Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 29, 2017

X. Jiagui
Teleinfo
J. Gould
VeriSign, Inc.
L. Hongyan
Teleinfo

October 26, 2016

Extensible Provisioning Protocol (EPP) China Name Verification Mapping draft-ietf-regext-nv-mapping-00

Abstract

This document describes an Extensible Provisioning Protocol (EPP) for the provisioning and management of Name Verification (NV) stored in a shared central repository in China. Specified in XML, the mapping defines EPP command syntax and semantics as applied to name verification.

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 29, 2017.

Copyright Notice

Table of Contents

1. Introduction

When creating a domain name which will be stored in a shared central repository, some registry administrative organizations require the verification of the domain name and the real name based on legal or policy requirements.

The domain name verification, means to verify the domain label is in compliance with laws, rules and regulations. The real name verification, means to verify that the registrant really exists and is authorized to register a domain name.

The verification of this document meets the requirements in China, but MAY be applicable outside of China.

In order to meet above requirements of the domain name registration, this document describes the Extensible Provisioning Protocol (EPP) Name Verification (NV) Mapping. This document is specified using the Extensible Markup Language (XML) 1.0 as described in [W3C.REC-xml-20040204] and XML Schema notation as described in [W3C.REC-xmlschema-1-20041028] and [W3C.REC-xmlschema-2-20041028].

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

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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

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

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

3. Definitions and Object Attributes

3.1. Definitions

The following definitions are used in this document:

- Domain Name Verification(DNV), represents the verification of the domain's label is in compliance with laws, rules and regulations.
- Real Name Verification(RNV), represents the verification of the registrant(real name) is in compliance with laws, rules and regulations.
- Name Verification(NV), represents DNV, RNV or both of them.
- Verification Service Provider(VSP), collects the proof of materials for Name Verification(NV) and performs the verification.
- Verification Code, which is described in [I-D.gould-eppext-verificationcode] ,is a formatted token, referred to as the Verification Code Token, that is digitally signed by a Verification Service Provider (VSP) using XML Signature in "W3C.CR-xmldsig-core2-20120124".
- Signed Code, which is described in

[I-D.gould-eppext-verificationcode], is the XML Signature format of the Verification Code.

- Encoded Signed Code, which is described in [I-D.gould-eppext-verificationcode], is the "base64" encoded XML Signature format of the Verification Code.

- Prohibited Name(PN), is a domain label that is prohibited from registration.

- Restricted Name(RN), is a domain label that is restricted from registration. Additional information is needed during Domain Name Verification(DNV) to authorize the registration of a Restricted Name.

3.2. Object Attributes

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

- Status Values. A NV object MUST always have one associated status value. The Status value can be set only by the server. The status value MAY be accompanied by a string of human-readable text that describes the rationale for the status applied to the object. The status of an object MAY change as a result of an action performed by a server operator. Status Value Descriptions:

  * pendingCompliant. The object verification is not complete and is pending completion. Please refer to Section 4.3 for details on handling offline review of NV objects with the pendingComplaint status.

  * compliant. The object is in compliance with the policy.

    * nonCompliant. The object is not in compliance with the policy.

- Dates and Times. Date and time attribute values MUST be represented in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in [W3C.REC-xmlschema-2-20041028] MUST be used to represent date-time values, as XML Schema does not support truncated date-time forms or lower case "T" and "Z" characters.

- Authorization Information. Authorization information is associated with NV objects to facilitate query operations. Authorization information is assigned when a NV object is created, and it might be updated in the future. This specification describes password-based authorization information, though other mechanisms are possible.

3.3. Name Verification Proofs

When performing name verification, some Verification Service Providers(VSP) MAY need to collect the proof of materials to verify the real name of a registrant. The proof of materials is defined with the following enumerated values:

- "poc" for Proof of Citizen(POC). The POC represents the citizen's identification card(ID) material.

- "poe" for Proof of Enterprise(POE). The POE represents the Organization Code Certificate(OCC) or Business License(BL) material.

- "poot" for Proof of Other Types(POOT). The POOT represents other certificate materials except the POC and POE.

4. EPP Command Mapping

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

managing NV via EPP.

4.1. EPP Query Commands

EPP provides three commands to retrieve NV information: <check> 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.

4.1.1. EPP <check> Command

The EPP <check> command is used to determine if the domain's label can be used to create a DNV object. It provides a hint that allows a client to anticipate the success or failure of creating a DNV object using the <create> command.

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

- One or more <nv:name> elements that contain the domain labels to be queried.

Example <check> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <nv:check xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:        <nv:name>example1</nv:name>
```

