV0ZpcGFJR2Vh0WxFR0ZNMew5K0JnN1h6Tm40blZMMWg9reUVCM2Jn
UzRzY0c2UXpuWDIZrKdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8L1g1MDlE
YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPc9Z2XJpZmljYXRp
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
```

Gould

Expires July 14, 2019

[Page 8]

```

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>

```

ICAgICAgPHZlcmImaWNhdGlvbkNvZGU6c2lnbmVkJ29kZQogICAgICAgIHhtbG5zOnZlcmImaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYW1zOnhtbDpuczp2ZXJpZmlyYXRpb25Db2RlLlTEuMCIKICAgICAgICAgICAgIGlkPSJzaWduZWRDb2RlIj4KICAgcQk8dmVyaWZpY2F0aW9uQ29kZTpb2RlPjEtYWJjMTIzPC92ZXJpZmlyYXRpb25Db2RlOmNvZGU+CjAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CjAgIDxTaWduZWRJbmZvPgogICAgPENhbm9uaWNhbG6YXRpb25NZXR0b2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8xMC94bWwtZXhjLWxwXG4jIi8+CjAgICA8U2lnbmF0dXJlTWV0aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNpZy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJPSiJc2lnbmVkJ29kZSI+CjAgICAgPFYyW5zZm9ybXM+CjAgICAgIDxUcmFuc2Zvcml0KIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3B1ZC1zaWduYXR1cmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CjAgICAgPERpZ2VzdE1ldGhvZAogQWxb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVuYyZzaGEyNTYiLz4KIDxEaWdldi3RlYyWx1ZT53Z3LXM25aUG9FZnBwdGxoUk1MS25PUW5iZHRVnkFyTTdTaHJBJzkhREZnPTwvRGlnZXN0VmFsdWU+CjAgICA8L1JlZmVyaW5jZT4KICAgPC9TaWduZWRJbmZvPgogICA8U2lnbmF0dXJlVmFsdWU+CjAgICAgICAgICAgUGZ5U0dpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRctnZTZYUStKbm1LRkZDdUNAUy8zU0xLQXgwTDF3CiBRREZPMmUwWjY5azJHNy9MR0UzN1gzdk9mbG9iRk0xb0d3amE4K0dnVnJhb3RvNkxhZDZQvQUY3ZUh1a2dBew1ECiBvOXRveG9hMmgweVY0QTRqbVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UETIwkJSeVXZWWhWRLerCiBqVWJNSUfNek01N0hIUUErNmVhVGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa d2JZY1pyQzZ3T2FRcXFXQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm SHhi0TdURDJncmXZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z VVf3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYyWx1ZT4KICAgPETleUluZm8+ CiAgICA8WDUwOURhdGE+CjAgICA8WDUwOUNlcnRpZmlyYXRlPjogTUlJRvNUQ0NB ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVZzRkFEQmLNUXN3Q1FZRFZRUUdF d0pwVXpFTAogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k bGJHVnpNUk13RVfZRFZRUUtFd3BKUTBGTWogVGlCVVRVtkLNUUN3R1FZRFZRUURF eEpKUTBGT1RpQlVUVU5JSUZSRlUxUWdRMEV3SGhjTk1UTXdNake0TURBdwogTURB d1doY05NVGd3TWpBM01qTTFPFVU1V2pCc01Rc3dDUVlEVlFR0V3SLZVekVMTUFR R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG UVlEVlFRS0V3NVdZV3hwWkdGMGIzSwdWRTFEU0RFaAogTUI4R0ExVUUVBeE1ZVnM1G

