Network Working Group
Request for Comments: 5731
STD: 69
Obsoletes: 4931
Category: Standards Track

S. Hollenbeck
VeriSign, Inc.

August 2009

Extensible Provisioning Protocol (EPP) Domain Name Mapping

Abstract

This document describes an Extensible Provisioning Protocol (EPP) mapping for the provisioning and management of Internet domain names stored in a shared central repository. Specified in XML, the mapping defines EPP command syntax and semantics as applied to domain names. This document obsoletes RFC 4931.

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

This document describes an Internet domain name mapping for version 1.0 of the Extensible Provisioning Protocol (EPP). This mapping 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]. This document obsoletes RFC 4931 [RFC4931].

[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.

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.

1.1. Relationship of Domain Objects and Host Objects

The EPP mapping for host objects is described in [RFC5732]. This document assumes that domain name objects have a superordinate relationship to subordinate host name objects. For example, domain name "example.com" has a superordinate relationship to host name "ns1.example.com". EPP actions (such as object transfers) that do not preserve this relationship MUST be explicitly disallowed.

A host name object can be created in a repository for which no superordinate domain name object exists. For example, host name "ns1.example.com" can be created in the ".example" repository so that DNS domains in ".example" can be delegated to the host. Such hosts are described as "external" hosts in this specification since the name of the host does not belong to the namespace of the repository in which the host is being used for delegation purposes.

Whether a host is external or internal relates to the repository in which the host is being used for delegation purposes. Whether or not an internal host is subordinate relates to a domain within the repository. For example, host ns1.example1.com is a subordinate host of domain example1.com, but it is not a subordinate host of domain example2.com. ns1.example1.com can be used as a name server for example2.com. In this case, ns1.example1.com MUST be treated as an internal host, subject to the rules governing operations on subordinate hosts within the same repository.

Name server hosts for domain delegation can be specified either as references to existing host objects or as domain attributes that describe a host machine. A server operator MUST use one name server

specification form consistently. A server operator that announces support for host objects in an EPP greeting MUST NOT allow domain attributes to describe a name server host machine. A server operator that does not announce support for host objects MUST allow domain attributes to describe a name server host machine. When domain attributes are used to describe a name server host machine, IP addresses SHOULD be required only as needed to generate DNS glue records.

Name servers are specified within a <domain:ns> element. This element MUST contain one or more <domain:hostObj> elements or one or more <domain:hostAttr> elements. A <domain:hostObj> element contains the fully qualified name of a known name server host object. A <domain:hostAttr> element contains the following child elements:

- A <domain:hostName> element that contains the fully qualified name of a host.

- Zero or more OPTIONAL <domain:hostAddr> elements that contain the IP addresses to be associated with the host. Each element MAY contain an "ip" attribute to identify the IP address format. Attribute value "v4" is used to note IPv4 address format. Attribute value "v6" is used to note IPv6 address format. If the "ip" attribute is not specified, "v4" is the default attribute value. IP address syntax requirements are described in Section 2.5 of the EPP host mapping [RFC5732].

Example host-object name server elements for domain example.com:

```
<domain:ns>
  <domain:hostObj>ns1.example.net</domain:hostObj>
  <domain:hostObj>ns2.example.net</domain:hostObj>
</domain:ns>
```

Example host-attribute name server elements for domain example.com:

```
<domain:ns>
  <domain:hostAttr>
    <domain:hostName>ns1.example.net</domain:hostName>
    <domain:hostAddr
     ip="v4">192.0.2.2</domain:hostAddr>
    <domain:hostAddr
     ip="v6">1080:0:0:0:8:800:200C:417A</domain:hostAddr>
  </domain:hostAttr>
  <domain:hostAttr>
    <domain:hostName>ns2.example.net</domain:hostName>
  </domain:hostAttr>
</domain:ns>
```

1.2. Conventions Used in This Document

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

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

2. Object Attributes

An EPP domain 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.

2.1. Domain and Host Names

The syntax for domain and host names described in this document MUST conform to [RFC0952] and [RFC1123]. At the time of this writing, RFC 3490 [RFC3490] describes a standard to use certain ASCII name labels to represent non-ASCII name labels. These conformance requirements might change as a result of progressing work in developing standards for internationalized domain names. A server MAY restrict allowable domain names to a particular top-level domain, second-level domain, or other domain for which the server is authoritative. The trailing dot required when these names are stored in a DNS zone is implicit and MUST NOT be provided when exchanging host and domain names.

2.2. Contact and Client Identifiers

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

2.3. Status Values

A domain object MUST always have at least one associated status value. Status values can be set only by the client that sponsors a domain object and by the server on which the object resides. A client can change the status of a domain object using the EPP

<update> command. Each status value MAY be accompanied by a string of human-readable text that describes the rationale for the status applied to the object.

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

Status values that can be added or removed by a client are prefixed with "client". Corresponding status values that can be added or removed by a server are prefixed with "server". Status values that do not begin with either "client" or "server" are server-managed.

Status Value Descriptions:

- clientDeleteProhibited, serverDeleteProhibited

  Requests to delete the object MUST be rejected.