```
C:         <nv:name>example2</nv:name>
C:         <nv:name>example3</nv:name>
C:       </nv:check>
C:     </check>
C:     <clTRID>ABC-12345</clTRID>
C:   </command>
C:</epp>
```

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

- A <nv:name> element that contains the queried domain label. This element MUST contain an "avail" attribute whose value indicates object availability (can it be created or not) at the moment the <check> command was completed. A value of "1" or "true" means that the object can be created. A value of "0" or "false" means that the object can not be created. This element SHOULD contain a "restricted" attribute whose value indicates this name is a RN or not, with a default value of "0". A value of "1" or "true" means that the object is a RN Name. A value of "0" or "false" means that the object is not restricted.

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

Example <check> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <nv:chkData
S:       xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:cd>
S:          <nv:name avail="1">example1</nv:name>
S:        </nv:cd>
S:        <nv:cd>
S:          <nv:name avail="0">example2</nv:name>
S:          <nv:reason>In Prohibited Lists.</nv:reason>
S:        </nv:cd>
S:        <nv:cd>
S:          <nv:name avail="0" restricted="1">example3</nv:name>
S:        </nv:cd>
S:      </nv:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

4.1.2. EPP <info> Command

The EPP <info> command is used to retrieve information associated with a NV object. The response to this command MAY vary depending on the identity of the querying client, and server policy towards unauthorized clients. If the querying client is the sponsoring client, all available information MUST be returned. If the querying client is not the sponsoring client but the client provides valid authorization information, all available information MUST be returned. If the querying client is not the sponsoring client and the client does not provide valid authorization information, server policy determines which OPTIONAL elements are returned.

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

- A <nv:code> element that contains the Verification Code Token value. An "type" attribute MUST be used to identify the type of the query(Signed Code or Input Data). If the type is "signedCode", the successful response of the server MUST be the Signed Code of the verification code. If the type is "input", the successful response of the server MUST be the verification input data and the verification status.

- An OPTIONAL <nv:authInfo> element that contains authorization information associated with the NV object. If this element is not provided or if the authorization information is invalid, server policy determines if the command is rejected or if response information will be returned to the client.

Example <info> command to query the signed 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:    <info>
C:      <nv:info xmlns:nv="urn:ietf:params:xml:ns:nv-1.0"
C:        type="signedCode">
C:        <nv:code>abc-123</nv:code>
C:      </nv:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> command to query the input data:

```
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:      <nv:info xmlns:nv="urn:ietf:params:xml:ns:nv-1.0"
C:       type="input">
C:        <nv:code>abc-123</nv:code>
C:      </nv:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <info> command with authorization information:

```
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:      <nv:info xmlns:nv="urn:ietf:params:xml:ns:nv-1.0"
C:        type="signedCode">
C:        <nv:code>abc-123</nv:code>
```

```
C:        <nv:authInfo>
C:          <nv:pw>2fooBAR</nv:pw>
C:        </nv:authInfo>
C:      </nv:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```


When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <nv:infData> element that identifies the nv namespace. The <nv:infData> element has two forms based on the query type provided in the command: the Signed Code Form and the Input Form. The child element of the <nv:infData> element is defined for each form.

The Signed Code Form is returned when the command "type" attribute is set to "signedCode". The <nv:signedCode> element is used for the Signed Code Form that contains the following child elements:

- A <nv:code> element that contains the Verification Code Token value of the signed code with the "type" attribute to indicate the type of NV object. The "type" attribute value of "domain" indicates a DNV object and "real-name" indicates a RNV object.

- An OPTIONAL <nv:status> element that contains the current status using the status values defined in Section 3.2.

- A <nv:encodedSignedCode> element include:

  * A <verificationCode:code>element that is a "base64" encoded form of the digitally signed <verificationCode:signedCode> as defined in [I-D.gould-eppext-verificationcode].

Example <info> response of a Signed 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:      <nv:infData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:signedCode>
S:         <nv:code type="domain">abc-123</nv:code>
S:          <nv:status s="compliant"/>
S:          <nv:authInfo>
S:            <nv:pw>2fooBAR</nv:pw>
S:          </nv:authInfo>
S:          <nv:encodedSignedCode>
S:ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
S:OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYW1zOnht
S:bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWRD
S:b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
S:ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
```



```
S:dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
S:PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
S:My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+CiAgICA8U2lnbmF0dXJlTWV0
S:aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNp
S:Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVk
S:Q29kZSI+CiAgICAgPFRyYW5zZm9ybXM+CiAgICAgIDxUcmFuc2Zvcm0gIEFsZ29y
S:aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
S:ZC1zaWduYXR1cmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CiAgICAgPERpZ2VzdE1l
S:dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
S:YyNzaGEyNTYiLz4KIDxEaWdlc3RWYWx1ZT53Z3lXM25aUG9FZnBwdGxoUklMS25P
S:UW5iZHRVNkFyTTdTaHJBZkhnREZnPTwvRGlnZXN0VmFsdWU+CiAgICA8L1JlZmVy
S:ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICA8U2lnbmF0dXJlVmFsdWU+CiBqTXU0
S:UGZ5UUdpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRCtnZTZYUStKbm1LRkZDdUNa
S:Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzN1gzdk9mbG9iRk0xb0d3
S:amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUh1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
S:bVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWhWRlEr
S:CiBqWVJNSUFNek01N0hIUUErNmVhWGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
S:d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
S:SHhiOTdURDJncmxZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
S:VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYWx1ZT4KICAgPEtleUluZm8+
S:CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTUlJRVNUQ0NB
S:ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVFzRkFEQmlNUXN3Q1FZRFZRUUdF
S:d0pWVXpFTAogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
S:bGJHVnpNUk13RVFZRFZRUUtFd3BKUTBGTwogVGlCVVRVTklNUnN3R1FZRFZRUURF
S:eEpKUTBGT1RpQlVUVU5JSUZSRlUxUWdRMEV3SGhjTk1UXdNakE0TURBdwogTURB
```

```
S:d1doY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
S:R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
S:UVlEVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
S:c2FXUmhkRzl5SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkJna3Foa2lHOXcw
S:QgogQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBby9jd3ZYaGJWWWwwUkRXV3ZveWVa
S:cEVUVlpWVmNNQ292VVZOZy9zdwogV2ludU1nRVdnVlFGcnoweEEwNHBFaFhDRlZ2
S:NGV2YlVwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
S:MFRNYU1rSVFqczdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5MVlQ
S:ck1QK291NwogNWJPVm5Jb3MrSGlmckF0ckl2NHFFcXdMTDRGVFpBVXBhQ2EyQm1n
S:WGZ5MkNTUlFieEQ1T3IxZ2NTYTN2dXJoNQogc1BNQ054cWFYbUlYbVFpcFMrRHVF
S:QnFNTTh0bGRhTjdSWW9qVUVLckdWc05rNWk5eTIvN3NqbjF6eXlVUGY3dgogTDRH
S:Z0RZcWhKWVdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0lEQVFB
S:Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkVj
S:SVFjRC9CajJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRWdZUXdnWUdBRk8w
S:LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdK
S:VgogVXpFTE1Ba0dBMVVFQ0JNQ1EwRXhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
S:R1Z6TVJNd0VRWURWUVFLRXdwSgogUTBGT1RpQlVUVU5JTVJzd0dRWURWUVFERXhK
S:SlEwRk9UaUJVVFVOSUlGUkZVMVFnUTBHQ0FRRXdEZ1lEVlIwUAogQVFIL0JBUURB
S:Z2VBTUM0R0ExVWRId1FuTUNVd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXTmhi
S:bTR1YjNKbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
S:MnFTeTd1aSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
S:REo1RXFpT2YvcVo0cGpCRCsrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFNzdQogeTUr
S:aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUzE5bDdEQktyYndiekFlYS8waktXVnpydm1W
```



```
S:N1RCZmp4RDNBUW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
S:aWh5aVVScEZEcHdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
S:L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWWFacHhTWUczMWFt
S:TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCM2Jn
S:UzRzY0c2UXpuWDIzRkdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8L1g1MDlE
S:YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPC92ZXJpZmljYXRp
S:b25Db2RlOnNpZ25lZENvZGU+Cg==
S:          </nv:encodedSignedCode>
S:        </nv:signedCode>
S:      </nv:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The Input Code Form is returned when the command "type" attribute is set to "input". The <nv:input> element is used for the Input Form and contains a choice of two different child elements dependent on the type of NV object that matches the <nv:code> in the command. The <nv:dnv> child element is used for a DNV object and the <nv:rnv> child element is used for a RNV object.

The <nv:dnv> element is used for a DNV object and contains the following child elements:

- A <nv:name> element that contains the label of the domain.
- An OPTIONAL <nv:rnvCode> element containing the Verification Code Token value of a RNV object used for verification of a Restricted Name.

Example <info> response of a DNV:

