Extensible Provisioning Protocol

Status of this Memo

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

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 document describes a connection-oriented, application layer client-server protocol for the provisioning and management of objects stored in a shared central repository. Specified in XML, the protocol defines generic object management operations and an extensible framework that maps protocol operations to 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].

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 is provided only to illustrate element relationships and is not a REQUIRED feature of this protocol.

XML protocol elements are case sensitive.
Table of Contents

1. Introduction .................................................. 3
2. Protocol Description ........................................... 4
  2.1 Protocol Identification .................................... 5
  2.2 Greeting Format ............................................ 5
  2.3 Command Format ............................................. 6
  2.4 Response Format ............................................ 7
  2.5 Protocol Extension Framework ................................ 10
2.6 Protocol Commands ........................................... 11
  2.6.1 Session Management Commands ............................ 11
  2.6.1.1 EPP <login> Command ................................ 11
  2.6.1.2 EPP <logout> Command ................................ 14
  2.6.2 Object Query Commands .................................. 15
  2.6.2.1 EPP <info> Command .................................. 15
  2.6.2.2 EPP <ping> Command .................................. 17
  2.6.2.3 EPP <transfer> Query Command ........................ 19
  2.6.3 Object Transform Commands ............................... 22
  2.6.3.1 EPP <create> Command ................................ 22
  2.6.3.2 EPP <delete> Command ................................ 23
  2.6.3.3 EPP <renew> Command ................................ 25
  2.6.3.4 EPP <transfer> Command ................................ 27
  2.6.3.5 EPP <update> Command ................................ 30
3. Result Codes .................................................. 33
4. Formal Syntax ................................................. 35
5. Internationalization Considerations .......................... 40
6. IANA Considerations .......................................... 41
7. Security Considerations ....................................... 42
8. References .................................................... 43
9. Author's Address ............................................. 44
10. Full Copyright Statement .................................... 45
 Appendix A: Object Mapping Outline ............................ 46
1. Introduction

This document describes specifications for the Extensible Provisioning Protocol (EPP) version 1.0, an XML text protocol that permits multiple service providers to perform object provisioning operations using a shared central object repository. EPP is specified using the Extensible Markup Language (XML) 1.0 as described in [XML] and XML Schema notation as described in [XML-SD] and [XML-SS]. EPP meets and exceeds the requirements for a generic registry registrar protocol as described in [GRRP].

The referenced XML Schema documents recently progressed from Working Draft status to Candidate Recommendation status. The references to these documents and the URIs used to refer to XML Schema namespaces MUST be changed once XML parsers that support the updated specifications are available.

It is important to note that XML is case sensitive. XML specifications and examples provided in this document MUST be interpreted in the exact character case presented to develop a conforming implementation.

This document is being discussed on the "rrp" mailing list. To join the list, send a message to <majordomo@NSIRegistry.net> with the words "subscribe rrp" in the body of the message. There is a web site for the list archives at <http://www.NSIRegistry.net/maillist/rrp>.
2. Protocol Description

EPP is a connection-oriented protocol that can be layered over multiple transport protocols. Clients establish a secure connection with a server, exchange identification, authentication, and option information, and then engage in a series of client-initiated command-response exchanges. All EPP commands are atomic and idempotent.

Specified in XML, EPP provides four basic service elements: a greeting, commands, responses, and an extension framework that supports future definition of managed objects and the relationship of protocol requests and responses to those objects.

An EPP server MUST respond to a successful connection by returning a greeting to the client. The client MUST wait for the greeting before sending an EPP command to the server. EPP commands and responses are exchanged serially between the client and the server. The server MUST respond to each EPP command with a coordinated response that describes the results of processing the command.

EPP commands fall into three categories: session management commands, query commands, and data transform commands. Session management commands are used to establish and end sessions with an EPP server. Query commands are used to perform read-only, object-based information retrieval operations. Transform commands are used to perform read-write object management operations.

EPP uses XML namespaces to provide an extensible object management framework and to identify schemas required for XML instance parsing and validation. These namespaces and schema definitions are used to identify both the base protocol schema and the schemas for managed objects.

All XML instances SHOULD begin with an <?xml?> processing instruction to identify the version of XML that is being used and to provide a hint to the XML parser that a schema file is needed to validate the XML instance. Use of character encodings other than those supported in Unicode MUST be noted by an "encoding" attribute within the XML processing instruction.

Example XML processing instruction:

<?xml version="1.0" standalone="no"?>

This processing instruction identifies the XML version as "1.0", specifies default Unicode character encoding, and tells an XML parser that all of the information needed to validate the XML instance is not included in the XML instance.
2.1 Protocol Identification

All XML instances of EPP MUST begin with an <epp> element. This element identifies the start of an EPP protocol element, the namespace used within the protocol, and the location of the protocol schema. This start element and the associated ending element MUST be applied to all greetings, commands, and responses sent by both clients and servers.

Example "start" and "end" XML elements:

```xml
<epp xmlns="urn:iana:xmlns:epp"
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
</epp>
```

2.2 Greeting Format

An EPP server responds to a successful connection by returning a greeting to the client. An EPP greeting SHALL contain the following elements:

- A <greeting> element that identifies the start of the greeting.
- A <server> element that contains the name of the server.
- A <server-date> element that contains the server’s current date and time in UTC.
- A <service-menu> element that identifies the features supported by the server, including:
  - One or more <version> elements that contain the protocol versions supported by the server.
  - One or more <lang> elements that contain the identifiers of the text response languages known by the server. Language identifiers MUST be structured as documented in [RFC1766]. Only language identifiers listed in [ISO639] MAY be used.
  - One or more object-specific <obj:service> elements that identify the objects that the server is capable of managing.