- clientHold, serverHold

  DNS delegation information MUST NOT be published for the object.

- clientRenewProhibited, serverRenewProhibited

  Requests to renew the object MUST be rejected.

- clientTransferProhibited, serverTransferProhibited

  Requests to transfer the object MUST be rejected.

- clientUpdateProhibited, serverUpdateProhibited

  Requests to update the object (other than to remove this status) MUST be rejected.

- inactive

  Delegation information has not been associated with the object. This is the default status when a domain object is first created and there are no associated host objects for the DNS delegation. This status can also be set by the server when all host-object associations are removed.

- ok

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

- pendingCreate, pendingDelete, pendingRenew, pendingTransfer, pendingUpdate

  A transform command has been processed for the object, but the action has not been completed by the server. Server operators can delay action completion for a variety of reasons, such as to allow for human review or third-party action. A transform command that is processed, but whose requested action is pending, is noted with response code 1001.

When the requested action has been completed, the pendingCreate, pendingDelete, pendingRenew, pendingTransfer, or pendingUpdate status value MUST be removed. All clients involved in the transaction MUST be notified using a service message that the action has been completed and that the status of the object has changed.

"ok" status MUST NOT be combined with any other status.

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

"pendingRenew" status MUST NOT be combined with either "clientRenewProhibited" or "serverRenewProhibited" status.

"pendingTransfer" status MUST NOT be combined with either "clientTransferProhibited" or "serverTransferProhibited" status.

"pendingUpdate" status MUST NOT be combined with either "clientUpdateProhibited" or "serverUpdateProhibited" status.

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

Other status combinations not expressly prohibited MAY be used.

2.4. Dates and Times

Date and time attribute values MUST be represented in Universal Coordinated Time (UTC) using the Gregorian calendar. The extended date-time form using upper case "T" and "Z" characters defined in

[W3C.REC-xmlschema-2-20041028] MUST be used to represent date-time values, as XML Schema does not support truncated date-time forms or lower case "T" and "Z" characters.

2.5. Validity Periods

A domain name object MAY have a specified validity period. If server policy supports domain-object validity periods, the validity period is defined when a domain object is created, and it MAY be extended by the EPP <renew> or <transfer> commands. As a matter of server policy, this specification does not define actions to be taken upon expiration of a domain object's validity period.

Validity periods are measured in years or months with the appropriate units specified using the "unit" attribute. Valid values for the "unit" attribute are "y" for years and "m" for months. The minimum allowable period value is one (1). The maximum allowable value is ninety-nine decimal (99). A server MAY support a lower maximum value.

2.6. Authorization Information

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

2.7. Other DNS Resource Record Attributes

While the DNS allows many resource record types to be associated with a domain, this mapping only explicitly specifies elements that describe resource records used for domain delegation and resolution. Facilities to provision other domain-related resource record types can be developed by extending this mapping.

The provisioning method described in this mapping separates discrete data elements by data type. This method of data definition allows XML Schema processors to perform basic syntax-validation tasks, reducing ambiguity and the amount of parsing and syntax-checking work required of protocol processors. Provisioning and extension methods that aggregate data into opaque strings are possible, but such methods should not be used because they impose additional parsing, interpretation, and validation requirements on protocol processors.

3. EPP Command Mapping

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

3.1. EPP Query Commands

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

3.1.1. EPP <check> Command

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

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

- One or more <domain:name> elements that contain the fully qualified names of the domain objects to be queried.

Example <check> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <domain:check
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:name>example.net</domain:name>
C:        <domain:name>example.org</domain:name>
C:      </domain: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 <domain:chkData> element that identifies the domain namespace. The <domain:chkData> element contains one or more <domain:cd> elements that contain the following child elements:

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

- An OPTIONAL <domain:reason> element that MAY be provided when an object cannot be provisioned. If present, this element contains server-specific text to help explain why the object cannot be provisioned. This text MUST be represented in the response language previously negotiated with the client; an OPTIONAL "lang" attribute 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:      <domain:chkData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:cd>
S:          <domain:name avail="1">example.com</domain:name>
S:        </domain:cd>
S:        <domain:cd>
S:          <domain:name avail="0">example.net</domain:name>
S:          <domain:reason>In use</domain:reason>
S:        </domain:cd>
S:        <domain:cd>
S:          <domain:name avail="1">example.org</domain:name>
S:        </domain:cd>
S:      </domain:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
```

```
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.1.2. EPP <info> Command

The EPP <info> command is used to retrieve information associated with a domain object. The response to this command MAY vary depending on the identity of the querying client, use of authorization information, and server policy towards unauthorized clients. 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 <domain:info> element that identifies the domain namespace. The <domain:info> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object to be queried. An OPTIONAL "hosts" attribute is available to control return of information describing hosts related to the domain object. A value of "all" (the default, which MAY be absent) returns information describing both subordinate and delegated hosts. A value of "del" returns information describing only delegated hosts. A value of "sub" returns information describing only subordinate hosts. A value of "none" returns no information describing delegated or subordinate hosts.