```
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:      <nv:infData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:input>
```

```
S:          <nv:dnv>
S:            <nv:name>example</nv:name>
S:          </nv:dnv>
S:          <nv:authInfo>
S:            <nv:pw>2fooBAR</nv:pw>
S:          </nv:authInfo>
S:        </nv:input>
S:      </nv:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The <nv:rnv> element is used for a RNV object. The "role" attribute MUST be used to identify the role of the RNV object with the possible values of "person" or "org".

The <nv:rnv> element contains the following child elements:

- A <nv:name> element that contains the full name of the contact.
- A <nv:num> element that contains the citizen or the organization ID of the contact.
- A <nv:proofType> element that contains the proof material type of the contact based on the enumerated values defined in Name Verification Proofs (Section 3.3).
- Zero or more <nv:document> elements that contains the following child elements:
  - A <nv:fileType> element contains the type of the file.

  - A <nv:fileContent> element contains the "base64" encoded content of the file.

Example <info> response of a RNV person:

```
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:      <nv:infData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:input>
S:         <nv:rnv role="person">
S:           <nv:name>John Xie</nv:name>
S:           <nv:num>1234567890</nv:num>
S:           <nv:proofType>poc</nv:proofType>
S:           <nv:document>
S:             <nv:fileType>jpg</nv:fileType>
S:             <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAAD
S:               </nv:fileContent>
S:           </nv:document>
S:         </nv:rnv>
S:         <nv:authInfo>
S:           <nv:pw>2fooBAR</nv:pw>
S:         </nv:authInfo>
S:        </nv:input>
S:      </nv:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Example <info> response of a RNV organization:

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

```
S:       <nv:input>
S:        <nv:rnv role="org">
S:         <nv:name>John Xie</nv:name>
S:         <nv:num>1234567890</nv:num>
S:         <nv:proofType>poe</nv:proofType>
S:         <nv:document>
S:          <nv:fileType>jpg</nv:fileType>
S:          <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAAD
S:            </nv:fileContent>
S:         </nv:document>
S:        </nv:rnv>
S:        <nv:authInfo>
S:         <nv:pw>2fooBAR</nv:pw>
S:        </nv:authInfo>
S:       </nv:input>
S:      </nv:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

A server with a different information-return policy MAY provide less information in a response for an unauthorized client.

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

4.1.3. EPP <transfer> Command

Transfer semantics do not apply to Name Verification (NV) objects, so there is no mapping defined for the EPP <transfer> command.

4.2. EPP Transform Commands

EPP provides five commands to transform NV 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.

4.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create an NV object. In addition to the standard EPP

command elements, the <create> command MUST contain a <nv:create>

element that identifies the NV namespace. The <nv:create> elements contains a choice of two different child elements dependent on the type of NV object to create. The <nv:dnv> child element is used to create a DNV object and the <nv:rnv> child element is used to create a RNV object. AN <nv:authInfo> element contains authorization information to be associated with the NV object.