A server MAY limit object management privileges on a per-client basis.
Example greeting:

S:<?xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <greeting>
S:    <server>Example Company EPP server epp.example.com</server>
S:    <server-date>2000-06-08T22:00:00.0Z</server-date>
S:    <service-menu>
S:      <version>1.0</version>
S:      <lang>en-US</lang>
S:      <lang>fr</lang>
S:      <obj1:service xmlns:obj1="urn:iana:xmlns:obj1"
S:       xsi:schemaLocation="urn:iana:xmlns:obj1 obj1.xsd"/>
S:      <obj2:service xmlns:obj2="urn:iana:xmlns:obj2"
S:       xsi:schemaLocation="urn:iana:xmlns:obj2 obj2.xsd"/>
S:      <obj3:service xmlns:obj3="urn:iana:xmlns:obj3"
S:       xsi:schemaLocation="urn:iana:xmlns:obj3 obj3.xsd"/>
S:    </service-menu>
S:  </greeting>
S:</epp>

2.3 Command Format

Once connected, an EPP client interacts with the EPP server by sending a command to the server and waiting for a response to the command before sending the next command. In addition to an EPP identification element, an EPP command MUST contain the following elements:

- A <command> element that identifies the start of the command.

- A command element whose name corresponds to one of the valid EPP commands described in this document.

- A <trans-id> element that uniquely identifies the command to both the client and the server. A transaction identifier SHALL include the following child elements:

  - A <date> element that contains the date of command execution.

  - A <client-id> element that matches the identifier used by the client when establishing a session with the server. Client identifiers MUST be assigned by and unique to the server.

  - A <code> element that is assigned by and MUST be unique to the client on a per-day basis. Code elements MUST contain a client-coded
combination of letters, numbers, and dashes.

Transaction identifiers provide command-response synchronization integrity. They MUST be logged, retained, and protected to ensure that both the client and the server have consistent transaction records. Their uniqueness and required longevity makes them useful as authorization identifiers for EPP commands that require inter-client knowledge of object sponsorship.

Example command:

S: <?xml version="1.0" standalone="no"?>
C: <epp xmlns="urn:iana:xmlns:epp"
C:   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:   xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:   <command>
C:     <ping>
C:       <obj:ping xmlns:obj="urn:iana:xmlns:obj"
C:         xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:         <obj:name>example</obj:name>
C:       </obj:ping>
C:     </ping>
C:     <trans-id>
C:       <date>2000-06-08</date>
C:       <client-id>ClientX</client-id>
C:       <code>ABC-12345-XYZ</code>
C:     </trans-id>
C:   </command>
C: </epp>

2.4 Response Format

An EPP server responds to client commands by returning a response to the client. EPP commands are atomic, so a command must either succeed completely or fail completely. Success and failure results MUST NOT be mixed. In addition to an EPP identification element, an EPP response SHALL contain the following elements:

- A <response> element that identifies the start of the command.

- One or more <result> elements that document the success or failure of command execution. If the command was processed successfully, only one <result> element SHALL be returned. If the command was not processed successfully, multiple <result> elements MAY be returned to document failure conditions. Each <result> element SHALL contain the following attribute and child elements:

  - A "code" attribute whose value is a four-digit, decimal number
that describes the success or failure of the command.

- A <text> element containing a human-readable description of the
  response code. The language of the response is identified via an
  OPTIONAL "lang" attribute. If not specified, the default attribute
  value SHALL be "en-US".

- An OPTIONAL <value> element that returns a client-provided value
  that caused a server error condition.

- An OPTIONAL <response-data> element that contains child elements
  specific to the command and the object subjected to the command.

- A <trans-id> element containing the transaction identifier provided
  with the command for which the response is being returned. The value
  of the transaction identifier elements MUST match those provided with
  the command for which the response is returned.

Example response without <error-value> or <response-data>:

S:<?xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>
Example response with <response-data>:

S: <?xml version="1.0" standalone="no"?>
S: <epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <response-data>
S:      <obj:obj-data xmlns:obj="urn:iana:xmlns:obj"
S:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
S:        <obj:name>example</obj:name>
S:      </obj:obj-data>
S:    </response-data>
S:  <trans-id>
S:    <date>2000-06-08</date>
S:    <client-id>ClientX</client-id>
S:    <code>ABC-12345-XYZ</code>
S:  </trans-id>
S: </response>
S:</epp>
Example response with error <value>:

S:<?xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S: xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S: xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S: <response>
S:  <result code="2005">
S:    <text>Parameter value range error</text>
S:    <value>2525</value>
S:  </result>
S:  <result code="2006">
S:    <text>Parameter value syntax error</text>
S:    <value>ex(ample</value>
S:    <value>abc.ex(ample</value>
S:  </result>
S:  <trans-id>
S:    <date>2000-06-08</date>
S:    <client-id>ClientX</client-id>
S:    <code>ABC-12345-XYZ</code>
S:  </trans-id>
S: </response>
S:</epp>

2.5 Protocol Extension Framework

EPP provides an extensible object management framework that defines the syntax and semantics of protocol operations applied to a managed object. This framework pushes the definition of each protocol operation into the context of a specific object, providing the ability to add mappings for new objects without having to modify the base protocol.

Protocol elements that contain data specific to objects are identified using XML namespace notation with a reference to an XML schema that defines the namespace. The schema for EPP supports use of dynamic object schemas on a per-command and per-response basis. For example, the start of an object-specific command element would be described in generic terms as follows:

C:<EPP-command-name>
C:  <object:command xmlns:object="urn:iana:xmlns:object"
C:   xsi:schemaLocation="urn:iana:xmlns:object object.xsd">
C:    <!-- One or more object-specific command elements. -->
C:  </object:command>
C:</EPP-command-name>
An object-specific response element would be described similarly:

S:<response-data>
S:  <object:response-data xmlns:object="urn:iana:xmlns:object"
S:    xsi:schemaLocation="urn:iana:xmlns:object object.xsd">
S:    <!-- One or more object-specific response elements. -->
S:  </object:response-data>
S:</response-data>

This document does not define mappings for specific objects. The mapping of EPP to an object MUST be described in separate documents that specifically address each command and response in the context of the object. A suggested object mapping outline is included as an appendix to this document.

2.6 Protocol Commands

EPP provides commands to manage sessions, retrieve object information, and perform transformation operations on objects. All EPP commands are atomic and idempotent, either succeeding completely or failing completely and producing predictable results in case of repeated execution. This section describes each EPP command, including examples with representative server responses.

2.6.1 Session Management Commands

EPP provides two commands for session management: <login> to establish a session with a server, and <logout> to end a session with a server.

2.6.1.1 EPP <login> Command

The EPP <login> command is used to establish a session with an EPP server in response to a greeting issued by the server. A <login> command MUST be sent to a server before any other EPP command.

