Internet Engineering Task Force
Internet-Draft
May, 2002

Eric Brunner-Williams
Editor

Expires December 2002

Extensible Provisioning Protocol Container <draft-ietf-provreg-epp-container-02.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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.''

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This memo defines an extension to EPP objects. The extension supports hierarchy and inheritance amongst EPP objects.

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].

The convention "TRY" is used for identifiers which incorporate the mnemonic acronymn for a registry, or the word "REGISTRY" is spelled out. The convention "ANT" is used for identifiers which incorporate the mnemonic acronymn for a registrar, or the word "REGISTRAR" is spelled out. The case-independent and optionally hypenated example "FooCorp" is used throughout as an example.

Table of Contents

1. Introduction

The Extensible Provisioning Protocol [2] defines generic object management operations and an extensible framework that maps protocol operations to objects stored in a shared central repository.

This memo documents how these generic object management operations are extended to hierarchies of objects, called container objects or

containers.

This memo is being discussed on the "ietf-provreg" mailing list. To join the list, send a message to <majordomo@cafax.se> with the words "subscribe ietf-provreg" in the body of the message. There is a web site for the list archives at http://www.cafax.se/ietf-provreg.

2. Containers

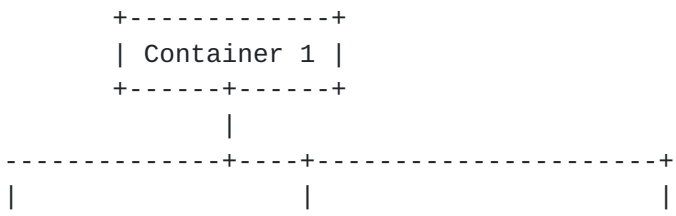
A container object is a collection of objects defined by a registry. A container object may contain any number of these defined objects, and may also contain other containers in a nested fashion. Containers simplify administration and management of associated objects.

Each object in a collection of objects in a container maintains a parent-child relationship with the container. The collection of containers and the objects are managed in a tree/forest fashion with one root container node. In a Container tree, non-leaf nodes are container objects, leaf nodes are other objects. Each object in the collection of a container will maintain a link to its parent container. A container without a parent container is the root node of a container tree.

Each child container inherits properties from the parent container. Additionally, each container derives properties from its directly associated leaf child object nodes, which override any mutual exclusive inherited properties. For root containers the properties are the properties of its directly associated leaf child object nodes alone.

This inheritance of properties among associated containers restricts containers to a single parent, or no parent in the case of the root container. Non-leaf nodes (containers) may maintain references to leaf-notes without restriction. Restated, a container can only have at most one parent container, and zero or more child containers.

Mutual exclusion of objects contained in registries and references to objects in multiple containers may be defined by the registry. Example Container:



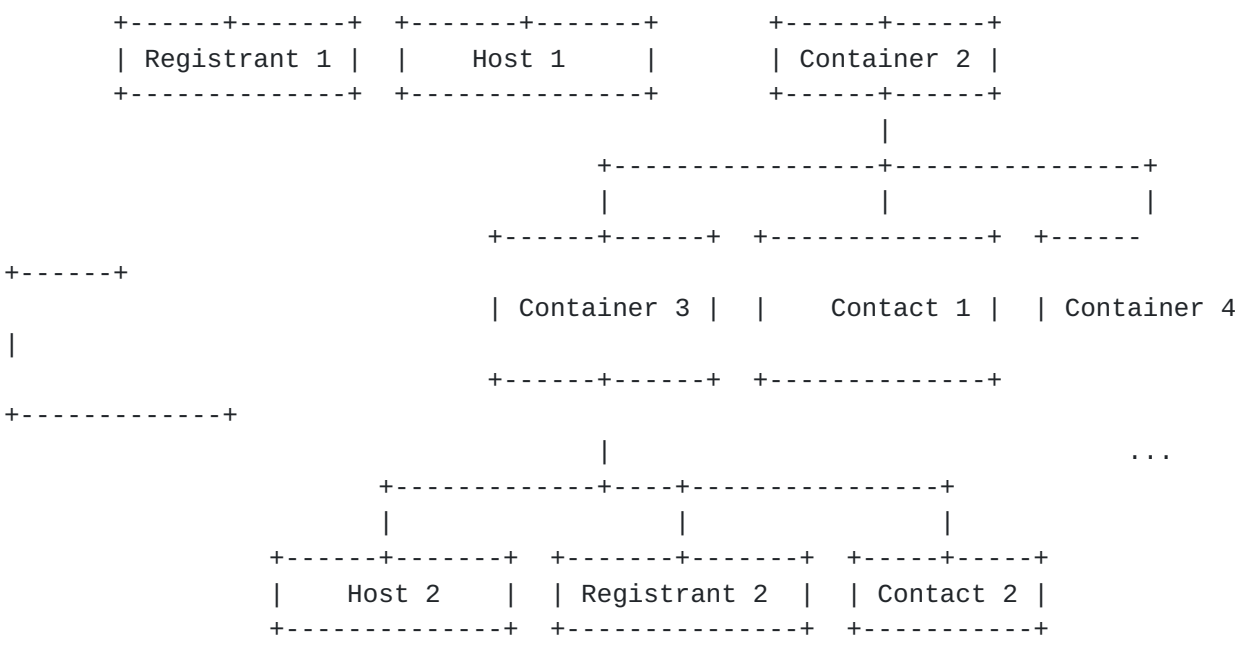


Figure 1: An Example of Container Tree

Container 1 has the properties:

"Registrant 1"
"Host 1"

Container 2 inherits the Container 1 properties and its child objects.

"Registrant 1"
"Host 1"
"Contact 1"

The derivation of Container 3, assuming the registry policy of type Container 3 is Unique Registrant, evaluates as:

"Registrant 2"
"Contact 1"
"Contact 2"
"Host 1"
"Host 2"

This example shows property derivation with a local policy (registrant 2 preference).