- The <nv:dnv> element is used for a DNV object and contains the following child elements:
  - A <nv:name> element that contains the label of the domain.
  - An OPTIONAL <nv:rnvCode> element containing the Verification Code Token value of a RNV object used for verification of a Restricted Name.
- The <nv:rnv> element is used for a RNV object. The "role" attribute MUST be used to identify the role of the RNV object with the possible values of "person" or "org". The <nv:rnv> element contains the following child elements:
  - A <nv:name> element that contains the full name of the contact.
  - A <nv:num> element that contains the citizen or the organization ID of the contact.
  - A <nv:proofType> element that contains the proof material type of the contact based on the enumerated values defined in Name Verification Proofs (Section 3.3).
  - Zero or more <nv:document> elements that contains the following child elements:
    - A <nv:fileType> element contains the type of the file.
    - A <nv:fileContent> element contains the "base64" encoded content of the file.

Example <create> command for a DNV object:

```
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:      <nv:create xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:       <nv:dnv>
C:         <nv:name>example</nv:name>
C:       </nv:dnv>
```



```
C:       <nv:authInfo>
C:         <nv:pw>2fooBAR</nv:pw>
C:       </nv:authInfo>
C:      </nv:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example <create> command for a RNV person object:

```
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:     <nv:create xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:       <nv:rnv role="person">
C:         <nv:name>John Xie</nv:name>
C:         <nv:num>1234567890</nv:num>
C:         <nv:proofType>poe</nv:proofType>
C:         <nv:document>
C:           <nv:fileType>jpg</nv:fileType>
C:           <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAAD
C:           </nv:fileContent>
C:         </nv:document>
C:       </nv:rnv>
C:       <nv:authInfo>
C:         <nv:pw>2fooBAR</nv:pw>
C:       </nv:authInfo>
C:      </nv:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
```

```
C:</epp>
```

Example <create> command for an RNV organization:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <nv:create xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:       <nv:rnv role="org">
C:        <nv:name>John Xie</nv:name>
C:        <nv:num>1234567890</nv:num>
C:        <nv:proofType>poe</nv:proofType>
C:        <nv:document>
C:         <nv:fileType>jpg</nv:fileType>
```



```
C:          <nv:fileContent>EABQRAQAAAAAAAAAAAAAAAAAAD
C:              </nv:fileContent>
C:        </nv:document>
C:       </nv:rnv>
C:       <nv:authInfo>
C:         <nv:pw>2fooBAR</nv:pw>
C:       </nv:authInfo>
C:      </nv:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <create> command has been processed successfully, the EPP <resData> element MUST contain a child <nv:creData> element that identifies the nv namespace. <nv:creData> element contains the either a <nv:success> element on success or a <nv:failed> element on failure.

- The <nv:success> element contains the following child elements:
  - A <nv:code> element that contains the id of the verification code with the required "type" attribute that defines the type of the verification code.
  - A <nv:status> element that contains the current status using

the status values defined in Section 3.2.

* A <nv:crDate> element that contains the date and time of nv object creation.

* A <nv:encodedSignedCode> element include:

  + A <verificationCode:code>element that is a "base64" encoded form of the digitally signed <verificationCode:signedCode> as defined in [I-D.gould-eppext-verificationcode].

o The <nv:failed> element contains the following child elements:

* A <nv:status> element that contains the current status using the status values defined in Section 3.2.

* A <nv:msg> element containing a human-readable description of the reason of the failure. The language of the response is identified via an OPTIONAL "lang" attribute. If not specified, the default attribute value MUST be "en" (English).

Example <create> response of success:


```
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:      <nv:creData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:       <nv:success>
S:        <nv:code type="domain">abc-123</nv:code>
S:        <nv:status s="compliant"/>
S:        <nv:crDate>2015-08-17T22:00:00.0Z</nv:crDate>
S:        <nv:encodedSignedCode>
S:ICAgICAgPHZlcmlmaWNhdGlvbkNvZGU6c2lnbmVkQ29kZQogICAgICAgIHhtbG5z
S:OnZlcmlmaWNhdGlvbkNvZGU9CiAgICAgICAgICAidXJuOmlldGY6cGFyYW1zOnht
S:bDpuczp2ZXJpZmljYXRpb25Db2RlLTEuMCIKICAgICAgICAgIGlkPSJzaWduZWRD
S:b2RlIj4KICAgCQk8dmVyaWZpY2F0aW9uQ29kZTpjb2RlPjEtYWJjMTIzPC92ZXJp
S:ZmljYXRpb25Db2RlOmNvZGU+CiAgPFNpZ25hdHVyZSB4bWxucz0iaHR0cDovL3d3
S:dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyI+CiAgIDxTaWduZWRJbmZvPgogICAg
```