A client identifier and initial password MUST be created on the server before a client can successfully complete a <login> command. The client identifier and initial password MUST be delivered to the client using an out-of-band method that protects the identifier and password from inadvertent disclosure.

In addition to the standard EPP command elements, the <login> command SHALL contain the following child elements:

- A <client-id> element that contains the client identifier assigned to the client by the server. The value of this element is case insensitive.
- A <password> element that contains the client’s plain text password. Note that EPP uses a variant of the PLAIN SASL authentication mechanism described in [RFC2595]. The value of this element is case sensitive.

- An OPTIONAL <new-password> element that contains a new plain text password to be assigned to the client for use with subsequent <login> commands. The value of this element is case sensitive.

- A <services> element that contains the following child elements:
  - A <version> element that contains the protocol version to be used during the session with the server.
  - A <lang> element that contains the text response language to be used during the session with the server.
  - One or more object <service> elements that identify the objects to be managed during the session.

The values of the <version> and <lang> elements MUST exactly match one of the available values presented within the EPP greeting.

The PLAIN SASL mechanism presented in [RFC2595] describes a format for providing a user identifier, an authorization identifier, and a password as part of a single plain text string. The EPP authentication mechanism is similar, though EPP does not require a separate authorization identifier and the user identifier and password are separated into distinct XML elements.
Example <login> command:

C:<?xml version="1.0" standalone="no"?>
C:<epp xmlns="urn:iana:xmlns:epp"
C:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <login>
C:      <client-id>ClientX</client-id>
C:      <password>foo-BAR2</password>
C:      <new-password>bar-FOO2</new-password>
C:      <services>
C:        <version>1.0</version>
C:        <lang>en-US</lang>
C:        <obj1:service xmlns:obj1="urn:iana:xmlns:obj1"
C:         xsi:schemaLocation="urn:iana:xmlns:obj1 obj1.xsd"/>
C:        <obj2:service xmlns:obj2="urn:iana:xmlns:obj2"
C:         xsi:schemaLocation="urn:iana:xmlns:obj2 obj2.xsd"/>
C:        <obj3:service xmlns:obj3="urn:iana:xmlns:obj3"
C:         xsi:schemaLocation="urn:iana:xmlns:obj3 obj3.xsd"/>
C:      </services>
C:    </login>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>

When a <login> command has been processed successfully, a server MUST respond with an EPP response with no <response-data> element. If successful, the server will respond by establishing a new session.
Example <login> response:

S:<?xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>

Authorization: all clients MUST be authorized to use the <login> command. A <login> command MUST be rejected if issued during an established session.

2.6.1.2 EPP <logout> Command

The EPP <logout> command is used to end a session with an EPP server. In addition to the standard EPP command elements, the <logout> command SHALL contain an empty <logout> command element.

Example <logout> command:

C:<epp xmlns="urn:iana:xmlns:epp"
C:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <logout/>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>

When a <logout> command has been processed successfully, a server MUST respond with an EPP response with no <response-data> element. If successful, the server MUST also end the current session.
Example <logout> response:

S:<?xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1500">
S:      <text>Command completed successfully; ending session</text>
S:    </result>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>

Authorization: all clients MUST be authorized to use the <logout> command. A <logout> command MUST NOT be accepted if issued outside the bounds of an established session.

2.6.2 Object Query Commands

EPP provides three commands to retrieve object information: <info> to retrieve detailed information associated with a known object, <ping> to determine if an object is known to the server, and <transfer> to retrieve known object transfer status information.

2.6.2.1 EPP <info> Command

The EPP <info> command is used to retrieve information associated with a known object. The elements needed to identify an object and the type of information associated with an object are both object-specific, so the child elements of the <info> command are specified using the EPP extension framework. In addition to the standard EPP command elements, the <info> command SHALL contain the following child elements:

- An object-specific <obj:info> element that identifies the object to be queried.
Example <info> command:

C: <?xml version="1.0" standalone="no"?>
C: <epp xmlns="urn:iana:xmlns:epp"
C:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <info>
C:      <obj:info xmlns:domain="urn:iana:xmlns:obj"
C:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:        <obj:name>example</obj:name>
C:      </obj:info>
C:    </info>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>

When an <info> command has been processed successfully, a server MUST respond with an EPP <response-data> element that MUST contain a child element that identifies the object namespace and the location of the object schema. The child elements of the <response-data> element are object-specific.
Example <info> response:

S: <?xml version="1.0" standalone="no"?>
S: <epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <response-data>
S:      <obj:info-data xmlns:obj="urn:iana:xmlns:obj"
S:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
S:        <obj:name>example</obj:name>
S:        <obj:client-id>ClientY</obj:client-id>
S:        <obj:created-by>ClientX</obj:created-by>
S:        <obj:created-date>1999-04-03T22:00:00.0Z</obj:created-date>
S:        <obj:last-updated-by>ClientX</obj:last-updated-by>
S:        <obj:last-updated-date>
S:          1999-12-03T09:00:00.0Z
S:        </obj:last-updated-date>
S:        <obj:auth-id>
S:          <obj:date>2000-04-08</obj:date>
S:          <obj:client-id>ClientY</obj:client-id>
S:          <obj:code>ABC-98765-XYZ</obj:code>
S:        </obj:auth-id>
S:      </obj:info-data>
S:    </response-data>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>

Authorization: all clients MUST be authorized to use the <info> command. Access to object information MAY be restricted to the client that manages the object. Access to an object’s authorization identifier, if present, MUST be restricted to the client that manages the object.

2.6.2.2 EPP <ping> Command

The EPP <ping> command is used to determine if an object is known to the server. The elements needed to identify an object are object-specific, so the child elements of the <ping> command are specified
using the EPP extension framework. In addition to the standard EPP command elements, the <ping> command SHALL contain the following child elements:

- An object-specific <obj:ping> element that identify the objects to be queried. Multiple objects of the same type MAY be queried within a single <ping> command.