Gould

Expires July 14, 2019

[Page 9]


```

WXB3RkR0MmZLY3JVcK1YV0hncE56K0oycTh6MwpTcVJMUEw0UmpnRww0eGhi0Xl5
cEx0ZC8xQXJXRVLhWwZEdUc1S3FYV05MRG5YVzJoQkEzK0R5wk82MFQKcTVPd0R5
ZVFSVLNPVWNXVE9FOTJssLZ4M014Q1V6d1hoL0Z0STlPbGtXK0ZPNVZNNTZlTmZq
UEhkULJVdjdZqZrM0NnWmFaSWFXNqp2RmJnTmJodFJVa0hsSVhnYVNGWDgvcFdV
RXFIY0dLTUxnRU1nbHBnQ3Rt0FlIcXVqb0tXUk0yUDNiK2h3ZTRsU0hSWVRjK0pB
eEluClU4Rdc1WnliWThnSWFuZUpR2dWVtk2T0tJTGQ5L0l0UVhaeHZnPT08L2Rz
aWc6U2lnbmF0dXJlVmFsdWU+PGRzaWc6S2V5SW5mbz48ZHNPZzpYNTA5RGF0YT48
ZHNPZzpYNTA5Q2VydGlmawNhdGU+TUlJRGlUQ0NBbkdnQXdJQkFnSUVmcXE2SFRB
TkJna3Foa2lHOxcwQkFRc0ZBREIxTVJBd0RnWURWUVFHRXdkVmJtdHViM2R1TVJB
dwpEZ1LEVlFRSUV3ZFZibXR1YjNkdU1SQXdEZ1LEVlFRSEV3ZFZibXR1YjNkdU1S
QXdEZ1LEVlFRS0V3ZFZibXR1YjNkdU1SQXdEZ1LEClZRUUxZFd2RWYm10dWIzZHVN
Umt3RndZRFZRUURFeEiYwLhKcFptbGpZWfJwYjI1RGIyUmXNQjRyRFRFMU1EWXh0
VEl4TURBeU1sb1gKRFRNMU1EWXhNREl4TURBeU1sb3dkVEVRTUE0R0ExVUVCaE1I
Vlc1cmJtOTNiakVRTUE0R0ExVUVdQk1IVlc1cmJtOTNiakVRTUE0RwpBMVVFQnhN
SFZXNXJibTkzYmpFUU1BNEdBmVVFQ2hNSFZXNXJibTkzYmpFUU1BNEdBmVVFQ3hN
SFZXNXJibTkzYmpFwk1CY0dBmVVFckF4TVFkbVZ5YVdacFkyRjBhVz1lUTI5a1pU
Q0NBU0l3RFFZSkvtWk1odmNOQVFFQkJRQRnZ0VQQURDQ0FRb0NnZ0VCQUppjY2pY
cmsKUWFJL2lHUEZ3WmVITjFnRFVhcTltVnJmQis2eWR5Qmdoc2FHVFZoaERIOFN0
TmtPamxIMkxCQ3J3TjhjVjhQZ1BPOXRwbG9rR2F5UwpXNktFaHZtTk03b1dsZk5L
SkdSdGnidGMzTnJuYzhiUUJacU1xcFo0ULNRTmh5QWh6Ri85UmErd3Rfc0JWeGF3
Vdc1L2J0SDZ1YytmClJ0dE5FcmhJdVlJUmN0WTZIRmRaR3Bls3cxYnlyK0RsNkJP
L3ZLdnQ4NDlly1R3aEZIcDUwWGH2NFVTL0Z5aWVLaGs3dDdHRnJGRlQKL2NCTGsy
WmxFa1lLcFlEU2dLc2lseFg2QkpTZVdCbXZlZlZlL2pBZDhNwRmRmVUg2aHNHRXB1
U1BmZkZQV3FwcXl6V0p5bG910XF4ZQpnUTZj0Fo2SVpXZkUzakxSOUVYSDhz0TFD
Mm1pTFZrQ0F3RUFBYU1oTUI4d0hrWURWUjBPQkJZRUIY0JLdk03dmk3dUZNTUx5
ZE43cmVGVXF2YzVVTUEwR0NtCudTSWIZRFFFQkN3VUFBNELCQVFBVjB2cmLrSWRB
d2l4THZ0NUx5eXpTnFdTU1d0dVlWl2JQMvg3NzVMRmYKSWh3a2xoMENidk5rYXlk
Tms2Tnp0eDlSc1AwNWZndkxrZER1N0V5cnRzY3I1ZVdETG1WMGtKMWE1N1Z4bnJh
aEdLTnM2Wit1Ui9pSapMatJXb3liWepFT2N0NwtJSjFzL05CeUUrkdGdjFoTmJz
dVVVUEVCYwVtawpYUFRO0WxxZE9uM1FIbktobXhsa1czYS9KbmhtT20vCkRWYTE0
NDJXTVVUSluyVflwVldtdUs2NFkwQXFRn2FlZkVzIzZEcrT2xh0W9VYnBrSXJr
dDRDN3hRa0d5SXN2eUo3bi910FhBRDIKbno1T1cvek5GwnlrZDAzT2N3M240NkZx
c1IwVDlBbFBewHqXUjlmMjZmd1lxdjk3dWtVNEcrMVRJNHORV0F2TctVRK9FVnNu
PC9kc2ln0lg1MDlDZjX0awZpY2F0ZT48L2RzaWc6WDUwOURhdGE+PC9kc2ln0ktl
eUluZm8+PC9kc2ln0lNpZ25hdHVyZT48L3ZlcmMawNhdGlvbkNvZGU6c2lnbmVl
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 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 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:registrar>jd1234</domain:registrar>