2.1 EPP Command Summary for Containers

2.1.1 Container Query Commands

```
   check
      check on a container
```

info
query the collection of objects using a container object

transfer query
query the status of a container transfer command

2.1.2 Container Transformation Commands

create
create a container

delete
delete a container

update
update a container (add/remove objects)

update
update a collection of associated objects by updating properties of a container object

delete
delete a container with an option to delete all associated objects.

transfer
transfer a container with an option to transfer all associated objects.

transfer
transfer a container with an option to transfer all directly associated leaf child objects and child containers.

transfer
transfer a container with an option to transfer both linked objects and directly associated leaf child objects and child containers.

2.2 EPP Command for Managing Objects with Containers

A detailed description of the EPP syntax and semantics can be found in [2]. The extended command mappings summarized here are described in section 5.

2.3 Container Templates

Each container can be subjected to a set of registry policies by binding a container to a template supported by Registry. Templates

are well-documented Registry policies that determine the functions and objects supported in a container.

Example Templates

Registrant Template

supports child objects:
o registrant
o contacts (multiple type associations: admin, tech, billing),
o hosts (1-13),
o domain and/or registrant containers

supports functions/commands:
o create a registrant container with supported child objects,
o create associated objects (domains, registrant container) using the registrant container,
o update container add/delete
  registrant,
  contacts (admin, tech, billing),
  hosts (1-13),
  container
o update container with an option to update associated objects
o delete container
o delete container with an option to delete all associated objects
o update add/delete other registrant containers, domain containers
o transfer a container to a different registrar
o transfer a container to a different registrar along with associated objects

Domain Template

support child objects:
o domains,

supports functions/commands to:
o create a domain container with supported child objects
o create a domain container under a registrant template and supported child objects
o create a domain container using a registrant template and supported child objects
o update container add/delete domains or other domain container
o transfer a container to a different registrar
o transfer a container to a different registrar along with child

objects
- o delete a container
- o delete a container with an option to delete all child objects

Registrar Account Template

support child objects:
- o contacts,
- o hosts (1-13),
- o registrant containers

supports functions/commands to:
- o create a registrant account container with supported child objects.
- o create associated objects (registrant containers)
- o update container add/delete any of the child objects
- o delete container
- o delete container with an option to delete all associated objects

Registrant Account Template

A Registrant Account Template can be used to group various objects or containers into a single aggregation. However, objects or containers in the registry should only belong to at most one container created with the Registrant Account Template.

support child objects:
- o domain objects,
- o host objects,
- o contact objects,
- o registrant containers,
- o domain containers

supports functions/commands to:
- o create a registrant account template with supported child objects
- o update container add/delete any supported child objects
- o delete a container
- o delete a container with an option to delete all child objects
- o transfer a container to a different registrar

2.4 Container Attributes

An EPP container object has attributes and associated values that may be viewed and modified by the sponsoring client or the server. This

section describes each attribute type in detail.

id
This element is the unique identifier of the container given by the registrar. The format is alphanumeric with the minimum and maximum length and character set specified in the XML [3] Schema Definition [4], [5].

Example:

```
<id>ANT-CONTAINER-42</id>
```

roid
This element is the unique identifier of the container given by the registry. Container identifiers use the "roidType" syntax described in [2].

Example:

```
<roid>CONTAINER001-TRY</roid>
```

status
This element describes the status of the container. This element may have multiple occurrences.

Example:

```
<status s="ok"/>
<status s="clientTransferProhibited"/>
```

parent
This element is the reference of the parent container object in the container tree by its <id>. This element is optional and if not specified the container is the root node of the tree.

The format is alphanumeric with the minimum and maximum length and character set specified in the XML [3] Schema Definition [4], [5].

Example:

```
<parent>00-CONT-21</parent>
```

template
This optional element is the registry provided template for creating containers. For a given template there are set of Registry defined policies for supported objects in a template and other validations for a container.

The format is alphanumeric with the minimum and maximum length and character set specified in the XML [3] Schema Definition [4], [5].

Example:

```
<template>Template100-Registrant</template>
```

child

This element is the reference to objects in the container. This element may have multiple occurrences. The child element in a container has attribute "obj" and optional "type" to reflect supported object types and optional type of association. The object types include all registry supported objects including containers.

Example:

```
<child object="registrant">foo-corp-corp</child>
<child object="contact" type="admin">foo-corp-admin</child>
```

derived

This element is the reference to objects inherited from ancestor containers. This element may have multiple occurrences. The derived element in a container has attribute "container", "object" and optional "type" to reflect the container from which the object is inherited, object type, and optional type of association. The object types include all registry supported objects.

Example:

```
<derived container="foocorp-container"
      object="registrant">foo-corp-corp
</derived>

<derived container="foocorp-container"
      object="contact"
      type="admin">foo-corp-admin
</derived>
```

linked

This element is the reference to other objects linked to this container. These are the objects created using container properties. Updates made to a container will result in updates to all linked objects. The "linked" element has attribute "type" to reflect the type of the linked object.

Example:

```
      <linked object="registrant">foo-corp-corp</linked>
      <linked object="contact" type="admin">foo-corp-admin</linked>
```

clID
   This element contains the identifier of the sponsoring registrar. The sponsoring registrar client has all the necessary administrative privileges to manage the object.

Example:

```
        <clID>clientXX</clID>
```

crID
   This element contains the reference to the registrar client that created the container.

Example:

```
        <crID>clientXX</crID>
```

crDate
   This element contains the reference to the date the container was created. The format of the time field is UCT extended format of ISO 8601.

Example:

```
     <crDate>2001-07-04:00:00.0Z</crDate>
```

upID
   This element contains the reference to the registrar client that modified the container.

Example:

```
        <upID>clientXX</upID>
```

upDate
   This element contains the reference to the date the container was modified. The format of the time field is UCT extended format of ISO 8601.

Example:

```
     <upDate>2001-07-04:00:00.0Z</upDate>
```

trDate
   This element contains the reference to the date the container was

transferred. The format of the time field is UCT extended format of ISO 8601.