```
S:PENhbm9uaWNhbGl6YXRpb25NZXRob2QKIEFsZ29yaXRobT0iaHR0cDovL3d3dy53
S:My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+CiAgICA8U2lnbmF0dXJlTWV0
S:aG9kCiBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNp
S:Zy1tb3JlI3JzYS1zaGEyNTYiLz4KICAgIDxSZWZlcmVuY2UgVVJJPSIjc2lnbmVk
S:Q29kZSI+CiAgICAgPFRyYW5zZm9ybXM+CiAgICAgIDxUcmFuc2Zvcm0KIEFsZ29y
S:aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
S:ZC1zaWduYXR1cmUiLz4KICAgICA8L1RyYW5zZm9ybXM+CiAgICAgPERpZ2VzdE1l
S:dGhvZAogQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVu
S:YyNzaGEyNTYiLz4KIDxEaWdlc3RWYWx1ZT53Z3lXM25aUG9FZnBwdGxoUklMS25P
S:UW5iZHRVNkFyTTdTaHJBZkhnREZnPTwvRGlnZXN0VmFsdWU+CiAgICA8L1JlZmVy
S:ZW5jZT4KICAgPC9TaWduZWRJbmZvPgogICA8U2lnbmF0dXJlVmFsdWU+CiBqTXU0
S:UGZ5UUdpSkJGMEdXU0VQRkNKam15d0NFcVIyaDRMRCtnZTZYUStKbm1LRkZDdUNa
S:Uy8zU0xLQXgwTDF3CiBRREZPMmUwWTY5azJHNy9MR0UzN1gzdk9mbG9iRk0xb0d3
S:amE4K0dNVnJhb3RvNXhBZDQvQUY3ZUh1a2dBeW1ECiBvOXRveG9hMmgweVY0QTRQ
S:bVh6c1U2Uzg2WHRDY1VFK1MvV003Mm55bjQ3em9VQ3p6UEtIWkJSeWVXZWhWRlEr
S:CiBqWVJNSUFNek01N0hIUUErNmVhWGVmUnZ0UEVUZ1VPNGFWSVZTdWdjNE9VQVpa
S:d2JZY1pyQzZ3T2FRcXFxQVppCiAzMGFQT0JZYkF2SE1TbVdTUytoRmtic2hvbUpm
S:SHhiOTdURDJncmxZTnJRSXpxWGs3V2JIV3kyU1lkQStzSS9aCiBpcEpzWE5hNm9z
S:VFV3MUN6QTdqZndBPT0KICAgPC9TaWduYXR1cmVWYWx1ZT4KICAgPEtleUluZm8+
S:CiAgICA8WDUwOURhdGE+CiAgICA8WDUwOUNlcnRpZmljYXRlPgogTUlJRVNUQ0NB
S:ekdnQXdJQkFnSUJBakFOQmdrcWhraUc5dzBCQVFzRkFEQmlNUXN3Q1FZRFZRUUdF
S:d0pWVXpFTAogTUFrR0ExVUVDQk1DUTBFeEZEQVNCZ05WQkFjVEMweHZjeUJCYm1k
S:bGJHVnpNUk13RVFZRFZRUUtFd3BKUTBGTwogVGlCVVRVTklNUnN3R1FZRFZRUURF
S:eEpKUTBGT1RpQlVUVU5JSUZSRlUxUWdRMEV3SGhjTk1UTXdNakE0TURBdwogTURB
S:d1doY05NVGd3TWpBM01qTTFPVFU1V2pCc01Rc3dDUVlEVlFRR0V3SlZVekVMTUFr
S:R0ExVUVDQk1DUTBFeAogRkRBU0JnTlZCQWNUQzB4dmN5QkJibWRsYkdWek1SY3dG
S:UVlEVlFRS0V3NVdZV3hwWkdGMGIzSWdWRTFEU0RFaAogTUI4R0ExVUVBeE1ZVm1G
S:c2FXUmhkRzl5SUZSTlEwZ2dWRVZUVkNCRFJWSlVNSUlCSWpBTkJna3Foa2lHOXcw
S:QgogQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBby9jd3ZYaGJWWWwwUkRXV3ZveWVa
```



```
S:cEVUVlpWVmNNQ292VVZOZy9zdwogV2ludU1nRVdnVlFGcnoweEEwNHBFaFhDRlZ2
S:NGV2YlVwZWtKNWJ1cVUxZ21ReU9zQ0tRbGhPSFRkUGp2a0M1dQogcERxYTUxRmxr
S:MFRNYU1rSVFqczdhVUtDbUE0Ukc0dFRUR0svRWpSMWl4OC9EMGdIWVZSbGR5MVlQ
S:ck1QK291NwogNWJPVm5Jb3MrSGlmckF0ckl2NHFFcXdMTDRGVFpBVXBhQ2EyQm1n
S:WGZ5MkNTUlFieEQ1T3IxZ2NTYTN2dXJoNQogc1BNQ054cWFYbUlYbVFpcFMrRHVF
S:QnFNTTh0bGRhTjdSWW9qVUVLckdWc05rNWk5eTIvN3NqbjF6eXlVUGY3dgogTDRH
S:Z0RZcWhKWVdWNjFEblhneC9KZDZDV3h2c25ERjZzY3NjUXpVVEVsK2h5d0lEQVFB
S:Qm80SC9NSUg4TUF3RwogQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZQWkVj
S:SVFjRC9CajJJRnovTEVSdW8yQURKdmlNSUdNQmdOVgogSFNNRWdZUXdnWUdBRk8w
S:LzdrRWgzRnVFS1MrUS9rWUhhRC9XNndpaG9XYWtaREJpTVFzd0NRWURWUVFHRXdK
S:VgogVXpFTE1Ba0dBMVVFQ0JNQ1EwRXhGREFTQmdOVkJBY1RDMHh2Y3lCQmJtZGxi
S:R1Z6TVJNd0VRWURWUVFLRXdwSgogUTBGT1RpQlVUVU5JTVJzd0dRWURWUVFERXhK
```

