Network Working Group
Request for Comments: 3735
Category: Informational

S. Hollenbeck
VeriSign, Inc.

March 2004

Guidelines for Extending the Extensible Provisioning Protocol (EPP)

Status of this Memo

Copyright Notice

Abstract

The Extensible Provisioning Protocol (EPP) is an 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. This document presents guidelines for use of EPP's extension mechanisms to define new features and object management capabilities.

Table of Contents

1. Introduction

The Extensible Provisioning Protocol (EPP, [2]) was originally designed to provide a standard Internet domain name registration protocol for use between Internet domain name registrars and domain name registries. However, specific design decisions were made to ensure that the protocol could also be used in other provisioning environments. Specifically:

- Extensibility has been a design goal from the very beginning. EPP is represented in the Extensible Markup Language (XML, [3]), and is specified in XML Schema ([4] and [5]) with XML namespaces [6] used to identify protocol grammars.
- The EPP core protocol specification describes general protocol functions, not objects to be managed by the protocol. Managed object definitions, such as the mapping for Internet domain names [10] (itself a protocol extension), are loosely coupled to the core specification.
- A concentrated effort was made to separate required minimum protocol functionality from object management operating logic.
- Several extension mechanisms were included to allow designers to add new features or to customize existing features for different operating environments.

This document describes EPP's extensibility features in detail and provides guidelines for their use. Though written specifically for protocol designers considering EPP as the solution to a provisioning problem, anyone interested in using XML to represent IETF protocols might find these guidelines useful.

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

1.1. 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 RFC 2119 [1].

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 specification.

2. Principles of Protocol Extension

The EPP extension model is based on the XML representation for a wildcard schema component using an <any> element information item (as described in section 3.10.2 of [4]) and XML namespaces [6]. This section provides guidelines for the development of protocol extensions and describes the extension model in detail.

Extending a protocol implies the addition of features without changing the protocol itself. In EPP that means that an extension MUST NOT alter an existing protocol schema as changes may result in new versions of an existing schema, not extensions of an existing schema. For example, a designer MUST NOT add new elements to an existing schema and call the result an "extension" of the protocol. The result is a new, non-backwards-compatible version of an existing schema. Extensions MUST adhere to the principles described in this section to be considered valid protocol extensions.

EPP extensions MUST be specified in XML. This ensures that parsers capable of processing EPP structures will also be capable of processing EPP extensions. Guidelines for the use of XML in IETF protocols (thus good information for extension designers) can be found in RFC 3470 [11].

A designer MUST remember that extensions themselves MAY also be extensible. A good chain of extensions is one in which the protocol schemas evolve from general functionality to more specific (perhaps even more limited) functionality.

2.1. Documenting Extensions

The EPP core specification [2] has an appendix that contains a suggested outline to document protocol extensions. Designers are free to use this template or any other format as they see fit, but the extension document SHOULD at a minimum address all of the topics listed in the template.

Extension designers need to consider the intended audience and consumers of their extensions. Extensions MAY be documented as Internet-Draft and RFC documents if the designer is facing requirements for coordination, interoperability, and broad distribution, though the intended maturity level (informational, proposed standard, etc.) largely depends on what is being extended

and the amount of general interest in the extension. An extension to a standards-track specification with broad interest might well be a candidate for standards track publication, whereas an extension to a standards track specification with limited interest might be better suited for informational publication.

Extensions need not be published as Internet-Draft or RFC documents if they are intended for operation in a closed environment or are otherwise intended for a limited audience. In such cases extensions MAY be documented in a file and structural format that is appropriate for the intended audience.

2.2. Identifying Extensions

An EPP extension is uniquely identified by a Uniform Resource Identifier (URI, defined in RFC 2396 [7]). The URI used to identify the extension MUST also be used to identify the XML namespace associated with the extension. Any valid URI MAY be used to identify an EPP extension, though the selection of a URI form (Uniform Resource Locator (URL) vs. Uniform Resource Name (URN), hierarchical vs. relative, etc.) SHOULD depend on factors such as organizational policies on change control and a balance between locating resources and requirements for persistence. An extension namespace MAY describe multiple extension mechanisms, such as definition of new protocol features, objects, or elements, within the schema used to define the namespace.