Example:

<trDate>2001-07-04:00:00.0Z</trDate>

authInfo
This element contains information associated with containers to facilitate transfer operations. Authorization information is assigned when a container is created, and it MAY be updated in the future. This specification describes password or certificate-based authorization information, though other mechanisms are possible.

Example:

<authInfo type="pw">2fooBAR</authInfo>

2.5 Status Values

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

Status value Descriptions:

active
Supports updates and usage in transactions.

inactive

Does not support updates or usage in transactions.

pendingTransfer
A transfer request has been received for the container, and completion of the request is pending. Container transformation commands other than <transfer> MUST be rejected while a container is in this state.

pendingDelete
A delete request has been received for the container, but the container has not yet been purged from the server database.

TransferProhibited, clientTransferProhibited
Requests to transfer the container MUST be rejected.

UpdateProhibited, clientUpdateProhibited
Requests to update the container MUST be rejected.

DeleteProhibited, clientDeleteProhibited
Requests to delete the container MUST be rejected.

linked
The container has at least one active association, such as a domain object of another container. Servers SHOULD provide services to determine existing object associations.

ok
This is the nominal status value for an container that has no pending operations or prohibitions.

3. EPP Command Mapping for Containers

A detailed description of the EPP syntax and semantics can be found in [2]. The command mappings described here are specifically for use in provisioning and managing containers via EPP.

3.1 EPP Query Commands

EPP provides two commands to retrieve container information: <check> to determine if a container is known to the server, <info> to retrieve detailed information associated with a container. [Editor: Resolve tri-value issue for container extensions, e.g., <transfer> to retrieve container transfer status information.]

3.1.1 EPP <check> Command

The EPP <check> command is used to determine if a container is known to the server. In addition to the standard EPP command elements, the <check> command MUST contain a <container:check> element that identifies the container namespace and the location of the container schema. The <container:check> element contains the following child elements:

One or more <container:id> elements that contain the server-unique identifier of the containers to be queried.

Example <check> command:

C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>

```
C:<epp xmlns="urn:iana:xml:ns:epp"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
C:    <check>
C:      <container:check xmlns:container="urn:iana:xml:ns:container"
C:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
C:       container-1.0.xsd">
C:        <container:id>CONTAINER-FooCorp-USA</container:id>
C:        <container:id>CONTAINER-FooCorp-Asia</container:id>
C:        <container:id>CONTAINER-FooCorp-Europe</container:id>
C:      </container:check>
C:    </check>
C:    <unspec/>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

When a <check> command has been processed successfully, the EPP <resData> element MUST contain a child <container:chkData> element that identifies the container namespace and the location of the container schema. The <container:chkData> element contains the following child elements:

One or more <container:cd> elements that contain the server-unique identifier for the queried containers and an "x" attribute whose value identifies the object as either "+" for a known object or "-" for an unknown container.

Example <check> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <container:chkData xmlns:container="urn:iana:xml:ns:container-1.0"
S:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
S:       container-1.0.xsd">
S:        <container:cd x="+">CONTAINER-FooCorp-USA</container:cd>
S:        <container:cd x="-">CONTAINER-FooCorp-Asia</container:cd>
S:        <container:cd x="+">CONTAINER-FooCorp-Europe</container:cd>
S:      </container:chkData>
S:    </resData>
```

```
S:    <unspec/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.1.2 EPP <info> Command

The EPP <info> command is used to retrieve information associated with a container. In addition to the standard EPP command elements, the <info> command MUST contain a <container:info> element that identifies the container namespace and the location of the container schema. The <container:info> element contains the following child elements:

A <container:id> element that contains the server-unique identifier of the container to be queried.

Example <info> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
C:    <info>
C:      <container:info xmlns:container="urn:iana:xml:ns:container-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
C:       container-1.0.xsd">
C:        <container:id>CONTAINER-FooCorp-USA</container:id>
C:      </container:info>
C:    </info>
C:    <unspec/>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

When an <info> command has been processed successfully, the EPP <resData> element MUST contain a child <container:infData> element that identifies the container namespace and the location of the container schema. The <container:infData> element contains the following child elements:

A <container:id> element that contains the server-unique identifier of the container.

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

One or more <container:status> elements that describe the status of the container.

An OPTIONAL <container:parent> element that contains the server-unique identifier of the parent container of which the container is a child.

An OPTIONAL <container:template> element that identifies the registry provided template, which is a set of registry defined policies, for maintaining the container.

Zero or more <container:child> elements that specify the server-unique identifiers of objects or containers that are included in the container. Each element MUST include one required "object" attribute for identifying the type of the child, and an OPTIONAL "type" attribute for identifying the sub-type of the child within the specific object type.

If supported by the server, one or more <container:derived> elements specify the server-unique identifiers of objects, excluding containers, that the container inherited from its ancestor containers. Each element MUST inlude one required "container" attribute for identifying the container from which the object is inherited, one required "object" attribute for identifying the type of the child, and an OPTIONAL "type" attribute for identifying the sub-type of the child within the specific object type.

If supported by the server, one or more <container:linked> elements that identifies types (via a REQUIRED "type" attribute) and server-unique identifiers of objects associated with the container, either directly or indirectly, specified by the OPTIONAL "directly" attribute, with either a "true" or "false" value, with "false" as the default.

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

A <container:crID> element that contains the identifier of the client that created the container.

A <container:crDate> element that contains the date and time of the container creation.

A <container:upID> element that contains the identifier of the client that last updated the container. This element MUST NOT be present if the container has never been modified.

A <container:upDate> element that contains the date and time of the most recent container modification. This element MUST NOT be present if the container has never been modified.

A <container:trDate> elements that contains the date and time of the most recent successful container transfer. This element MUST NOT be provided if the container has never been transferred.

A <container:authInfo> element that contains authorization information to be associated with the container. This element MUST NOT be provided if the querying client is not the current sponsoring client.

Example <info> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <container:infData xmlns:container="urn:iana:xml:ns:container-1.0"
S:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
S:       container-1.0.xsd">
S:        <container:id>CONTAINER-FooCorp-USA</container:id>
S:        <container:roid>CONTAINER12345-REGISTRY</container:roid>
S:        <container:status s="linked"/>
S:        <container:status s="clientDeleteProhibited"/>
S:        <container:status s="clientTransferProhibited"/>
S:        <container:parent>CONTAINER-FooCorp</container:id>
S:        <container:template>TEMPLATE-FooCorp</container:template>
S:        <container:child object="container">CONTAINER-FooCorp-Georgia
S:        </container:child>
S:        <container:child object="registrant">CONTACT-FooCorp-USA
S:        </container:child>
S:        <container:child object="contact" type="tech">CONTACT-FooCorp-Tech
S:        </container:child>
S:        <container:child object="host">ns1.foocorp.tld</container:child>
S:        <container:child object="host">ns2.foocorp.tld</container:child>
S:        <container:child object="host">ns3.foocorp.tld</container:child>
S:        <container:derived container="CONTAINER-FooCorp" object="host">
```

```
S:        ns.foocorp.tld</container:derived>
S:        <container:linked object="domain" directly="true">foocorp.tld
S:        </container:linked>
S:        <container:linked object="domain" directly="true">foocorp-ga.tld
S:        </container:linked>
S:        <container:linked object="contact" directly="false">foocorp-va.tld
S:        </container:linked>
S:        <container:clID>ClientY</container:clID>
S:        <container:crID>ClientX</container:crID>
S:        <container:crDate>2001-04-03T22:00:00.0Z</container:crDate>
S:        <container:upID>ClientX</container:upID>
S:        <container:upDate>2001-12-03T09:00:00.0Z</container:upDate>
S:        <container:trDate>2002-04-08T09:00:00.0Z</container:trDate>
S:        <container:authInfo type="pw">2fooBAR</container:authInfo>
S:      </container:infData>
S:    </resData>
S:    <unspec/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.1.3 EPP <transfer> Command

The EPP <transfer> command provides a query operation that allows a client to determine 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 <container:transfer> element that identifies the container namespace and the location of the container schema. The <container:transfer> element MUST contain the following child elements:

A <container:id> element that contains the server-unique identifier of the container to be queried.

Example <transfer> query command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
```

```
C:    <transfer op="query">
C:    <transfer op="query">
C:      <container:transfer xmlns:container="urn:iana:xml:ns:container-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
C:       container-1.0.xsd">
C:        <container:id>CONTAINER-FooCorp-USA</container:id>
C:      </container:transfer>
C:    </transfer>
C:    <unspec/>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

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

A <container:id> element that contains the server-unique identifier for the queried container.

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

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

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

A <container:acID> element that contains the identifier of the client that SHOULD act upon the transfer request.

A <container: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 SHOULD be taken by the server. For all other status types, the value identifies the date and time when the request was completed.

Example <transfer> query response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
```

```
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <container:trnData xmlns:container="urn:iana:xml:ns:container-1.0"
S:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
S:       container-1.0.xsd">
S:        <container:id>CONTAINER-FooCorp-USA</container:id>
S:        <container:trStatus>pending</container:trStatus>
S:        <container:reID>ClientX</container:reID>
S:        <container:reDate>2000-06-06T22:00:00.0Z</container:reDate>
S:        <container:acID>ClientY</container:acID>
S:        <container:acDate>2000-06-11T22:00:00.0Z</container:acDate>
S:      </container:trnData>
S:    </resData>
S:    <unspec/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

An EPP error response MUST be returned if a <transfer> query command can not be processed for any reason.

3.2 EPP Transform Commands

EPP provides four commands to transform container information: <create> to create an instance of a container, <delete> to delete an instance of a container, <transfer> to manage container sponsorship changes, and <update> to change information associated with a container. This document does not define a mapping for the EPP <renew> command.

3.2.1 EPP <create> Command