- An OPTIONAL <domain:authInfo> element that contains authorization information associated with the domain object or authorization information associated with the domain object's registrant or associated contacts. An OPTIONAL "roid" attribute MUST be used to identify the registrant or contact object if and only if the given authInfo is associated with a registrant or contact object, and not the domain object itself. 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 without 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:      <domain:info
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name hosts="all">example.com</domain:name>
C:      </domain: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:      <domain:info
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name hosts="all">example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw>2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain: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 <domain:infData> element that identifies the domain namespace. Elements that are not OPTIONAL MUST be returned; OPTIONAL elements are returned based on client authorization and server policy. The <domain:infData> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object.

- A <domain:roid> element that contains the Repository Object IDentifier assigned to the domain object when the object was created.

- Zero or more OPTIONAL <domain:status> elements that contain the current status descriptors associated with the domain.

- If supported by the server, one OPTIONAL <domain:registrant> element and one or more OPTIONAL <domain:contact> elements that contain identifiers for the human or organizational social information objects associated with the domain object.

- An OPTIONAL <domain:ns> element that contains the fully qualified names of the delegated host objects or host attributes (name servers) associated with the domain object. See Section 1.1 for a description of the elements used to specify host objects or host attributes.

- Zero or more OPTIONAL <domain:host> elements that contain the fully qualified names of the subordinate host objects that exist under this superordinate domain object.

- A <domain:clID> element that contains the identifier of the sponsoring client.

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

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

- An OPTIONAL <domain:exDate> element that contains the date and time identifying the end of the domain object's registration period.

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

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

- An OPTIONAL <domain:trDate> element that contains the date and time of the most recent successful domain-object transfer. This element MUST NOT be provided if the domain object has never been transferred.

- An OPTIONAL <domain:authInfo> element that contains authorization information associated with the domain object. This element MUST only be returned if the querying client is the current sponsoring client or if the client supplied valid authorization information with the command.

Example <info> response for an authorized client:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:registrant>jd1234</domain:registrant>
S:        <domain:contact type="admin">sh8013</domain:contact>
S:        <domain:contact type="tech">sh8013</domain:contact>
S:        <domain:ns>
S:          <domain:hostObj>ns1.example.com</domain:hostObj>
S:          <domain:hostObj>ns1.example.net</domain:hostObj>
S:        </domain:ns>
S:        <domain:host>ns1.example.com</domain:host>
S:        <domain:host>ns2.example.com</domain:host>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientX</domain:upID>
S:        <domain:upDate>1999-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo>
S:          <domain:pw>2fooBAR</domain:pw>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

A server with a different information-return policy MAY provide less information in a response.

Example <info> response for an unauthorized 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>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:clID>ClientX</domain:clID>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.1.3. EPP <transfer> Query Command

The EPP <transfer> command provides a query operation that allows a client to determine the real-time status of pending and completed transfer requests. In addition to the standard EPP command elements, the <transfer> command MUST contain an "op" attribute with value "query", and a <domain:transfer> element that identifies the domain namespace. The <domain:transfer> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object to be queried.

- An OPTIONAL <domain:authInfo> element that contains authorization information associated with the domain object or authorization information associated with the domain object's registrant or associated contacts. An OPTIONAL "roid" attribute MUST be used to identify the registrant or contact object if and only if the given authInfo is associated with a registrant or contact object, and

not the domain object itself. 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 <transfer> query command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="query">
C:      <domain:transfer
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw roid="JD1234-REP">2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:transfer>
C:    </transfer>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <transfer> query command has been processed successfully, the EPP <resData> element MUST contain a child <domain:trnData> element that identifies the domain namespace. The <domain:trnData> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object.

- A <domain:trStatus> element that contains the state of the most recent transfer request.

- A <domain:reID> element that contains the identifier of the client that requested the object transfer.

- A <domain:reDate> element that contains the date and time that the transfer was requested.

- A <domain:acID> element that contains the identifier of the client that SHOULD act upon a PENDING transfer request. For all other status types, the value identifies the client that took the indicated action.

- A <domain:acDate> element that contains the date and time of a required or completed response. For a PENDING request, the value identifies the date and time by which a response is required

before an automated response action will be taken by the server. For all other status types, the value identifies the date and time when the request was completed.

- An OPTIONAL <domain:exDate> element that contains the end of the domain object's validity period if the <transfer> command caused or causes a change in the validity period.

Example <transfer> query response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:trnData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:trStatus>pending</domain:trStatus>
S:        <domain:reID>ClientX</domain:reID>
S:        <domain:reDate>2000-06-06T22:00:00.0Z</domain:reDate>
S:        <domain:acID>ClientY</domain:acID>
S:        <domain:acDate>2000-06-11T22:00:00.0Z</domain:acDate>
S:        <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
S:      </domain:trnData>
S:    </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 <transfer> query command cannot be processed for any reason.