Example <ping> command:

C: <?xml version="1.0" standalone="no"?>
C: <epp xmlns="urn:iana:xmlns:epp"
C:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <ping>
C:      <obj:ping xmlns:obj="urn:iana:xmlns:obj"
C:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:        <obj:name>example1</obj:name>
C:        <obj:name>example2</obj:name>
C:        <obj:name>example3</obj:name>
C:      </obj:ping>
C:    </ping>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>

When an <ping> command has been processed successfully, a server MUST respond with an EPP <response-data> element that MUST contain a child element that identifies the object namespace and the location of the object schema. The child elements of the <response-data> element are object-specific.
Example <ping> response:

```xml
S: <?xml version="1.0" standalone="no"?>
S: <epp xmlns="urn:iana:xmlns:epp"
S:   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:   xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <response-data>
S:      <obj:ping-data xmlns:obj="urn:iana:xmlns:obj"
S:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
S:        <obj:name result="known">example1</obj:name>
S:        <obj:name result="unknown">example2</obj:name>
S:        <obj:name result="known">example3</obj:name>
S:      </obj:ping-data>
S:    </response-data>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>
```

Authorization: all clients MUST be authorized to use the <ping> command.

### 2.6.2.3 EPP <transfer> Query Command

The EPP <transfer> command provides a query operation that allows a client to determine real-time status of pending and completed transfer requests. The elements needed to identify an object that is the subject of a transfer request are object-specific, so the child elements of the <transfer> query command are specified using the EPP extension framework. In addition to the standard EPP command elements, the <transfer> command SHALL contain an "op" attribute with value "query", and the following child elements:

- An object-specific <obj:transfer> element that identifies the object whose transfer status is requested.

- An <auth-id> element that contains the data from the <trans-id> element identifying the most recent sponsorship association.
Example <transfer> query command:

```xml
<?xml version="1.0" standalone="no"?>
<epp xmlns="urn:iana:xmlns:epp"
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
  <command>
    <transfer op="query">
      <obj:transfer-query xmlns:obj="urn:iana:xmlns:obj"
       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
      <obj:name>example</obj:name>
      </obj:transfer-query>
      <auth-id>
        <date>1999-06-08</date>
        <client-id>ClientX</client-id>
        <code>ABC-98765-XYZ</code>
      </auth-id>
      <trans-id>
        <date>2000-06-08</date>
        <client-id>ClientX</client-id>
        <code>ABC-12345-XYZ</code>
      </trans-id>
    </transfer>
  </command>
</epp>
```

When an <transfer> query command has been processed successfully, a server MUST respond with an EPP <response-data> element that MUST contain a child element that identifies the object namespace and the location of the object schema. The child elements of the <response-data> element are object-specific, but each response MUST include the following information:

- An object-specific <obj:transfer-data> element that identifies the type of object whose transfer status is requested.
- An object-specific element that provides the name or other identifier used to uniquely identify the object.
- An object-specific <obj:request-client> element that provides the identifier of the client that initiated the transfer request.
- An object-specific <obj:action-client> element that provides the identifier of the client that SHOULD respond to the transfer request.
- An object-specific <obj:transfer-status> element that contains the state of the most recent transfer request. Valid values are
"PENDING", "APPROVED", "REJECTED", "AUTO-APPROVED", "AUTO-REJECTED", and "CANCELLED".

- An object-specific <obj:request-date> element that contains the date and time that the transfer was requested.

- An object-specific <obj:action-date> element that contains the date and time of a required or completed response. For a PENDING request, the value contains the date and time by which a response is required before an automated response action SHALL be taken by the server. For all other status types, the value contains the date and time when the request was completed.

Example <transfer> query response:

S:<?xml version="1.0" standalone="no"?>
S: <epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <response-data>
S:      <obj:transfer-data xmlns:obj="urn:iana:xmlns:obj"
S:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
S:        <obj:name>example</obj:name>
S:        <obj:request-client>ClientX</obj:request-client>
S:        <obj:action-client>ClientY</obj:action-client>
S:        <obj:transfer-status>PENDING</obj:transfer-status>
S:        <obj:request-date>2000-06-06T22:00:00.0Z</obj:request-date>
S:        <obj:action-date>2000-06-11T22:00:00.0Z</obj:action-date>
S:      </obj:transfer-data>
S:    </response-data>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>

Authorization: all clients MUST be authorized to use the <transfer> query command. A client MUST be authorized to query an object for which they are either the requesting or the responding client. A client MUST NOT be authorized to query an object for which they are neither the requesting or the responding client.
2.6.3 Object Transform Commands

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

2.6.3.1 EPP <create> Command

The EPP <create> command is used to create an instance of an object. An object may be created for an indefinite period of time, or an object may be created for a specific validity period. The EPP mapping for an object MUST describe the status of an object with respect to time, to include expected client and server behavior if a validity period is used.

The elements needed to identify an object and associated attributes are object-specific, so the child elements of the <create> command are specified using the EPP extension framework. In addition to the standard EPP command elements, the <create> command SHALL contain the following child elements:

- An object-specific <obj:create> element that identifies the object to be created and the elements that are required to create the object.

Example <create> command:

```xml
C:<?xml version="1.0" standalone="no"?>
C:<epp xmlns="urn:iana:xmlns:epp"
C:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <create>
C:      <obj:create xmlns:obj="urn:iana:xmlns:obj"
C:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:        <obj:name>example</obj:name>
C:      </obj:create>
C:    </create>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>
```

The child elements of the <trans-id> element provided by the client
are also used to authorize transfer commands. If the <create> command was performed on behalf of a third party, the client executing the <create> command MUST provide the transaction identifier information to the third party for use in future transfer requests. This identifying information MUST NOT be available to anyone except the client and the third party. Only the third party MAY disclose this information to another client to authorize a transfer request.

When a <create> command has been processed successfully, a server MUST respond with an EPP <response-data> element that MUST contain a child element that identifies the object namespace and the location of the object schema.

Example <create> response:

```xml
S:<?xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <response-data>
S:      <obj:expire-data xmlns:obj="urn:iana:xmlns:obj"
S:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
S:        <obj:name>example</obj:name>
S:        <obj:expiration-date>
S:          2002-06-08T22:00:00.0Z
S:        </obj:expiration-date>
S:      </obj:expire-data>
S:    </response-data>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>
```

Authorization: all clients MUST be authorized to use the <create> command.

2.6.3.2 EPP <delete> Command

The EPP <delete> command is used to remove an instance of a known object. The elements needed to identify an object are object-specific, so the child elements of the <delete> command are specified
using the EPP extension framework. In addition to the standard EPP command elements, the <delete> command SHALL contain the following child elements:

- An object-specific <obj:delete> element that identifies the object to be deleted.