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

A <container:id> element that contains the desired server-unique identifier for the container to be created.

An OPTIONAL <container:parent> element that contains the server-

unique identifier of the parent container of which the container is a child.

An OPTIONAL <container:template> element that identifies the registry provided template, which is a set of registry defined policies, in maintaining the container.

Zero or more <container:child> elements that specify the server-unique identifiers of objects or containers that are included in the container. Each element MUST include one required "object" attribute for identifying the type of the child, and an OPTIONAL "type" attribute for identifying the sub-type of the child within the specific object type. If an element is a container, its "parent" value MUST NOT have been specified.

A <container:authInfo> element that contains authorization information to be associated with the container.

Example <create> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
C:    <create>
C:      <container:create xmlns:container="urn:iana:xml:ns:container-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
C:       container-1.0.xsd">
C:        <container:id>CONTAINER-FooCorp-USA</container:id>
C:        <container:parent>CONTAINER-FooCorp</container:parent>
C:        <container:template>TEMPLATE-FooCorp</container:template>
S:        <container:child object="container">CONTAINER-FooCorp-Georgia
S:        </container:child>
S:        <container:child object="registrant">CONTACT-FooCorp-USA
S:        </container:child>
S:        <container:child object="contact" type="tech">CONTACT-FooCorp-Tech
S:        </container:child>
S:        <container:child object="host">ns1.foocorp.tld</container:child>
S:        <container:child object="host">ns2.foocorp.tld</container:child>
S:        <container:child object="host">ns3.foocorp.tld</container:child>
C:        <container:authInfo type="pw">2fooBAR</container:authInfo>
C:      </container:create>
C:    </create>
C:    <unspec/>
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 <container:creData> element that identifies the container namespace and the location of the container schema. The <container:creData> element contains the following child elements:

A <container:id> element that contains the server-unique identifier for the created container.

Example <create> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <container:creData xmlns:container="urn:iana:xml:ns:container-1.0"
S:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
S:       container-1.0.xsd">
S:        <container:id>CONTAINER-FooCorp-USA</container:id>
S:      </container:creData>
S:    </resData>
S:    <unspec/>
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 can not 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 container. In addition to the standard EPP command elements, the <delete> command MUST contain a <container:delete> element that identifies the container namespace and the location of the container schema. The <container:delete> element MUST contain the following child elements:

A <container:id> element that contains the server-unique identifier of the container to be deleted.

An OPTIONAL <container:option> elements that indicates if all objects associated with the container should be deleted or the associations to be broken. This element MAY take one of the following possible values, with "none" as the default:

"none" for taking no actions on associated objects

"delete" for deleting all associated objects

"break" for breaking the association relationship of all associated objects without deleting them.

A <container:authInfo> element that contains authorization information to be associated with the container.

A container object SHOULD NOT be deleted if it is associated with other known objects or containers. An associated container SHOULD NOT be deleted until associations with other known objects or containers have been broken.

Example <delete> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
C:    <delete>
C:      <container:delete xmlns:container="urn:iana:xml:ns:container-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
C:       container-1.0.xsd">
C:        <container:id>CONTAINER-FooCorp-USA</container:id>
C:        <container:option>break</container:option>
C:      </container:delete>
C:    </delete>
C:    <unspec/>
C:    <clTRID>ABC-12346</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:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <unspec/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.2.3 EPP <renew> Command

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

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 container. In addition to the standard EPP command elements, the <transfer> command MUST contain a <container:transfer> element that identifies the container namespace and the location of the container schema. The <container:transfer> element contains the following child elements:

A <container:id> element that contains the server-unique identifier of the container for which a transfer request is to be created, approved, rejected, or cancelled.

An OPTIONAL <container:option> element that contains instruction if and how the objects associated with container are transferred. This element is REQUIRED only when a transfer is requested, and it MUST be ignored if used otherwise. The element takes one of the following possible values, with "none" as default:

"none" for no actions to be taken for all associated objects

"linked" for transferring all directly associated objects

"child" for transferring all child objects or containers

"all" for transferring both all associated objects, directly or

indirectly, and all child objects or containers.

If there are any objects or containers with a status that prohibites transfer operation, the transfer operation as a whole will fail.

A <container:authInfo> element that contains authorization information associated with the container. This element is REQUIRED only when a transfer is requested, and it MUST be ignored if used otherwise.

Every EPP <transfer> command MUST contain an "op" attribute that identifies the transfer operation to be performed as defined in [2].

The EPP <transfer> command MAY be restricted to certain containers, based on the container templates or registry policies.

Example <transfer> request command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
C:    <transfer op="request">
C:      <container:transfer xmlns:container="urn:iana:xml:ns:container-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
C:       container-1.0.xsd">
C:        <container:id>CONTAINER-FooCorp-USA</container:id>
C:        <container:authInfo type="pw">2fooBAR</container:authInfo>
C:      </container:transfer>
C:    </transfer>
C:    <unspec/>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

When a <transfer> command has been processed successfully, the EPP <resData> element MUST contain a child <container:trnData> element that identifies the container namespace and the location of the container schema. The <container: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:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <container:trnData xmlns:container="urn:iana:xml:ns:container-1.0"
S:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
S:       container-1.0.xsd">
S:        <container:id>CONTAINER-FooCorp-USA</container:id>
S:        <container:trStatus>pending</container:trStatus>
S:        <container:reID>ClientX</container:reID>
S:        <container:reDate>2000-06-08T22:00:00.0Z</container:reDate>
S:        <container:acID>ClientY</container:acID>
S:        <container:acDate>2000-06-13T22:00:00.0Z</container:acDate>
S:      </container:trnData>
S:    </resData>
S:    <unspec/>
S:    <trID>
S:      <clTRID>ABC-12346</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 container. In addition to the standard EPP command elements, the <update> command MUST contain a <container:update> element that identifies the container namespace and the location of the container schema. The <container:update> element contains the following child elements:

A <container:id> element that contains the server-unique identifier of the container object to be updated.

An OPTIONAL <container:add> element that contains attribute values to be added to the container

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

An OPTIONAL <container:chg> element that contains container attribute values to be changed.

At least one <container:add>, <container:rem>, or <container:chg> element MUST be provided. The <container:add> and <container:rem> elements contain the following child elements.
At least one child element MUST be present:

Zero or more <container:child> elements that specify the server-unique identifiers of objects or containers that are included in the container. Each element MUST include one required "object" attribute for identifying the type of the child, and an OPTIONAL "type" attribute for identifying the sub-type of the child within the specific object type. The removal of any children of a container MUST ensure data integrity of all objects associated with the container, directly or indirectly.

Zero or more <container:status> elements that contain status values to be associated with or removed from the container. 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 <container:chg> element contains the following OPTIONAL child elements. At least one child element MUST be present:

A <container:parent> element that contains the server-unique identifier of the new parent container of which the container is a child. The change of the container child-parent relationship MUST ensure data integrity of all objects associated with the container, directly or indirectly.

If supported by the server, an <container:template> element that identifies the new registry provided template, which is a set of registry defined policies, in maintaining the container.

A <container:authInfo> element that contains authorization information associated with the container object.

Example <update> command:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
C:    <update>
C:      <container:update xmlns:container="urn:iana:xml:ns:container-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:container-1.0
C:       container-1.0.xsd">
C:        <container:id>CONTAINER-FooCorp-USA</container:id>
C:        <container:add>