3.2. EPP Transform Commands

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

Transform commands are typically processed and completed in real time. Server operators MAY receive and process transform commands but defer completing the requested action if human or third-party review is required before the requested action can be completed. In such situations the server MUST return a 1001 response code to the client to note that the command has been received and processed but that the requested action is pending. The server MUST also manage the status of the object that is the subject of the command to reflect the initiation and completion of the requested action. Once the action has been completed, all clients involved in the transaction MUST be notified using a service message that the action has been completed and that the status of the object has changed. Other notification methods MAY be used in addition to the required service message.

Server operators SHOULD confirm that a client is authorized to perform a transform command on a given object. Any attempt to transform an object by an unauthorized client MUST be rejected, and the server MUST return a 2201 response code to the client to note that the client lacks privileges to execute the requested command.

3.2.1. EPP <create> Command

The EPP <create> command provides a transform operation that allows a client to create a domain object. In addition to the standard EPP command elements, the <create> command MUST contain a <domain:create> element that identifies the domain namespace. The <domain:create> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object to be created.

- An OPTIONAL <domain:period> element that contains the initial registration period of the domain object. A server MAY define a default initial registration period if not specified by the client.

- An OPTIONAL <domain:ns> element that contains the fully qualified names of the delegated host objects or host attributes (name servers) associated with the domain object to provide resolution services for the domain; see Section 1.1 for a description of the elements used to specify host objects or host attributes. A host object MUST be known to the server before the host object can be associated with a domain object.

- An OPTIONAL <domain:registrant> element that contains the identifier for the human or organizational social information (contact) object to be associated with the domain object as the

object registrant. This object identifier MUST be known to the server before the contact object can be associated with the domain object. The EPP mapping for contact objects is described in [RFC5733].

- Zero or more OPTIONAL <domain:contact> elements that contain the identifiers for other contact objects to be associated with the domain object. Contact object identifiers MUST be known to the server before the contact object can be associated with the domain object.

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

Example <create> command:

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

When a <create> command has been processed successfully, the EPP <resData> element MUST contain a child <domain:creData> element that identifies the domain namespace. The <domain:creData> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object.

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

- An OPTIONAL <domain:exDate> element that contains the date and time identifying the end of the domain object's registration period.

Example <create> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:creData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:creData>
S:    </resData>
S:    <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.

3.2.2. EPP <delete> Command

The EPP <delete> command provides a transform operation that allows a client to delete a domain object. In addition to the standard EPP command elements, the <delete> command MUST contain a <domain:delete> element that identifies the domain namespace. The <domain:delete> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object to be deleted.

A domain object SHOULD NOT be deleted if subordinate host objects are associated with the domain object. For example, if domain "example.com" exists and host object "ns1.example.com" also exists, then domain "example.com" SHOULD NOT be deleted until host "ns1.example.com" has either been deleted or renamed to exist in a different superordinate domain. A server SHOULD notify clients that object relationships exist by sending a 2305 error response code when a <delete> command is attempted and fails due to existing object relationships. Delegated and subordinate host objects associated with a domain object can be determined using the <info> query command for the domain object.

Example <delete> command:

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

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

Example <delete> response:

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

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

3.2.3. EPP <renew> Command

The EPP <renew> command provides a transform operation that allows a client to extend the validity period of a domain object. In addition to the standard EPP command elements, the <renew> command MUST contain a <domain:renew> element that identifies the domain namespace. The <domain:renew> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object whose validity period is to be extended.

- A <domain:curExpDate> element that contains the date on which the current validity period ends. This value ensures that repeated <renew> commands do not result in multiple, unanticipated successful renewals.

- An OPTIONAL <domain:period> element that contains the number of units to be added to the registration period of the domain object. The number of units available MAY be subject to limits imposed by the server.

Example <renew> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <renew>
C:      <domain:renew
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:curExpDate>2000-04-03</domain:curExpDate>
C:        <domain:period unit="y">5</domain:period>
C:      </domain:renew>
C:    </renew>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <renew> command has been processed successfully, the EPP <resData> element MUST contain a child <domain:renData> element that identifies the domain namespace. The <domain:renData> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object.

- An OPTIONAL <domain:exDate> element that contains the date and time identifying the end of the domain object's registration period.

Example <renew> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:renData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:renData>
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 <renew> command cannot be processed for any reason.

3.2.4. EPP <transfer> Command

The EPP <transfer> command provides a transform operation that allows a client to manage requests to transfer the sponsorship of a domain object. In addition to the standard EPP command elements, the <transfer> command MUST contain a <domain:transfer> element that identifies the domain namespace. The <domain:transfer> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object for which a transfer request is to be created, approved, rejected, or cancelled.

- An OPTIONAL <domain:period> element that contains the number of units to be added to the registration period of the domain object at completion of the transfer process. This element can only be used when a transfer is requested, and it MUST be ignored if used otherwise. The number of units available MAY be subject to limits imposed by the server.

- A <domain:authInfo> element that contains authorization information associated with the domain object or authorization information associated with the domain object's registrant or associated contacts. An OPTIONAL "roid" attribute MUST be used to identify the registrant or contact object if and only if the given authInfo is associated with a registrant or contact object, and not the domain object itself.