Example <delete> command:

```
C:<?xml version="1.0" standalone="no"?>
C:<epp xmlns="urn:iana:xmlns:epp"
C:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <delete>
C:      <obj:delete xmlns:obj="urn:iana:xmlns:obj"
C:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:        <obj:name>example</obj:name>
C:      </obj:delete>
C:    </delete>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>
```

When a <delete> command has been processed successfully, a server MUST respond with an EPP response with no <response-data> element. If successful, the server will respond by returning a response code that confirms that the <delete> command has been accepted.
Example <delete> response:

S: <?xml version="1.0" standalone="no"?>
S: <epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S: </epp>

Authorization: all clients MUST be authorized to use the <delete> command. An object MAY be deleted only by the current sponsoring client.

2.6.3.3 EPP <renew> Command

The EPP <renew> command is used to extend the validity period of an object. The elements needed to identify and extend the validity period of an object are object-specific, so the child elements of the <renew> command are specified using the EPP extension framework. In addition to the standard EPP command elements, the <renew> command SHALL contain the following child elements:

- An object-specific <obj:renew> element that identifies the object to be renewed and the elements that are required to extend the validity period of the object.
Example <renew> command:

C: <?xml version="1.0" standalone="no"?>
C: <epp xmlns="urn:iana:xmlns:epp"
C:   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:   xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <renew>
C:      <obj:renew xmlns:obj="urn:iana:xmlns:obj"
C:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:        <obj:name>example</obj:name>
C:        <obj:current-expiration-year>
C:          2000
C:        </obj:current-expiration-year>
C:        <obj:period>2</obj:period>
C:      </obj:renew>
C:    </renew>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>

When an <renew> command has been processed successfully, a server MUST respond with an EPP <response-data> element that MUST contain a child element that identifies the object namespace and the location of the object schema. Object-specific response elements SHALL be returned as child elements of a <response-data> element.
Example <renew> response:

S:?xml version="1.0" standalone="no"?><epp xmlns="urn:iana:xmlns:epp"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
  <response>
    <result code="1000">
      <text>Command completed successfully</text>
    </result>
    <response-data>
      <obj:expire-data xmlns:obj="urn:iana:xmlns:obj"
        xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
        <obj:name>example</obj:name>
        <obj:expiration-date>2005-04-03T22:00:00.0Z</obj:expiration-date>
      </obj:expire-data>
    </response-data>
    <trans-id>
      <date>2000-06-08</date>
      <client-id>ClientX</client-id>
      <code>ABC-12345-XYZ</code>
    </trans-id>
  </response>
</epp>

Authorization: all clients MUST be authorized to use the <renew> command. An object MAY be renewed only by the current sponsoring client. Object renewal MAY be limited to time limitations that are server-specific.

2.6.3.4 EPP <transfer> Command

The EPP <transfer> command is used to manage changes in client sponsorship of a known object. Clients may initiate a transfer request, cancel a transfer request, approve a transfer request, and reject a transfer request using the "op" command attribute.

Every <transfer> command MUST include an authorization identifier to confirm transfer authority. This identifier is a copy of the transaction identifier associated with the most recent command causing a change of sponsorship, such as the most recently successful <transfer> command or the original <create> command. The identifier associated with the original <create> command MUST be used to authorize the first transfer of an object. After an object has been successfully transferred at least once, the identifier associated wit
the most recent successful <transfer> command MUST be used to authorize transfer of an object. Clients performing a <transfer> command on behalf of a third party MUST provide the third party with a copy of the transaction identifier used to request the transfer.

Authorization identifier information MUST NOT be disclosed to any other client or third party. A client who wishes to transfer an object on behalf of a third party MUST receive authorization identifier information from the third party before a <transfer> command can be successful.

A client who wishes to assume sponsorship of a known object from another client uses the <transfer> command with the value of the "op" attribute set to "request". Once a transfer has been requested, the same client may cancel the request using a <transfer> command with the value of the "op" attribute set to "cancel". A request to cancel the transfer MUST be sent to the server before the current sponsoring client either approves or rejects the transfer request and before the server automatically processes the request due to responding client inactivity.

Once a transfer request has been received by the server, the server MUST notify the current sponsoring client of the requested transfer. This notification MUST be done using an out-of-band communication mechanism such as offline reports and/or electronic mail. The current status of a pending <transfer> command for any object MAY be found using the <transfer> query command.

The current sponsoring client MAY explicitly approve or reject the transfer request. The client may approve the request using a <transfer> command with the value of the "op" attribute set to "approve". The client may reject the request using a <transfer> command with the value of the "op" attribute set to "reject".

A server MUST automatically approve or reject all transfer requests that are not explicitly approved or rejected by the current sponsoring client within a fixed amount of time. The amount of time to wait for explicit action and the default server behavior are local matters not specified by EPP, but they SHOULD be documented in a server-specific profile document that describes default server behavior for client information.

EPP does not provide a mechanism to push notice of new transfer requests to clients. A server MUST provide out-of-band services to inform clients of a transfer for which a response is expected; electronic mail and/or offline reporting MAY be used to provide clients with transfer notices. Once a client is aware of a requested transfer, information about the request may be found using the
<transfer> query command.

The elements needed to identify and complete the transfer of an object are object-specific, so the child elements of the <transfer> command are specified using the EPP extension framework. In addition to the standard EPP command elements, the <transfer> command SHALL contain the following child elements:

- An object-specific <obj:transfer> element that identifies the object to be transferred and the elements that are required to process the transfer command.

- An <auth-id> element that contains the data from the <trans-id> element identifying the most recent sponsorship association.