C:          <container:status s="clientDeleteProhibited"/>
```

```
C:          <container:child object="host">ns4.foocorp.tld</container:child>
C:        </container:add>
C:        <container:chg>
C:          <container:parent>CONTAINER-FooCorp-All</container:parent>
C:        </container:chg>
C:      </container:update>
C:    </update>
C:    <unspec/>
C:    <clTRID>ABC-12346</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:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <unspec/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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

3.3 Formal Syntax for Container

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.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:iana:xml:ns:container-1.0"
        xmlns:container="urn:iana:xml:ns:container-1.0"
        xmlns:epp="urn:iana:xml:ns:epp-1.0"
        xmlns:eppcom="urn:iana:xml:ns:eppcom-1.0"
```

```
          xmlns="http://www.w3.org/2001/XMLSchema"
          elementFormDefault="qualified">
  <!--
  Import common element types.
  -->
    <import namespace="urn:iana:xml:ns:eppcom-1.0"
            schemaLocation="eppcom-1.0.xsd"/>
    <import namespace="urn:iana:xml:ns:epp-1.0"
            schemaLocation="epp-1.0.xsd"/>
    <annotation>
      <documentation>
        Extensible Provisioning Protocol v1.0
        container provisioning schema.
      </documentation>
    </annotation>
  <!--
  Types used within an EPP greeting.
  -->
    <element name="svc"/>
  <!--
  Container identifier type
  -->
    <simpleType name="containerIDType">
      <restriction base="token">
        <minLength value="3"/>
        <maxLength value="64"/>
      </restriction>
    </simpleType>
  <!--
  Template identifier type
  -->
    <simpleType name="templateIDType">
      <restriction base="token">
        <minLength value="3"/>
        <maxLength value="64"/>
      </restriction>
    </simpleType>
  <!--
  Child/Derived/Linked type
  -->
    <complexType name="childType">
      <simpleContent>
        <extension base="normalizedString">
          <attribute name="object" type="container:objectType"/>
          <attribute name="type" type="token" use="optional"/>
        </extension>
      </simpleContent>
    </complexType>
```

```
    <complexType name="derivedType">
      <simpleContent>
        <extension base="normalizedString">
          <attribute name="container" type="container:containerIDType"/>
          <attribute name="object" type="container:objectType"/>
          <attribute name="type" type="token" use="optional"/>
        </extension>
      </simpleContent>
    </complexType>
    <complexType name="linkedType">
      <simpleContent>
        <extension base="normalizedString">
          <attribute name="object" type="container:objectType"/>
          <attribute name="directly" type="boolean"
          use="optional" default="true"/>
        </extension>
      </simpleContent>
    </complexType>
    <simpleType name="objectType">
      <restriction base="token">
        <enumeration value="container"/>
        <enumeration value="registrant"/>
        <enumeration value="contact"/>
        <enumeration value="domain"/>
        <enumeration value="host"/>
      </restriction>
    </simpleType>
  <!--
  Child elements found in EPP commands.
  -->
    <element name="check" type="container:mIDType"/>
    <element name="create" type="container:createType"/>
    <element name="delete" type="container:deleteType"/>
    <element name="info" type="container:sIDType"/>
    <element name="transfer" type="container:transferType"/>
    <element name="update" type="container:updateType"/>
  <!--
  Child elements of the <create>, <delete> and <info> commands.
  -->
    <complexType name="createType">
      <sequence>
        <element name="id" type="container:containerIDType"/>
        <element name="parent" type="container:containerIDType"
         minOccurs="0"/>
        <element name="template" type="container:templateIDType"
         minOccurs="0"/>
        <element name="child" type="container:childType"/>
        <element name="authInfo" type="eppcom:authInfoType"/>
```

```
      </sequence>
    </complexType>
    <complexType name="deleteType">
      <sequence>
        <element name="id" type="container:containerIDType"/>
        <element name="option" type="container:deleteOptType"
         minOccurs="0"/>
        <element name="authInfo" type="eppcom:authInfoType"/>
      </sequence>
    </complexType>
    <simpleType name="deleteOptType">
      <restriction base="token">
        <enumeration value="none"/>
        <enumeration value="delete"/>
        <enumeration value="break"/>
      </restriction>
    </simpleType>
  <!--
  Child element of commands that require only an identifier.
  -->
    <complexType name="sIDType">
      <sequence>
        <element name="id" type="container:containerIDType"/>
      </sequence>
    </complexType>
  <!--
  Child element of commands that accept multiple identifiers.
  -->
    <complexType name="mIDType">
      <sequence>
        <element name="id" type="container:containerIDType"
         maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  <!--
  Child elements of the <transfer> command.
  -->
    <complexType name="transferType">
      <sequence>
        <element name="id" type="container:containerIDType"/>
        <element name="option" type="container:transferOptType"
         minOccurs="0"/>
        <element name="authInfo" type="eppcom:authInfoType"
         minOccurs="0"/>
      </sequence>
    </complexType>
    <simpleType name="transferOptType">
      <restriction base="token">
```

```
        <enumeration value="none"/>
        <enumeration value="linked"/>
        <enumeration value="child"/>
        <enumeration value="all"/>
      </restriction>
    </simpleType>
  <!--
  Child elements of the <update> command.
  -->
    <complexType name="updateType">
      <sequence>
        <element name="id" type="container:containerIDType"/>
        <choice minOccurs="1" maxOccurs="3">
          <element name="add" type="container:addRemType"
           minOccurs="0"/>
          <element name="rem" type="container:addRemType"
           minOccurs="0"/>
          <element name="chg" type="container:chgType"
           minOccurs="0"/>
         </choice>
      </sequence>
    </complexType>
  <!--
  Data elements that can be added or removed.
  -->
    <complexType name="addRemType">
      <choice minOccurs="1" maxOccurs="2">
        <element name="status" type="container:statusType"
         minOccurs="0" maxOccurs="9"/>
        <element name="child" type="container:childType"
         minOccurs="0" maxOccurs="unbounded"/>
      </choice>
    </complexType>
  <!--
  Data elements that can be changed.
  -->
    <complexType name="chgType">
      <choice minOccurs="1" maxOccurs="3">
        <element name="parent" type="container:containerIDType"
         minOccurs="0"/>
        <element name="template" type="container:templateIDType"
         minOccurs="0"/>
        <element name="authInfo" type="eppcom:authInfoType"
         minOccurs="0"/>
      </choice>
    </complexType>
  <!--
  Child response elements.
```

```
   -->
     <element name="chkData" type="container:chkDataType"/>
     <element name="creData" type="container:creDataType"/>
     <element name="infData" type="container:infDataType"/>
     <element name="trnData" type="container:trnDataType"/>
   <!--
   <check> response elements.
   -->
     <complexType name="chkDataType">
       <sequence>
         <element name="cd" type="container:checkType"
          maxOccurs="unbounded"/>
       </sequence>
     </complexType>
     <complexType name="checkType">
       <simpleContent>
         <extension base = "container:containerIDType">
           <attribute name="x" type="eppcom:checkType"
            use="required"/>
         </extension>
       </simpleContent>
     </complexType>
   <!--
   <create> response elements.
   -->
     <complexType name="creDataType">
       <sequence>
         <element name="id" type="container:containerIDType"/>
       </sequence>
     </complexType>
   <!--
   <info> response elements.
   -->
     <complexType name="infDataType">
       <sequence>
         <element name="id" type="container:containerIDType"/>
         <element name="roid" type="eppcom:roidType"/>
         <element name="status" type="container:statusType"
          maxOccurs="8"/>
         <element name="parent" type="container:containerIDType"
          minOccurs="0"/>
         <element name="template" type="container:templateIDType"
          minOccurs="0"/>
         <element name="child" type="container:childType"
          minOccurs="0" maxOccurs="unbounded"/>
         <element name="derived" type="container:derivedType"
          minOccurs="0" maxOccurs="unbounded"/>
         <element name="linked" type="container:linkedType"
```

```
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="clID" type="eppcom:clIDType"/>
        <element name="crID" type="eppcom:clIDType"/>
        <element name="crDate" type="dateTime"/>
        <element name="upID" type="eppcom:clIDType"
         minOccurs="0"/>
        <element name="upDate" type="dateTime"
         minOccurs="0"/>
        <element name="trDate" type="dateTime"
         minOccurs="0"/>
        <element name="authInfo" type="eppcom:authInfoType"/>
      </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="container:statusValueType"
           use="required"/>
          <attribute name="lang" type="language"
           use="optional" default="en"/>
        </extension>
      </simpleContent>
    </complexType>
    <simpleType name="statusValueType">
      <restriction base="token">
        <enumeration value="clientDeleteProhibited"/>
        <enumeration value="clientTransferProhibited"/>
        <enumeration value="clientUpdateProhibited"/>
        <enumeration value="linked"/>
        <enumeration value="ok"/>
        <enumeration value="pendingDelete"/>
        <enumeration value="pendingTransfer"/>
        <enumeration value="serverDeleteProhibited"/>
        <enumeration value="serverTransferProhibited"/>
        <enumeration value="serverUpdateProhibited"/>
      </restriction>
    </simpleType>
  <!--
  <transfer> response elements.
  -->
    <complexType name="trnDataType">
      <sequence>
        <element name="id" type="container:containerIDType"/>
        <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"/>
      </sequence>
    </complexType>
  <!--
  End of schema.
  -->
  </schema>
```