Every EPP <transfer> command MUST contain an "op" attribute that identifies the transfer operation to be performed. Valid values, definitions, and authorizations for all attribute values are defined in [RFC5730].

Transfer of a domain object MUST implicitly transfer all host objects that are subordinate to the domain object. For example, if domain object "example.com" is transferred and host object "ns1.example.com" exists, the host object MUST be transferred as part of the "example.com" transfer process. Host objects that are subject to transfer when transferring a domain object are listed in the response to an EPP <info> command performed on the domain object.

Example <transfer> request command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      <domain:transfer
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:period unit="y">1</domain:period>
C:        <domain:authInfo>
C:          <domain:pw roid="JD1234-REP">2fooBAR</domain:pw>
C:        </domain:authInfo>
C:      </domain:transfer>
C:    </transfer>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

When a <transfer> command has been processed successfully, the EPP <resData> element MUST contain a child <domain:trnData> element that identifies the domain namespace. The <domain:trnData> element contains the same child elements defined for a transfer query response.

Example <transfer> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <domain:trnData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:trStatus>pending</domain:trStatus>
S:        <domain:reID>ClientX</domain:reID>
S:        <domain:reDate>2000-06-08T22:00:00.0Z</domain:reDate>
S:        <domain:acID>ClientY</domain:acID>
S:        <domain:acDate>2000-06-13T22:00:00.0Z</domain:acDate>
S:        <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
S:      </domain:trnData>
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 <transfer> command can not be processed for any reason.

3.2.5. EPP <update> Command

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

- A <domain:name> element that contains the fully qualified name of the domain object to be updated.

- An OPTIONAL <domain:add> element that contains attribute values to be added to the object.

- An OPTIONAL <domain:rem> element that contains attribute values to be removed from the object.

- An OPTIONAL <domain:chg> element that contains object attribute values to be changed.

At least one <domain:add>, <domain:rem>, or <domain:chg> element MUST be provided if the command is not being extended. All of these elements MAY be omitted if an <update> extension is present. The <domain:add> and <domain:rem> elements contain the following child elements:

- An OPTIONAL <domain:ns> element that contains the fully qualified names of the delegated host objects or host attributes (name servers) associated with the domain object to provide resolution services for the domain; see Section 1.1 for a description of the elements used to specify host objects or host attributes. A host object MUST be known to the server before the host object can be associated with a domain object. If host attributes are used to specify name servers, note that IP address elements are not needed to identify a name server that is being removed. IP address elements can safely be absent or ignored in this situation.

- Zero or more <domain:contact> elements that contain the identifiers for contact objects to be associated with or removed from the domain object. Contact object identifiers MUST be known to the server before the contact object can be associated with the domain object.

- Zero or more <domain:status> elements that contain status values to be applied to or removed from the object. When specifying a value to be removed, only the attribute value is significant; element text is not required to match a value for removal.

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

- A <domain:registrant> element that contains the identifier for the human or organizational social information (contact) object to be associated with the domain object as the object registrant. This object identifier MUST be known to the server before the contact object can be associated with the domain object. An empty element can be used to remove registrant information.

- A <domain:authInfo> element that contains authorization information associated with the domain object. This mapping includes a password-based authentication mechanism, but the schema allows new mechanisms to be defined in new schemas. A <domain: null> element can be used within the <domain:authInfo> element to remove authorization information.

Example <update> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:add>
C:          <domain:ns>
C:            <domain:hostObj>ns2.example.com</domain:hostObj>
C:          </domain:ns>
C:          <domain:contact type="tech">mak21</domain:contact>
C:          <domain:status s="clientHold"
C:           lang="en">Payment overdue.</domain:status>
C:        </domain:add>
C:        <domain:rem>
C:          <domain:ns>
C:            <domain:hostObj>ns1.example.com</domain:hostObj>
C:          </domain:ns>
C:          <domain:contact type="tech">sh8013</domain:contact>
C:          <domain:status s="clientUpdateProhibited"/>
C:        </domain:rem>
C:        <domain:chg>
C:          <domain:registrant>sh8013</domain:registrant>
C:          <domain:authInfo>
C:            <domain:pw>2BARfoo</domain:pw>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain: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.

3.3. 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:      <domain:creData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
S:      </domain:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

The status of the domain object after returning this response MUST include "pendingCreate". The server operator reviews the request offline, and informs the client of the outcome of the review 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 request.

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 <domain: panData> element that identifies the domain namespace. The <domain: panData> element contains the following child elements:

- A <domain:name> element that contains the fully qualified name of the domain object. The <domain:name> element contains a REQUIRED "paResult" attribute. A positive boolean value indicates that the request has been approved and completed. A negative boolean value indicates that the request has been denied and the requested action has not been taken.

- A <domain:paTRID> element that contains the client transaction identifier and server transaction identifier returned with the original response to process the command. The client transaction identifier is OPTIONAL and will only be returned if the client provided an identifier with the original <create> command.