```
S:SlEwRk9UaUJVVFVOSUlGUkZVMVFnUTBHQ0FRRXdEZ1lEVlIwUAogQVFIL0JBUURB
S:Z2VBTUM0R0ExVWRId1FuTUNVd0k2QWhvQitHSFdoMGRIQTZMeTlqY213dWFXTmhi
S:bTR1YjNKbgogTDNSdFkyZ3VZM0pzTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQVFC
S:MnFTeTd1aSs0M2NlYktVS3dXUHJ6ejl5LwogSWtyTWVKR0tqbzQwbis5dWVrYXcz
S:REo1RXFpT2YvcVo0cGpCRCsrb1I2QkpDYjZOUXVRS3dub0F6NWxFNFNzdQogeTUr
S:aTkzb1QzSGZ5VmM0Z05NSW9IbTFQUzE5bDdEQktyYndiekFlYS8waktXVnpydm1W
S:N1RCZmp4RDNBUW8xUgogYlU1ZEJyNklqYmRMRmxuTzV4MEcwbXJHN3g1T1VQdXVy
S:aWh5aVVScEZEcHdIOEtBSDF3TWNDcFhHWEZSdEdLawogd3lkZ3lWWUF0eTdvdGts
S:L3ozYlprQ1ZUMzRnUHZGNzBzUjYrUXhVeTh1MEx6RjVBL2JlWWFacHhTWUczMWFt
S:TAogQWRYaXRUV0ZpcGFJR2VhOWxFR0ZNMEw5K0JnN1h6Tm40blZMWG9reUVCM2Jn
S:UzRzY0c2UXpuWDIzRkdrCiAgIDwvWDUwOUNlcnRpZmljYXRlPgogICA8L1g1MDlE
S:YXRhPgogICA8L0tleUluZm8+CiAgPC9TaWduYXR1cmU+CgkJPC92ZXJpZmljYXRp
S:b25Db2RlOnNpZ25lZENvZGU+Cg==
S:         </nv:encodedSignedCode>
S:        </nv:success>
S:       </nv:creData>
S:     </resData>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S:</epp>
```

Example <create> response of failed:

```
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:     <nv:creData xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:      <nv:failed>
S:        <nv:status s="nonCompliant"/>
```



```
S:        <nv:msg lang="en">
S:        The name of the object is not correct.
S:        </nv:msg>
S:      </nv:failed>
S:     </nv:creData>
```

```
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

4.2.2. EPP <delete> Command

Delete semantics do not apply to Name Verification (NV) objects, so there is no mapping defined for the EPP <delete> command.

4.2.3. EPP <renew> Command

Renew semantics do not apply to Name Verification (NV) objects, so there is no mapping defined for the EPP <renew> command.

4.2.4. EPP <transfer> Command

Transfer semantics do not apply to Name Verification (NV) objects, so there is no mapping defined for the EPP <transfer> command.

4.2.5. EPP <update> Command

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

- A <nv:code> element that contains the code of the a NV object to be updated.
- A <nv:chg> element that contains object attribute values to be changed.

A <nv:chg> element contains the following child elements:

o A <nv:authInfo> element that contains authorization information associated with the NV object. This mapping includes a password-based authentication mechanism, but the schema allows new mechanisms to be defined in new schemas.

Example <update> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <nv:update xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
C:        <nv:code>abc-123</nv:code>
C:        <nv:chg>
C:          <nv:authInfo>
C:            <nv:pw>2BARfoo</nv:pw>
C:          </nv:authInfo>
C:        </nv:chg>
C:      </nv:update>
C:    </update>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

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

Example <update> response:

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

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

4.3. Offline Review of Requested Actions

Commands are processed by a server in the order they are received from a client. Though an immediate response confirming receipt and processing of the command is produced by the server, a server operator MAY perform an offline review of requested transform commands before completing the requested action. In such situations, the response from the server MUST clearly note that the transform command has been received and processed but that the requested action is pending. The status of the corresponding object MUST clearly reflect processing of the pending action. The server MUST notify the client when offline processing of the action has been completed.

Examples describing a <create> command that requires offline review are included here. Note the result code and message returned in response to the <create> command.

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <nv:creData
S:       xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:pending>
S:          <nv:code type="domain">abc-123</nv:code>
S:          <nv:status s="pendingCompliant"/>
S:          <nv:crDate>2015-09-03T22:00:00.0Z</nv:crDate>
S:        </nv:pending>
S:      </nv:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The status of the NV object after returning this response MUST include "pendingCompliant". The server operator reviews the request

offline, and informs the client of the outcome of the review either by queuing a service message for retrieval via the <poll> command or by using an out-of-band mechanism to inform the client of the outcome of the review.

The service message MUST contain text that describes the notification in the child <msg> element of the response <msgQ> element. In addition, the EPP <resData> element MUST contain a child <nv:panData> element that identifies the NV namespace. The <nv:panData> element contains the following child elements:

A <nv:code> element that contains the id of the verification code with the required "type" attribute that defines the type of the verification code.

A <nv:paStatus> element that contains the current status descriptors associated with the NV.

A <nv:msg> element containing a human-readable description of the result. The language of the response is identified via an OPTIONAL "lang" attribute. If not specified, the default attribute value MUST be "en" (English).

A <nv:paDate> element that contains the date and time describing when review of the requested action was completed.