4. EPP Command Mapping for Objects with Containers

A detailed description of the EPP syntax and semantics can be found in [2]. The extended command mappings described here are specifically for use in provisioning and managing objects using containers via EPP. The objects can be maintained with containers specified here are domains. Other types of objects managed with containers will be included in the future as required. However, any objects included in a container cannot be removed via the <delete> commands, or modified via the <update> command unless the data integrity of all objects associated with the container are ensured.

The EPP commands that can be applied to domain objects MUST be <create> or <update> commands. In addition, the responses to <info> commands MAY contains an OPTIONAL "container" element that indicating the container used in maintaining domain objects. Finally, in order to <transfer> a domain object that is associated with a container, the association MUST be cut-off first via the <update> command for nullifying its container value.

4.1 EPP Command <create> for Domain Object With Container

A domain object can be created with a container. The extended schema for domain object will include an OPTIONAL "container" element in the <create> command.

If the OPTIONAL "container" element is present in the <create> command for creating a domain object, all other required elements described in [6], except for the fully required name of the domain object, become OPTIONAL. All other REQUIRED or OPTIONAL elements specified in [6] are derived from the container specified. If some of the REQUIRED elements cannot be found in the container or any of the ancestor containers of the container, the domain object will not be created. Any REQUIRED or OPTIONAL elements specified in the <create> command will override the values derived from the container object.

Example <create> command with container:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
C:    <create>
C:      <domain:create xmlns:domain="urn:iana:xml:ns:domain-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:domain-1.0
C:       domain-1.0.xsd">
C:        <domain:name>foocorp-ga.tld</domain:name>
C:        <domain:container>CONTAINER-FooCorp-USA</domain:container>
C:        <domain:registrant>FooCorp-Georgia</container:registrant>
C:      </container:create>
C:    </create>
C:    <unspec/>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The response to a <create> command is described in [6].

4.2 EPP Command <update> for Domain Object With Container

Additional domain attribute values can be added or changed via the <update> command to override the default values derived from containers. However, only attribute values specified via the <create> command or added via the <update> command can be removed, and data integrity of the domain object is ensured.

When a domain object is associated with a container, the association can be cut-off or changed with another container. All attribute values for the domain object will be copied from containers into the domain object itself.

When the <chg> element of the <update> command for a domain object contains an OPTIONAL <container> element, which is either empty or a server-unique container identifier, the association to the current container, if present, will be cut-off or changed to the new container. However, if the domain object does not associated with any container, the update command SHOULD fail.

Example <update> command for updating container association:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:iana:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
C:  <command>
```

```
C:    <update>
C:      <domain:update xmlns:domain="urn:iana:xml:ns:domain-1.0"
C:       xsi:schemaLocation="urn:iana:xml:ns:domain-1.0
C:       domain-1.0.xsd">
C:        <domain:name>foocorp-va.com</domain:name>
C:        <domain:chg>
C:          <domain:container>CONTAINER-FooCorp-Virginia</domain:container>
C:          <domain:authInfo type="pw">2BARfoo</domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <unspec/>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

The response to a <update> command is described in [6].

4.3 EPP Command <info> for Domain Object With Container

If a domain object is associated with a container, the response data to the <info> command will contain an OPTIONAL "container" element whose value is the server-unique identifier of the container used for maintaining the domain object.