Example <transfer> request command:

```
C:<?xml version="1.0" standalone="no"?>
C:<epp xmlns="urn:iana:xmlns:epp"
C:    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:    xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <transfer op="request">
C:      <obj:transfer xmlns:obj="urn:iana:xmlns:obj"
C:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:        <obj:name>example</obj:name>
C:      </obj:transfer>
C:      <auth-id>
C:        <date>1999-06-08</date>
C:        <client-id>ClientY</client-id>
C:        <code>ABC-98765-XYZ</code>
C:      </auth-id>
C:    </transfer>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C:</epp>
```

When an <transfer> command has been processed successfully, a server MUST respond with an EPP <response-data> element that MUST contain a child element that identifies the object namespace and the location of the object schema. Object-specific response elements SHALL be returned as child elements of a <response-data> element.
Example <transfer> response:

S:<?xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <response-data>
S:      <obj:transfer-data xmlns:obj="urn:iana:xmlns:obj"
S:       xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
S:        <obj:name>example</obj:name>
S:        <obj:request-client>ClientX</obj:request-client>
S:        <obj:action-client>ClientY</obj:action-client>
S:        <obj:transfer-status>PENDING</obj:transfer-status>
S:        <obj:request-date>2000-06-08T22:00:00.0Z</obj:request-date>
S:        <obj:action-date>2000-06-13T22:00:00.0Z</obj:action-date>
S:      </obj:transfer-data>
S:    </response-data>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>

Authorization: all clients MUST be authorized to use the <transfer> command. All <transfer> commands MUST be accompanied by the authorization identifier associated with the object. A <transfer> request MUST only be accepted from a client other than the current sponsoring client. A <transfer> approval request MUST only be accepted from the current sponsoring client. A <transfer> cancellation request MUST be accepted ONLY from the original requesting client.

2.6.3.5 EPP <update> Command

The EPP <update> command is used to change information associated with a known object. The elements needed to identify and modify an object are object-specific, so the child elements of the <update> command are specified using the EPP extension framework. In addition to the standard EPP command elements, the <update> command SHALL contain the following child elements:
An object-specific `<obj:update>` element that identifies the object to be renewed and the elements that are required to modify the object. Object-specific elements MUST identify values to be added, values to be removed, or values to be changed.

Example `<update>` command:

```
C: <?xml version="1.0" standalone="no"?>
C: <epp xmlns="urn:iana:xmlns:epp"
C:   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
C:   xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
C:  <command>
C:    <update>
C:      <obj:update xmlns:obj="urn:iana:xmlns:obj"
C:        xsi:schemaLocation="urn:iana:xmlns:obj obj.xsd">
C:        <obj:name>example</obj:name>
C:        <obj:add>
C:          <obj:foo>example</obj:foo>
C:        </obj:add>
C:        <obj:remove>
C:          <obj:bar>example</obj:bar>
C:        </obj:remove>
C:      </obj:update>
C:    </update>
C:    <trans-id>
C:      <date>2000-06-08</date>
C:      <client-id>ClientX</client-id>
C:      <code>ABC-12345-XYZ</code>
C:    </trans-id>
C:  </command>
C: </epp>
```

When an `<update>` command has been processed successfully, a server MUST respond with an EPP response with no `<response-data>` element. If successful, the server will respond by returning a result code that confirms that the `<update>` command has been accepted.
Example <update> response:

S:<xml version="1.0" standalone="no"?>
S:<epp xmlns="urn:iana:xmlns:epp"
S:     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
S:     xsi:schemaLocation="urn:iana:xmlns:epp epp.xsd">
S:  <response>
S:    <result code="1000">
S:      <text>Command completed successfully</text>
S:    </result>
S:    <trans-id>
S:      <date>2000-06-08</date>
S:      <client-id>ClientX</client-id>
S:      <code>ABC-12345-XYZ</code>
S:    </trans-id>
S:  </response>
S:</epp>
S:</epp>

Authorization: all clients MUST be authorized to use the <update> command. An object MAY be updated only by the current sponsoring client.
3. Result Codes

EPP result codes are based on the Theory of Reply Codes described in Appendix E of [RFC821]. EPP uses four decimal digits to describe the success or failure of each EPP command. The four digits of the reply each have special significance.

The first digit denotes whether the response marks command success or failure. A client that wants to know approximately what kind of error occurred (command syntax error, security error, system error, etc.) may examine the second digit. The third and fourth digits are used to provide explicit information detail.

There are two values for the first digit of the reply code:

1yzz  Positive completion reply. The command has been accepted and processed by the system without error.

2yzz  Negative completion reply. The command was not accepted and the requested action did not occur.

The second digit groups responses into specific categories:

x0zz  Protocol Syntax
x1zz  Implementation-specific Rules
x2zz  Security
x3zz  Data Management
x4zz  Server System
x5zz  Connection Management

The third and fourth digits provide response detail within the categories defined by the first and second digits.

Every EPP response MUST include a result code and a human-readable description of the result code. The language used to represent the description MAY be identified using an instance of the "lang" attribute within the <response-text> element. If not specified, the default language is US English, identified as "en-US". A description of the structure of valid values for the "lang" attribute is described in [RFC1766]. A list of valid values for the "lang" attribute is available in [ISO639].
Successful command completion responses:

<table>
<thead>
<tr>
<th>Code</th>
<th>Response text in US English</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Command completed successfully</td>
</tr>
<tr>
<td>1500</td>
<td>Command completed successfully; ending session</td>
</tr>
</tbody>
</table>

Command error responses:

<table>
<thead>
<tr>
<th>Code</th>
<th>Response text in US English</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>Unknown command</td>
</tr>
<tr>
<td>2001</td>
<td>Invalid command sequence</td>
</tr>
<tr>
<td>2002</td>
<td>Invalid command structure</td>
</tr>
<tr>
<td>2003</td>
<td>Unknown parameter</td>
</tr>
<tr>
<td>2004</td>
<td>Required parameter missing</td>
</tr>
<tr>
<td>2005</td>
<td>Parameter value range error</td>
</tr>
<tr>
<td>2006</td>
<td>Parameter value syntax error</td>
</tr>
<tr>
<td>2100</td>
<td>Billing failure</td>
</tr>
<tr>
<td>2102</td>
<td>Object is not eligible for renewal</td>
</tr>
<tr>
<td>2103</td>
<td>Object is not eligible for transfer</td>
</tr>
<tr>
<td>2200</td>
<td>Authentication failure</td>
</tr>
<tr>
<td>2201</td>
<td>Authorization failure</td>
</tr>
<tr>
<td>2202</td>
<td>Invalid authorization identifier</td>
</tr>
<tr>
<td>2203</td>
<td>Object authorization failure</td>
</tr>
<tr>
<td>2300</td>
<td>Object pending transfer</td>
</tr>
<tr>
<td>2301</td>
<td>Object not pending transfer</td>
</tr>
<tr>
<td>2302</td>
<td>Object not unique</td>
</tr>
<tr>
<td>2303</td>
<td>Object not known</td>
</tr>
<tr>
<td>2304</td>
<td>Parent object not known</td>
</tr>
<tr>
<td>2305</td>
<td>Object status prohibits operation</td>
</tr>
<tr>
<td>2306</td>
<td>Parent object status prohibits operation</td>
</tr>
<tr>
<td>2307</td>
<td>Invalid parameter value</td>
</tr>
<tr>
<td>2308</td>
<td>Duplicate transaction identifier</td>
</tr>
<tr>
<td>2400</td>
<td>Command failed</td>
</tr>
<tr>
<td>2500</td>
<td>Command failed; server ending session</td>
</tr>
<tr>
<td>2501</td>
<td>Timeout; server ending session</td>
</tr>
<tr>
<td>2502</td>
<td>Connection limit exceeded; server ending session</td>
</tr>
</tbody>
</table>
4. Formal Syntax