Example "review completed" service message:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Command completed successfully; ack to dequeue</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>2015-09-04T22:01:00.0Z</qDate>
S:      <msg>Pending action completed successfully.</msg>
S:    </msgQ>
S:    <resData>
S:      <nv:panData
S:       xmlns:nv="urn:ietf:params:xml:ns:nv-1.0">
S:        <nv:code type="domain">abc-123</nv:code>
S:        <nv:paStatus s="compliant"/>
S:        <nv:msg>The object has passed verification,
S:                      signed code was generated.</nv:msg>
S:        <nv:paDate>2015-09-04T22:00:00.0Z</nv:paDate>
S:      </nv:panData>
S:    </resData>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5. Formal Syntax

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

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

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



```
  elementFormDefault="qualified">

<annotation>
  <documentation>
         Extensible Provisioning Protocol v1.0
         Name Verification provisioning schema.
  </documentation>
</annotation>

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

<!--
Child elements found in EPP commands.
-->
<element name="check" type="nv:mNameType"/>
```

```
   <element name="create" type="nv:createType"/>
   <element name="info" type="nv:infoType"/>
   <element name="update" type="nv:updateType"/>

   <!--
   Child element of <check> command.
   -->
   <complexType name="mNameType">
    <sequence>
           <element name="name" type="eppcom:labelType"
            maxOccurs="unbounded"/>
    </sequence>
   </complexType>

   <!--
   Child elements of the <create> command.
   -->
   <complexType name="createType">
    <sequence>
           <choice>
             <element name="dnv" type="nv:dnvType"/>
             <element name="rnv" type="nv:rnvType"/>
           </choice>
     <element name="authInfo" type="nv:authInfoChgType"/>
    </sequence>
   </complexType>
```



```
   <complexType name="dnvType">
    <sequence>
           <element name="name" type="eppcom:labelType"/>
       <element name="rnvCode" type="token"
         minOccurs="0"/>
    </sequence>
   </complexType>

   <complexType name="rnvType">
    <sequence>
           <element name="name" type="nv:nameType"/>
           <element name="num" type="nv:nameType"/>
           <element name="proofType" type="nv:proofEumType"/>
           <element name="document" type="nv:documentType"
```

```
          minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="role" type="nv:roleType"
           default="person"/>
  </complexType>

  <simpleType name="proofEumType">
    <restriction base="token">
           <enumeration value="poc"/>
           <enumeration value="poe"/>
           <enumeration value="poot"/>
    </restriction>
  </simpleType>

  <simpleType name="roleType">
          <restriction base="token">
            <enumeration value="person"/>
            <enumeration value="org"/>
    </restriction>
  </simpleType>

  <simpleType name="nameType">
          <restriction base="token">
                  <minLength value="1"/>
                  <maxLength value="255"/>
          </restriction>
  </simpleType>

  <complexType name="documentType">
    <sequence>
          <element name="fileType" type="nv:fileType"/>
          <element name="fileContent" type="base64Binary"/>
    </sequence>
  </complexType>
```



```
  <simpleType name="fileType">
    <restriction base="token">
           <enumeration value="pdf"/>
           <enumeration value="jpg"/>
    </restriction>
  </simpleType>
```

```
<!--
Child elements of the <info> command.
-->
<complexType name="infoType">
 <sequence>
        <element name="code" type="token"/>
        <element name="authInfo" type="nv:authInfoChgType"
         minOccurs="0"/>
 </sequence>
 <attribute name="type" type="nv:infoFormType"
        default="signedCode"/>
</complexType>


<simpleType name="infoFormType">
  <restriction base="token">
         <enumeration value="input"/>
         <enumeration value="signedCode"/>
 </restriction>
</simpleType>

<!--
Child elements of the <update> command.
-->
<complexType name="updateType">
 <sequence>
        <element name="code" type="token"/>
        <element name="chg" type="nv:chgType"/>
 </sequence>
</complexType>

<!--
Data elements that can be changed.
-->
<complexType name="chgType">
 <sequence>
        <element name="authInfo" type="nv:authInfoChgType"/>
 </sequence>
</complexType>
```

```
<!--
Data elements of authInfoChgType.
-->
<complexType name="authInfoChgType">
 <choice>
        <element name="pw" type="eppcom:pwAuthInfoType"/>
        <element name="ext" type="eppcom:extAuthInfoType"/>
 </choice>
</complexType>

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

<!--
<check> response elements.
-->
<complexType name="chkDataType">
 <sequence>
        <element name="cd" type="nv:checkType"
         maxOccurs="unbounded"/>
 </sequence>
</complexType>

<complexType name="checkType">
 <sequence>
        <element name="name" type="nv:checkNameType"/>
        <element name="reason" type="eppcom:reasonType"
         minOccurs="0"/>
 </sequence>
</complexType>

<complexType name="checkNameType">
 <simpleContent>
        <extension base="eppcom:labelType">
          <attribute name="avail" type="boolean"
                use="required"/>
         <attribute name="restricted" type="boolean"
          default="0"/>
        </extension>
 </simpleContent>
</complexType>

<!--
```

```
   <create> response elements.
   -->
   <complexType name="creDataType">
    <choice>
           <element name="success" type="nv:successType"/>
           <element name="failed" type="nv:failedType"/>
           <element name="pending" type="nv:pendingType"/>
    </choice>
   </complexType>

   <complexType name="successType">
    <sequence>
     <element name="code" type="nv:verificationCodeType"/>
     <element name="status" type="nv:statusType"/>
           <element name="crDate" type="dateTime"/>
           <element name="encodedSignedCode"
           type="verificationCode:encodedSignedCodeType"/>
    </sequence>
   </complexType>

   <complexType name="failedType">
    <sequence>
           <element name="status" type="nv:statusType"/>
           <element name="msg" type="nv:msgType"/>
    </sequence>
   </complexType>

   <complexType name="pendingType">
    <sequence>
           <element name="code" type="nv:verificationCodeType"/>
     <element name="status" type="nv:statusType"/>
           <element name="crDate" type="dateTime"/>
    </sequence>
   </complexType>

   <!--
   Pending action notification response elements.
   -->
   <complexType name="panDataType">
    <sequence>
           <element name="code" type="nv:verificationCodeType"/>
           <element name="paStatus" type="nv:statusType"/>
           <element name="msg" type="nv:msgType"/>
```

```
        <element name="paDate" type="dateTime"/>
 </sequence>
</complexType>

<!--
```



```
<info> response elements.
-->
<complexType name="infDataType">
 <sequence>
  <choice>
        <element name="signedCode" type="nv:signedCodeType"/>
        <element name="input" type="nv:createType"/>
  </choice>
</sequence>
</complexType>

<complexType name="signedCodeType">
 <sequence>
        <element name="code" type="nv:verificationCodeType"/>
        <element name="status" type="nv:statusType"
         minOccurs="0"/>
        <element name="authInfo" type="nv:authInfoChgType"/>
        <element name="encodedSignedCode"
        type="verificationCode:encodedSignedCodeType"/>
 </sequence>
</complexType>

<complexType name="verificationCodeType">
  <simpleContent>
    <extension base="token">
      <attribute name="type" type="token"
        use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
Status is a combination of attributes and an optional
human-readable message that may be expressed in languages other
than English.
-->
```

```
<complexType name="statusType">
 <simpleContent>
        <extension base="normalizedString">
          <attribute name="s" type="nv:statusValueType"
                use="required"/>
          <attribute name="lang" type="language"
                default="en"/>
        </extension>
 </simpleContent>
</complexType>

<complexType name="msgType">
```