- A <domain: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>1999-04-04T22:01:00.0Z</qDate>
S:      <msg>Pending action completed successfully.</msg>
S:    </msgQ>
S:    <resData>
S:      <domain:panData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name paResult="1">example.com</domain:name>
S:        <domain:paTRID>
S:          <clTRID>ABC-12345</clTRID>
S:          <svTRID>54321-XYZ</svTRID>
```

```
S:        </domain:paTRID>
S:        <domain:paDate>1999-04-04T22:00:00.0Z</domain:paDate>
S:      </domain:panData>
S:    </resData>
S:    <trID>
S:      <clTRID>BCD-23456</clTRID>
S:      <svTRID>65432-WXY</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

4. Formal Syntax

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

```
   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

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

   <schema targetNamespace="urn:ietf:params:xml:ns:domain-1.0"
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
        xmlns:host="urn:ietf:params:xml:ns:host-1.0"
        xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
        xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
        xmlns="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">

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

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

   <!--
   Child elements found in EPP commands.
   -->
   <element name="check" type="domain:mNameType"/>
   <element name="create" type="domain:createType"/>
   <element name="delete" type="domain:sNameType"/>
   <element name="info" type="domain:infoType"/>
   <element name="renew" type="domain:renewType"/>
   <element name="transfer" type="domain:transferType"/>
   <element name="update" type="domain:updateType"/>
   <!--
   Child elements of the <create> command.
   -->
   <complexType name="createType">
    <sequence>
      <element name="name" type="eppcom:labelType"/>
      <element name="period" type="domain:periodType"
       minOccurs="0"/>
      <element name="ns" type="domain:nsType"
```

```
       minOccurs="0"/>
      <element name="registrant" type="eppcom:clIDType"
       minOccurs="0"/>
      <element name="contact" type="domain:contactType"
       minOccurs="0" maxOccurs="unbounded"/>
      <element name="authInfo" type="domain:authInfoType"/>
    </sequence>
  </complexType>

  <complexType name="periodType">
    <simpleContent>
      <extension base="domain:pLimitType">
        <attribute name="unit" type="domain:pUnitType"
         use="required"/>
      </extension>
    </simpleContent>
  </complexType>

  <simpleType name="pLimitType">
    <restriction base="unsignedShort">
      <minInclusive value="1"/>
      <maxInclusive value="99"/>
    </restriction>
  </simpleType>

  <simpleType name="pUnitType">
    <restriction base="token">
      <enumeration value="y"/>
      <enumeration value="m"/>
    </restriction>
  </simpleType>

  <complexType name="nsType">
    <choice>
      <element name="hostObj" type="eppcom:labelType"
       maxOccurs="unbounded"/>
      <element name="hostAttr" type="domain:hostAttrType"
       maxOccurs="unbounded"/>
    </choice>
  </complexType>
  <!--
  Name servers are either host objects or attributes.
  -->

  <complexType name="hostAttrType">
    <sequence>
      <element name="hostName" type="eppcom:labelType"/>
      <element name="hostAddr" type="host:addrType"
```

```
      minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <!--
  If attributes, addresses are optional and follow the
  structure defined in the host mapping.
  -->

  <complexType name="contactType">
   <simpleContent>
     <extension base="eppcom:clIDType">
       <attribute name="type" type="domain:contactAttrType"/>
     </extension>
   </simpleContent>
  </complexType>

  <simpleType name="contactAttrType">
   <restriction base="token">
     <enumeration value="admin"/>
     <enumeration value="billing"/>
     <enumeration value="tech"/>
   </restriction>
  </simpleType>

  <complexType name="authInfoType">
   <choice>
     <element name="pw" type="eppcom:pwAuthInfoType"/>
     <element name="ext" type="eppcom:extAuthInfoType"/>
   </choice>
  </complexType>

  <!--
  Child element of commands that require a single name.
  -->
  <complexType name="sNameType">
   <sequence>
     <element name="name" type="eppcom:labelType"/>
   </sequence>
  </complexType>
  <!--
  Child element of commands that accept multiple names.
  -->
  <complexType name="mNameType">
   <sequence>
     <element name="name" type="eppcom:labelType"
      maxOccurs="unbounded"/>
   </sequence>
  </complexType>
```

```
   <!--
   Child elements of the <info> command.
   -->
   <complexType name="infoType">
    <sequence>
      <element name="name" type="domain:infoNameType"/>
      <element name="authInfo" type="domain:authInfoType"
       minOccurs="0"/>
    </sequence>
   </complexType>

   <complexType name="infoNameType">
    <simpleContent>
      <extension base = "eppcom:labelType">
        <attribute name="hosts" type="domain:hostsType"
         default="all"/>
      </extension>
    </simpleContent>
   </complexType>

   <simpleType name="hostsType">
    <restriction base="token">
      <enumeration value="all"/>
      <enumeration value="del"/>
      <enumeration value="none"/>
      <enumeration value="sub"/>
    </restriction>
   </simpleType>

   <!--
   Child elements of the <renew> command.
   -->
   <complexType name="renewType">
    <sequence>
      <element name="name" type="eppcom:labelType"/>
      <element name="curExpDate" type="date"/>
      <element name="period" type="domain:periodType"
       minOccurs="0"/>
    </sequence>
   </complexType>

   <!--
   Child elements of the <transfer> command.
   -->
   <complexType name="transferType">
    <sequence>
      <element name="name" type="eppcom:labelType"/>
      <element name="period" type="domain:periodType"
```

```
       minOccurs="0"/>
      <element name="authInfo" type="domain:authInfoType"
       minOccurs="0"/>
    </sequence>
  </complexType>

  <!--
  Child elements of the <update> command.
  -->
  <complexType name="updateType">
    <sequence>