EPP is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of EPP suitable for automated validation of EPP XML instances.

```xml
<?xml version="1.0"?>
<!-- This is the base schema for the Extensible Provisioning Protocol version 1.0. -->
<schema xmlns="http://www.w3.org/1999/XMLSchema"
        xmlns:epp="urn:iana:xmlns:epp"
        targetNamespace="urn:iana:xmlns:epp"
        elementFormDefault="qualified">
  <annotation>
    <documentation>
      Extensible Provisioning Protocol version 1.0 schema.
    </documentation>
  </annotation>

  <!-- An EPP XML instance must begin with this element. -->
  <element name="epp" type="epp:eppType"/>

  <!-- An EPP XML instance must contain a greeting, command, or response. -->
  <complexType name="eppType" content="elementOnly">
    <choice>
      <element name="greeting" type="epp:greetingType"/>
      <element name="command" type="epp:commandType"/>
      <element name="response" type="epp:responseType"/>
    </choice>
  </complexType>

  <!-- A server greeting identifies available service options. -->
  <complexType name="serviceMenuType" content="elementOnly">
    <element name="version" type="epp:versionType"
              minOccurs="1" maxOccurs="unbounded"/>
    <element name="lang" type="language"
              minOccurs="1" maxOccurs="unbounded"/>
  </complexType>
</schema>
```
A client login identifies the services it wishes to use.

<!--
Various simple type definitions.
-->

<simpleType name="serverIDType" base="string">
    <minLength value="10"/>
    <maxLength value="80"/>
</simpleType>

<simpleType name="passwordType" base="string">
    <minLength value="6"/>
    <maxLength value="16"/>
    <pattern value="[!-9;-~]{6,16}"/>
</simpleType>

<simpleType name="clientIDType" base="string">
    <minLength value="3"/>
    <maxLength value="16"/>
    <pattern value="[A-Za-z0-9]{3,16}"/>
</simpleType>

<complexType name="emptyType" content="empty"/>

<!--
An EPP version number is a dotted pair of decimal numbers.
-->

<simpleType name="versionType" base="string">
    <enumeration value="1.0"/>
    <pattern value="[1-9].[0-9]"/>
</simpleType>

<!--
An EPP greeting is sent by a server in response to a client connection.
-->

<complexType name="greetingType" content="elementOnly">
    <element name="server" type="epp:serverIDType"/>
</complexType>
<element name="server-date" type="timeInstant"/>
<element name="service-menu" type="epp:serviceMenuType"/>
</complexType>

EPP commands are listed here. Only one command is allowed per
EPP element.

<complexType name="commandType" content="elementOnly">
  <choice>
    <element name="create" type="epp:objectSpecificType"/>
    <element name="delete" type="epp:objectSpecificType"/>
    <element name="info" type="epp:objectSpecificType"/>
    <element name="login" type="epp:loginType"/>
    <element name="logout" type="epp:emptyType"/>
    <element name="ping" type="epp:loginType"/>
    <element name="renew" type="epp:objectSpecificType"/>
    <element name="transfer" type="epp:transferType"/>
    <element name="update" type="epp:objectSpecificType"/>
  </choice>
  <element name="trans-id" type="epp:xidType"/>
</complexType>

The <login> command.

<complexType name="loginType" content="elementOnly">
  <element name="client-id" type="epp:clientIDType"/>
  <element name="password" type="epp:passwordType"/>
  <element name="new-password" type="epp:passwordType"
            minOccurs="0" maxOccurs="1"/>
  <element name="services" type="epp:servicesType"/>
</complexType>

The <transfer> command type. This is object-specific, and uses
attributes to identify the requested operation.

<complexType name="transferType" content="elementOnly">
  <any namespace="##other"/>
  <element name="auth-id" type="epp:xidType"/>
</complexType>
<attribute name="op" use="required" type="epp:transferOpType"/>
</complexType>

<!--
All other object-centric commands. EPP doesn’t specify the syntax or
semantics of object-centric command elements. The elements MUST be
described in detail in another schema specific to the object.
-->
<complexType name="objectSpecificType" content="elementOnly">
  <any namespace="##other"/>
</complexType>

<!--
Transaction and authorization identifier type.
-->
<complexType name="xidType" content="elementOnly">
  <element name="date" type="date"/>
  <element name="client-id" type="epp:clientIDType"/>
  <element name="code" type="epp:codeType"/>
</complexType>

<!--
Human-readable text returned to a client may be in a language other
than English. This type allows specification of alternative languages.
-->
<complexType name="clientTextType" content="textOnly" base="string" derivedBy="extension">
  <minLength value="3"/>
  <maxLength value="80"/>
  <attribute name="lang" type="language" use="default" value="en-US"/>
</complexType>