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

<simpleType name="statusValueType">
 <restriction base="token">
        <enumeration value="compliant"/>
        <enumeration value="nonCompliant"/>
  <enumeration value="pendingCompliant"/>
 </restriction>
</simpleType>

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

6. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the Unicode character set and its more compact representations including UTF-8. Conformant XML processors recognize both UTF-8 and UTF-16. Though XML includes provisions to identify

and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED.

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

7. IANA Considerations

7.1. XML Namespace

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

Registration request for the NV namespace:

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

- Registrant Contact: See the "Author's Address" section of this document.
- XML: None. Namespace URI does not represent an XML specification.

Registration request for the NV XML schema:

- URI: urn:ietf:params:xml:schema:nv-1.0
- Registrant Contact: See the "Author's Address" section of this document.
- XML: See the "Formal Syntax" section of this document.

7.2. EPP Extension Registry

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

Name of Extension: Extensible Provisioning Protocol (EPP) China Name Verification Mapping

Document status: Informational

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

8. Security considerations

Verification Code Tokens are digitally signed using XML Signature technology. The security considerations described in Section 12 of the W3C XML Signature Syntax and Processing Candidate Recommendation [W3C.CR-xmldsig-core2-20120124] apply to this specification as well. The object mapping described in this document does not provide any other security services or introduce any additional considerations beyond those described by [RFC5730] or those caused by the protocol layers used by EPP.

9. Acknowledgements

The authors especially thank the author of [RFC5730].

Useful comments and contributions were made by TBD.

10. Change History

RFC Editor: Please remove this section.

10.1. draft-xie-eppext-nv-mapping-00: Version 00

- First draft.

10.2. Change from 00 to 01

1. Made the <nv:code> element of the panDataType and pendingType require the "type" attribute in the XML schema.

2. Fixed the XML schema to include the <nv:rnvCode> OPTIONAL element.

3. Added the support for the OPTIONAL "lang" attribute for the <nv:msg> element of the <nv:failed> and <nv:panData> elements.

10.3. Change from 01 to 02

1. Ping update.

10.4. Change from EPPEXT 02 to REGEXT 00

1. Changed to regext working group draft by changing draft-xie-eppext-nv-mapping-02 to draft-ietf-regext-nv-mapping-00.

11. Normative References

[I-D.gould-eppext-verificationcode]
Gould, J., "Verification Code Extension for the Extensible Provisioning Protocol (EPP)", draft-gould-eppext-verificationcode-04 (work in progress), September 2016.

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

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

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

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <http://www.rfc-editor.org/info/rfc7451>.

[W3C.CR-xmldsig-core2-20120124]
Bartel, M., Boyer, J., Fox, B., LaMacchia, B., and E. Simon, ""XML Signature Syntax and Processing Version 2.0", W3C Candidate Recommendation 24 January 2012", January 2012, <http://www.w3.org/TR/2012/CR-xmldsig-core2-20120124/>.

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

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

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

Authors' Addresses

Xie Jiagui

Teleinfo
1#-21,gaolizhang Street,Haidian District
Beijing, Beijing 100095
China

Phone: +86 10 5884 6931
Email: xiejiagui@teleinfo.cn

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

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

Liu Hongyan
Teleinfo
1#-21,gaolizhang Street,Haidian District
Beijing, Beijing 100095
China

Phone: +86 10 5884 6931
Email: liuhongyan@teleinfo.cn