      <element name="name" type="eppcom:labelType"/>
      <element name="add" type="domain:addRemType"
       minOccurs="0"/>
      <element name="rem" type="domain:addRemType"
       minOccurs="0"/>
      <element name="chg" type="domain:chgType"
       minOccurs="0"/>
    </sequence>
  </complexType>

  <!--
  Data elements that can be added or removed.
  -->
  <complexType name="addRemType">
    <sequence>
      <element name="ns" type="domain:nsType"
       minOccurs="0"/>
      <element name="contact" type="domain:contactType"
       minOccurs="0" maxOccurs="unbounded"/>
      <element name="status" type="domain:statusType"
       minOccurs="0" maxOccurs="11"/>
    </sequence>
  </complexType>

  <!--
  Data elements that can be changed.
  -->
  <complexType name="chgType">
    <sequence>
      <element name="registrant" type="domain:clIDChgType"
       minOccurs="0"/>
      <element name="authInfo" type="domain:authInfoChgType"
       minOccurs="0"/>
    </sequence>
  </complexType>
```

```
   <!--
   Allow the registrant value to be nullified by changing the
   minLength restriction to "0".
   -->
   <simpleType name="clIDChgType">
    <restriction base="token">
      <minLength value="0"/>
      <maxLength value="16"/>
    </restriction>
   </simpleType>

   <!--
   Allow the authInfo value to be nullified by including an
   empty element within the choice.
   -->
   <complexType name="authInfoChgType">
    <choice>
      <element name="pw" type="eppcom:pwAuthInfoType"/>
      <element name="ext" type="eppcom:extAuthInfoType"/>
      <element name="null"/>
    </choice>
   </complexType>

   <!--
   Child response elements.
   -->
   <element name="chkData" type="domain:chkDataType"/>
   <element name="creData" type="domain:creDataType"/>
   <element name="infData" type="domain:infDataType"/>
   <element name="panData" type="domain:panDataType"/>
   <element name="renData" type="domain:renDataType"/>
   <element name="trnData" type="domain:trnDataType"/>

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

   <complexType name="checkType">
    <sequence>
      <element name="name" type="domain: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"/>
      </extension>
    </simpleContent>
   </complexType>

   <!--
   <create> response elements.
   -->
   <complexType name="creDataType">
    <sequence>
      <element name="name" type="eppcom:labelType"/>
      <element name="crDate" type="dateTime"/>
      <element name="exDate" type="dateTime"
       minOccurs="0"/>
    </sequence>
   </complexType>

   <!--
   <info> response elements.
   -->

   <complexType name="infDataType">
    <sequence>
      <element name="name" type="eppcom:labelType"/>
      <element name="roid" type="eppcom:roidType"/>
      <element name="status" type="domain:statusType"
       minOccurs="0" maxOccurs="11"/>
      <element name="registrant" type="eppcom:clIDType"
       minOccurs="0"/>
      <element name="contact" type="domain:contactType"
       minOccurs="0" maxOccurs="unbounded"/>
      <element name="ns" type="domain:nsType"
       minOccurs="0"/>
      <element name="host" type="eppcom:labelType"
       minOccurs="0" maxOccurs="unbounded"/>
      <element name="clID" type="eppcom:clIDType"/>
      <element name="crID" type="eppcom:clIDType"
       minOccurs="0"/>
      <element name="crDate" type="dateTime"
       minOccurs="0"/>
      <element name="upID" type="eppcom:clIDType"
```

```
       minOccurs="0"/>
      <element name="upDate" type="dateTime"
       minOccurs="0"/>
      <element name="exDate" type="dateTime"
       minOccurs="0"/>
      <element name="trDate" type="dateTime"
       minOccurs="0"/>
      <element name="authInfo" type="domain:authInfoType"
       minOccurs="0"/>
    </sequence>
  </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="domain:statusValueType"
         use="required"/>
        <attribute name="lang" type="language"
         default="en"/>
      </extension>
    </simpleContent>
  </complexType>

  <simpleType name="statusValueType">
    <restriction base="token">
      <enumeration value="clientDeleteProhibited"/>
      <enumeration value="clientHold"/>
      <enumeration value="clientRenewProhibited"/>
      <enumeration value="clientTransferProhibited"/>
      <enumeration value="clientUpdateProhibited"/>
      <enumeration value="inactive"/>
      <enumeration value="ok"/>
      <enumeration value="pendingCreate"/>
      <enumeration value="pendingDelete"/>
      <enumeration value="pendingRenew"/>
      <enumeration value="pendingTransfer"/>
      <enumeration value="pendingUpdate"/>
      <enumeration value="serverDeleteProhibited"/>
      <enumeration value="serverHold"/>
      <enumeration value="serverRenewProhibited"/>
      <enumeration value="serverTransferProhibited"/>
      <enumeration value="serverUpdateProhibited"/>
    </restriction>
```

```
   </simpleType>

   <!--
   Pending action notification response elements.
   -->
   <complexType name="panDataType">
    <sequence>
      <element name="name" type="domain:paNameType"/>
      <element name="paTRID" type="epp:trIDType"/>
      <element name="paDate" type="dateTime"/>
    </sequence>
   </complexType>

   <complexType name="paNameType">
    <simpleContent>
      <extension base="eppcom:labelType">
        <attribute name="paResult" type="boolean"
         use="required"/>
      </extension>
    </simpleContent>
   </complexType>

   <!--
   <renew> response elements.
   -->
   <complexType name="renDataType">
   <sequence>
    <element name="name" type="eppcom:labelType"/>
    <element name="exDate" type="dateTime"
     minOccurs="0"/>
   </sequence>
   </complexType>

   <!--
   <transfer> response elements.
   -->
   <complexType name="trnDataType">
   <sequence>
    <element name="name" type="eppcom:labelType"/>
    <element name="trStatus" type="eppcom:trStatusType"/>
    <element name="reID" type="eppcom:clIDType"/>
    <element name="reDate" type="dateTime"/>
    <element name="acID" type="eppcom:clIDType"/>
    <element name="acDate" type="dateTime"/>
    <element name="exDate" type="dateTime"
     minOccurs="0"/>
   </sequence>
   </complexType>
```