<!--
EPP <response> type.
-->
<complexType name="resultCodeType" base="unsignedShort">
  <enumeration value="1000"/>
  <enumeration value="1500"/>
  <enumeration value="2000"/>
  <enumeration value="2001"/>
  <enumeration value="2002"/>
  <enumeration value="2003"/>
</complexType>
<simpleType>
  <enumeration value="2004"/>
  <enumeration value="2005"/>
  <enumeration value="2006"/>
  <enumeration value="2100"/>
  <enumeration value="2101"/>
  <enumeration value="2102"/>
  <enumeration value="2200"/>
  <enumeration value="2201"/>
  <enumeration value="2202"/>
  <enumeration value="2203"/>
  <enumeration value="2300"/>
  <enumeration value="2301"/>
  <enumeration value="2302"/>
  <enumeration value="2303"/>
  <enumeration value="2304"/>
  <enumeration value="2305"/>
  <enumeration value="2306"/>
  <enumeration value="2307"/>
  <enumeration value="2308"/>
  <enumeration value="2400"/>
  <enumeration value="2500"/>
  <enumeration value="2501"/>
  <enumeration value="2502"/>
  <minInclusive value="1000"/>
  <maxInclusive value="9999"/>
</simpleType>

<complexType name="resultType" content="elementOnly">
  <element name="text" type="epp:clientTextType"/>
  <element name="value" type="string"
            minOccurs="0" maxOccurs="unbounded"/>
  <attribute name="code" use="required"
             type="epp:resultCodeType"/>
</complexType>

<complexType name="responseType" content="elementOnly">
  <element name="result" type="epp:resultType"
           minOccurs="1" maxOccurs="unbounded"/>
  <element name="response-data" type="epp:objectSpecificType"
           minOccurs="0" maxOccurs="1"/>
  <element name="trans-id" type="epp:xidType"/>
</complexType>

<!-- End of schema. -->

Hollenbeck                Expires May 10, 2001                 [Page 39]
5. Internationalization Considerations

EPP is represented in XML, which provides native support for encoding information using the double-byte Unicode character set and its more compact representations including UTF-8. Compliant XML processors are required to understand both UTF-8 and raw Unicode character sets; XML also includes a provision for identifying other character sets through use of an "encoding" attribute in an <?xml?> processing instruction. The complete list of character set encoding identifiers is maintained by IANA and is described in [ CHARSET] and [RFC1700].

EPP includes a provision for returning a human-readable message with every result code. This document describes result codes in US English, but the actual text returned with a result MAY be provided in a language negotiated when a session is established. Languages other than US English MUST be noted through specification of a "lang" attribute for text-based elements. Valid values for the "lang" attribute and "lang" negotiation elements are described in [RFC1766].

All date-time values presented via EPP MUST be expressed in Universal Coordinated Time. The XML Schema "date" format allows use of time zone identifiers to indicate offsets from the zero meridian, but this option MUST NOT be used within EPP. Both extended and truncated date and time forms defined in [ISO8601] MAY be used.
6. IANA Considerations

XML schemas require a URI for unique identification. Schemas MUST be registered to ensure URI uniqueness, but the IETF does not currently have a recommended repository for the registration of XML schemas. This document uses URNs to describe XML namespaces and XML schemas. IANA SHOULD maintain a registry of XML namespace and schema URI assignments. Per policies described in [IANA], URI assignment requests SHOULD be reviewed by a designated expert, and values SHOULD be assigned only as a result of standards action taken by the IESG.
7. Security Considerations

EPP provides only simple client authentication services. A passive
attack is sufficient to recover client identifiers and passwords,
allowing trivial command forgery. Protection against most common
attacks must be provided by other protocols.

EPP uses a variant of the PLAIN SASL mechanism described in [RFC2595]
to provide a simple application-layer authentication service. Where
the PLAIN SASL mechanism specifies provision of an authorization
identifier, authentication identifier, and password as a single string
separated by ASCII NUL characters, EPP specifies use of a combined
authorization and authentication identifier and a password provided as
distinct XML elements.

Repeated password guessing attempts can be discouraged by limiting the
number of <login> attempts that can be attempted on an open
connection. A server MUST close an open connection if three <login>
Attempts are made with either an invalid client identifier, an invalid
password, or both an invalid client identifier and an invalid
password.

EPP uses transaction identifier information to authorize transfer
commands. When an object is created or transferred on behalf of a
third party, the identifier associated with the EPP <create> or
<transfer> command MUST be provided to the third party using a
facility that provides privacy services.
8. References


9. Author’s Address

Scott Hollenbeck
VeriSign Global Registry Services
21345 Ridgetop Circle
Dulles, VA 20166-6503
USA
shollenbeck@verisign.com
10. Full Copyright Statement

Copyright (C) The Internet Society 2000. 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

Funding for the RFC Editor function is currently provided by the Internet Society.
Appendix A: Object Mapping Outline

This appendix describes a recommended outline for documenting the EPP mapping of an object. Documents that describe EPP object mappings SHOULD describe the mapping in a format similar to the one used here. Note that additional sections will be required if the object mapping is written in Internet-Draft format.

1. Introduction

Provide an introduction that describes the object and an overview of the mapping to EPP.

2. Object Attributes

Describe the attributes associated with the object, including references to syntax specifications as appropriate. Examples of object attributes include a name or identifier and dates associated with modification events.

3. EPP Command Mapping

3.1 EPP Query Commands

3.1.1 EPP <info> Command

Describe the object-specific mappings required to implement the EPP <info> command. Include both sample commands and sample responses.

3.1.2 EPP <ping> Command

Describe the object-specific mappings required to implement the EPP <ping> command. Include both sample commands and sample responses.

3.1.3 EPP <transfer> Command

Describe the object-specific mappings required to implement the EPP <transfer> query command. Include both sample commands and sample responses.

3.2 EPP Transform Commands

3.2.1 EPP <create> Command

Describe the object-specific mappings required to implement the EPP <create> command. Include both sample commands and sample responses. Describe the status of the object with respect to time, including expected client and server behavior if a validity period is used.
3.2.2 EPP <delete> Command

Describe the object-specific mappings required to implement the EPP <delete> command. Include both sample commands and sample responses.

3.2.3 EPP <renew> Command

Describe the object-specific mappings required to implement the EPP <renew> command. Include both sample commands and sample responses.

3.2.4 EPP <transfer> Command

Describe the object-specific mappings required to implement the EPP <transfer> command. Include both sample commands and sample responses.

3.2.5 EPP <update> Command

Describe the object-specific mappings required to implement the EPP <update> command. Include both sample commands and sample responses.

4. Formal Syntax

Provide the XML schema for the object mapping. An XML DTD MUST NOT be used as DTDs do not provide sufficient support for XML namespaces and strong data typing.