Example <info> response:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:iana:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData xmlns:domain="urn:iana:xml:ns:domain-1.0"
S:       xsi:schemaLocation="urn:iana:xml:ns:domain-1.0
S:       domain-1.0.xsd">
S:        <domain:name>foocorp-va.tld</domain:name>
S:        <domain:roid>FOOCORP-12-REGISTRY</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:container>CONTAINER-FooCorp-USA</domain:container>
S:        <domain:registrant>FooCorp-Corp</domain:registrant>
S:        <domain:contact type="admin">FooCorpAdmin</domain:contact>
S:        <domain:contact type="tech">FooCorpTech</domain:contact>
S:        <domain:contact type="billing">FooCorpBilling</domain:contact>
S:        <domain:ns>ns1.foocorp.tld</domain:ns>
S:        <domain:ns>ns2.foocorp.tld</domain:ns>
```

```
S:        <domain:host>ns1.foocorp.tld</domain:host>
S:        <domain:host>ns2.foocorp.tld</domain:host>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:crID>ClientY</domain:crID>
S:        <domain:crDate>2001-04-03T22:00:00.0Z</domain:crDate>
S:        <domain:upID>ClientX</domain:upID>
S:        <domain:upDate>2001-12-03T09:00:00.0Z</domain:upDate>
S:        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
S:        <domain:trDate>2002-04-08T09:00:00.0Z</domain:trDate>
S:        <domain:authInfo type="pw">2fooBAR</domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <unspec/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

4.4 Formal Syntax for Domain Object with Container

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.

Instead of giving a full XML Schema for domain object with container extension, the following is the diff output comparing to the original [6] schema. [Editor: This needs to be updated when [6] finishes WGLC.]

```
*** domain-1.0.xsd Tue Jun 26 08:33:20 2001
--- domain-1.1.xsd Thu Aug  2 10:48:50 2001
***************
*** 1,7 ****
  <?xml version="1.0" encoding="UTF-8"?>

! <schema targetNamespace="urn:iana:xml:ns:domain-1.0"
!         xmlns:domain="urn:iana:xml:ns:domain-1.0"
          xmlns:epp="urn:iana:xml:ns:epp-1.0"
          xmlns:eppcom="urn:iana:xml:ns:eppcom-1.0"
          xmlns="http://www.w3.org/2001/XMLSchema"
--- 1,7 ----
  <?xml version="1.0" encoding="UTF-8"?>

! <schema targetNamespace="urn:iana:xml:ns:domain-1.1"
!         xmlns:domain="urn:iana:xml:ns:domain-1.1"
```

```
            xmlns:epp="urn:iana:xml:ns:epp-1.0"
            xmlns:eppcom="urn:iana:xml:ns:eppcom-1.0"
            xmlns="http://www.w3.org/2001/XMLSchema"
***************
*** 18,24 ****
    <annotation>
      <documentation>
        Extensible Provisioning Protocol v1.0
!       domain provisioning schema.
      </documentation>
    </annotation>

--- 18,24 ----
    <annotation>
      <documentation>
        Extensible Provisioning Protocol v1.0
!       domain provisioning schema, with container extension.
      </documentation>
    </annotation>

***************
*** 28,33 ****
--- 28,42 ----
    <element name="svc"/>

  <!--
+ Container identifier type, imported from container-1.0.xsd
+ -->
+   <simpleType name="containerIDType">
+     <restriction base="token">
+       <minLength value="3"/>
+       <maxLength value="64"/>
+     </restriction>
+   </simpleType>
+ <!--
  Child elements found in EPP commands.
  -->
    <element name="check" type="domain:mNameType"/>
***************
*** 44,49 ****
--- 53,60 ----
    <complexType name="createType">
      <sequence>
        <element name="name" type="eppcom:labelType"/>
+       <element name="container" type="domain:containerIDType"
+        minOccurs="0"/>
        <element name="period" type="domain:periodType"
         minOccurs="0"/>
```

```
        <element name="ns" type="eppcom:labelType"
***************
*** 52,58 ****
         minOccurs="0"/>
        <element name="contact" type="domain:contactType"
         minOccurs="0" maxOccurs="unbounded"/>
!       <element name="authInfo" type="eppcom:authInfoType"/>
      </sequence>
    </complexType>

--- 63,70 ----
         minOccurs="0"/>
        <element name="contact" type="domain:contactType"
         minOccurs="0" maxOccurs="unbounded"/>
!       <element name="authInfo" type="eppcom:authInfoType"
!        minOccurs="0"/>
      </sequence>
    </complexType>

***************
*** 228,233 ****
--- 240,246 ----
      <sequence>
        <element name="name" type="eppcom:labelType"/>
        <element name="roid" type="eppcom:roidType"/>
+       <element name="container" type="domain:containerIDType"/>
        <element name="status" type="domain:statusType"
         maxOccurs="14"/>
        <element name="registrant" type="domain:contactType"
```

5. Internationalization Considerations

This memo introduces no international considerations beyond those introduced in [2].

6. IANA Considerations

This memo introduces no IANA considerations beyond those introduced in [EPP].

7. Security Considerations

This memo introduces no security considerations beyond those introduced in [2].

References

[1] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Hollenbeck, S., "Extensible Provisioning Protocol", Internet-Draft <draft-ietf-provreg-epp-NN.txt>, work in progress.

[3] Bray, T., et al, "Extensible Markup Language (XML) 1.0 (Second Edition)", 6 October 2000.

[4] Thompson, H., Beech, D., Maloney, M. and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC-xmlschema-1, May 2001, <http://www.w3.org/TR/xmlschema-1/>.

[5] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes", W3C REC-xmlschema-2, May 2001, <http://www.w3.org/TR/xmlschema-2/>.

[6] Hollenbeck, S., "EPP Domain mapping", Internet-Draft <draft-ietf-provreg-domain-NN.txt>, work in progress.

Editor's Address

Eric Brunner-Williams
1415 Forest Ave.,
Portland, ME 04103

brunner@nic-naa.net

Full Copyright Statement

Copyright (C) The Internet Society (2001). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Many people have contributed input and commentary to earlier versions of this document, including but not limited to: Ayesha Demaraju (co-author), Ning Zhang (co-author), and Scott Hollenbeck and Dan Manley, working group participants.

Funding for the RFC editor function is currently provided by the Internet Society.