 C: <domain:contact type="admin">sh8013</domain:contact>
 C: <domain:contact type="tech">sh8013</domain:contact>
 C: <domain:authInfo>
 C: <domain:pw>2fooBAR</domain:pw>
 C: </domain:authInfo>
 C: </domain:create>
 C: </create>
 C: <extension>
 C: <verificationCode:encodedSignedCode
 C: xmlns:verificationCode=
 C: "urn:ietf:params:xml:ns:verificationCode-1.0">
 C: <verificationCode:code>
 C: ICAgICAgPHZlcmLmaWNhdGlvbknvZGU6c2lnbmkVbQ29kZQogICAgICAgIHhtbG5z
 C: OnZlcmLmaWNhdGlvbknvZGU9CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: bDpucz2ZXJpZmZlYXRpb25Db2RlLTEuMCIKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
 C: ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
 C: dy53My5vcvcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbMZVPgogICAg
 C: PENhbm9uawNhbG6YXRpb25NZXR0b2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
 C: My5vcvcvMjAwMS8xMC94bWwtZXhjLWVxNG4jIi8+CiAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG5zZHNp
 C: Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: Q29kZSI+CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: aXRobT0iaHR0cDovL3d3dy53My5vcvcvMjAwMC8wOS94bWxkc2lnI2VudmVs3B3L3d3dy53
 C: ZC1zaWduYXR1cmUiLz4KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: dGhvZAogQWxb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbG9u
 C: YyNzaGEyNTYiLz4KIDxTaWduZWRJbMZVPgogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: UW5iZHRVnkFyTtdTaHJBZkhREZnPTwvRGlnZXN0VmFsdWU+CiAgICAgICAgICAgICAgICAgICAgICAg
 C: ZW5jZT4KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
 C: UGZ5UUpSkJGMEduU0VQRkNkM15d0NFcVIyaDRMRcTnZTZYUStKbm1LRkZDdUNa
 C: Uy8zU0xLQXgwTDF3CiBRREZPMUwWTY5azJHNY9MR0UzN1gzdk9mbG9iRk0xb0d3
 C: amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUhh1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
 C: bVh6c1U2UzgzWHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UETIwkJSewVXZWhwRlEr
 C: CiBqVWJNSUFNek01N0hIUUErNmVhVGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
 C: d2JZY1pyQzZ3T2FRcXFxQVppCiAZMm95b2p5Z3Z3T2FRcXFxQVppCiAZMm95b2p5Z3Z3T2FRcXFxQVpp
 C: SHhi0TdURDIncMzTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
 C: VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYXZ3Z3T2FRcXFxQVppCiAZMm95b2p5Z3Z3T2FRcXFxQVpp
 C: CiAgICAg
 C: ekdnQXdJkFkFnSUJBakF0QmdrcWhraUc5dzBCQVFzRkFEQmLnUXN3Q1FZRFZRUF
 C: d0pWVXpFTAogTUFrR0ExVUdVQk1DUTBFEEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
 C: bGJVHvnpNuk13RVFZFZRUFtFd3BKUTBGTWogVGlcVVRVtkLnUnN3R1FZRFZRUF