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

5. 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 [RFC2781]. Though XML includes provisions to identify and use other character encodings through use of an "encoding" attribute in an <?xml?> declaration, use of UTF-8 is RECOMMENDED in environments where parser encoding support incompatibility exists.

All date-time values presented via EPP MUST be expressed in Universal Coordinated Time using the Gregorian calendar. XML Schema allows use of time zone identifiers to indicate offsets from the zero meridian, but this option MUST NOT be used with EPP. 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.

This document requires domain and host name syntax as specified in [RFC0952] as updated by [RFC1123]. At the time of this writing, RFC 3490 [RFC3490] describes a standard to use certain ASCII name labels to represent non-ASCII name labels. These conformance requirements might change as a result of progressing work in developing standards for internationalized domain names.

6. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688]. Two URI assignments have been registered by the IANA.

Registration request for the domain namespace:

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

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

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

Registration request for the domain XML schema:

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

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

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

7. Security Considerations

Authorization information as described in Section 2.6 is REQUIRED to create a domain object. This information is used in some query and transfer operations as an additional means of determining client authorization to perform the command. Failure to protect authorization information from inadvertent disclosure can result in unauthorized transfer operations and unauthorized information release. Both client and server MUST ensure that authorization information is stored and exchanged with high-grade encryption mechanisms to provide privacy services.

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.

8. Acknowledgements

RFC 3731 is a product of the PROVREG working group, which suggested improvements and provided many invaluable comments. The author wishes to acknowledge the efforts of WG chairs Edward Lewis and Jaap Akkerhuis for their process and editorial contributions. RFC 4931 and this document are individual submissions, based on the work done in RFC 3731.

Specific suggestions that have been incorporated into this document were provided by Joe Abley, Chris Bason, Eric Brunner-Williams, Jordyn Buchanan, Dave Crocker, Ayesha Damaraju, Anthony Eden, Sheer El-Showk, Klaus Malorny, Dan Manley, Michael Mealling, Patrick Mevzek, Asbjorn Steira, Bruce Tonkin, and Rick Wesson.

9. References

9.1. Normative References

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.

[RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, August 2009.

[RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, August 2009.

[W3C.REC-xml-20040204]
Sperberg-McQueen, C., Maler, E., Yergeau, F., Paoli, J., and T. Bray, "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]
Maloney, M., Thompson, H., Mendelsohn, N., and D. Beech, "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]
Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

9.2. Informative References

[RFC2781] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC4931] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", RFC 4931, May 2007.

Appendix A. Changes from RFC 4931

1. Changed "This document obsoletes RFC 3731" to "This document obsoletes RFC 4931".

2. Replaced references to RFC 3731 with references to 4931.

3. Replaced references to RFC 4930 with references to 5730.

4. Replaced references to RFC 4932 with references to 5732.

5. Replaced references to RFC 4933 with references to 5733.

6. Updated description of inactive status in Section 2.3.

7. Fixed example host names in the Section 1.1 and Section 3.2.1 examples.

8. Changed "but such methods SHOULD NOT be used" to "but such methods should not be used" in Section 2.7.

9. Added "Other notification methods MAY be used in addition to the required service message" in Section 3.2.

10. Added 2201 response code text in Section 3.2.

11. Added BSD license text to XML schema section.

Author's Address

Scott Hollenbeck
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
US

EMail: shollenbeck@verisign.com