 C: eEpkUTBGT1RpQlVUUV5JSUZSRlUxUWdRMEV3SGhjT1UTxdNake0TURBdwogTURB
 C: d1doY05NVGd3TwpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3S1ZVekVMTUFR


```

C: R0ExVUVDQk1DUTBFaAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
C: UVlEVlFRS0V3NvdZV3hwWkdGMGIzSwdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
C: c2FXUmhkRz15SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkJna3Foa2lHOXcw
C: QgogQVFFRkFBT0NBUThtBTUljQkNnS0NBuUVBby9jd3ZYaGJWwwwUkRXV3ZveVva
C: cEVUVlpwVmNNQ292VVZ0Zy9zdWogV2ludU1nRVdnVlFGcnoweEEwNHBFaFhDRlZ2
C: NGV2YlVwZwtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
C: MFRNYU1rSVFqcZdhVUtDbUE0Ukc0dFRUR0svRwPsmWl40C9EMGdIwVZSbGR5MVlQ
C: ck1QK291NwogNWJPVm5Jb3MrSGlmcKf0ckl2NHFFcXdmTDRGVFPbVXBhQ2EyQm1n
C: WGZ5MkNTUlFieEQ1T3IxZ2NTYTN2dXJONQogc1BNQ054cWFYbUlYbVFcFMrRHVF
C: QnFNTTh0bGRhtJdSWw9qVUVLckdWc05rNwk5eTivN3NqbjF6eXlVUGY3dgogTDRH
C: Z0RZcWhKWdwnJfEBlhneC9KZDZDV3h2c25ERjZzY3NjUXpvVEVsK2h5d0lEQVFB
C: Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hrWURWUjBPQkJZRZUQWkVj
C: SVFjRC9CajJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRwdZUXdnWUdBRk8w
C: LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUvFHRXdk
C: VgogVXpFTE1Ba0dBmVVFQ0JNQ1EwRXhGREFTQmdOVkBY1RDMHh2Y3lCQmJtZGxi
C: R1Z6TVJNd0VRWURWUvFLRXdwSgogUTBGT1RpQlVUVU5JTVJzd0dRURWUvFERXhK
C: SlEwRk9UaUJVVfV0SulGukZVMVFntUBHQ0FRRXdEZ1lEVlIwUAogQVFIL0JBUURB
C: Z2VBTUM0R0ExVVRid1FuTUNvd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXtmhi
C: bTR1YjNkbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
C: MnFTEtd1aSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWvrYXcz
C: REo1RXFpT2YvcVo0cGpCRCSrb1I2QkpDYjZ0UXVRS3dub0F6NwxFNFNzdQogeTUR
C: aTkzb1QzSGZ5VmM0Z05NSW9IbTFQzE5bDdEQktyYndiekFLYS8waktXVnpydm1W
C: N1RCZmp4RDNBuW8xUgogYlU1ZEJyNkLqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
C: aWh5aVVSceZEchdIOEtBSDF3TWndcFhHWEZSdEdLawogd3lkZ3lWwUF0eTdvdGts
C: L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWwFacHhTWUczMwFt
C: TAogQWRyaXRUV0ZpcGFJR2Vh0WxFR0ZNMew5K0JnN1h6Tm40bLZMWG9reUVM2Jn
C: UzRzY0c2UXpuWDIZrKdrCiAgIDwvDUU0UNlcnRpZmljYXRlPogICA8L1g1MDlE
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-xmlsig-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-xmlsig-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-xmlsig-core2-20120124, January 2012, <<http://www.w3.org/TR/2012/CR-xmlsig-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:

- o Gurshabad Grover
- o Rick Wilhelm
- o 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>