The following are sample extension-identifying URIs:

```
urn:ietf:params:xml:ns:foo-ext1

http://custom/obj1ext-1.0
```

Extension designers MAY include version information in the URI used to identify an extension. If version information is included in the URI, the URI itself will need to change as the extension is revised or updated.

2.2.1. Standards Track Extensions

URIs for extensions intended for IETF standards track use MUST conform to the URN syntax specifications and registration procedures described in [8].

2.2.2. Other Extensions

URIs for extensions that are not intended for IETF standards track use MUST conform to the URI syntax specifications described in RFC 2396.

2.3. Extension Announcement and Selection

An EPP server MUST announce extensions that are available for client use as part of a <greeting> element that is sent to a client before the client establishes an interactive session with the server. The <greeting> element contains zero or more <svcExtension> elements that, if present, contain a URI identifying an available extension:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
S:     epp-1.0.xsd">
S:  <greeting>
S:    <svID>Example EPP server epp.example.com</svID>
S:    <svDate>2000-06-08T22:00:00.0Z</svDate>
S:    <svcMenu>
S:      <version>1.0</version>
S:      <lang>en</lang>
S:      <lang>fr</lang>
S:      <objURI>urn:ietf:params:xml:ns:obj1</objURI>
S:      <objURI>urn:ietf:params:xml:ns:obj2</objURI>
S:      <objURI>urn:ietf:params:xml:ns:obj3</objURI>
S:      <svcExtension>
S:        <extURI>urn:ietf:params:xml:ns:foo-ext1</extURI>
S:        <extURI>http://custom/obj1ext-1.0</extURI>
S:      </svcExtension>
S:    </svcMenu>
S:    <dcp>
S:      <access><all/></access>
S:      <statement>
S:        <purpose><admin/><prov/></purpose>
S:        <recipient><ours/><public/></recipient>
S:        <retention><stated/></retention>
S:      </statement>
S:    </dcp>
S:  </greeting>
S:</epp>
```

In the example above, the server is announcing the availability of two extensions:

urn:ietf:params:xml:ns:foo-ext1, and

http://custom/obj1ext-1.0

An EPP client MUST establish a session with an EPP server using the EPP <login> command before attempting to use any standard commands or extensions. The <login> command contains zero or more <svcExtension> elements that, if present, contain a URI identifying an available extension that the client wishes to use during the course of the session:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
C:     epp-1.0.xsd">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>foo-BAR2</pw>
C:      <newPW>bar-FOO2</newPW>
C:      <options>
C:        <version>1.0</version>
C:        <lang>en</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>http://custom/obj1ext-1.0</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

In the example above, the client indicates that it wishes to use an extension identified by the http://custom/obj1ext-1.0 URI during the session established upon successful completion of the <login> command.

An EPP server MUST announce all extensions that are publicly available for client use. An EPP client MUST NOT request an extension that has not been announced by the server. An EPP server MAY restrict a client’s ability to select an extension based on a client’s identity and authorizations granted by the server operator.

2.4. Protocol-level Extension

EPP defines a set of structures for client-server command-response interaction, but additional structures MAY be added to the protocol. New structure definition is a matter of defining a schema for the structures that defines needed functionality and assigning a URI to uniquely identify the object namespace and schema. Specific protocol-level extension mechanisms are described in section 2.7.1 of the EPP core protocol specification [2].

2.5. Object-level Extension

EPP commands and responses do not contain attributes that are specific to any managed object. Every command and response MUST contain elements bound to an object namespace. Object definition is a matter of defining a schema for the object that defines functionality for each needed command and associated response, and assigning a URI to uniquely identify the object namespace and schema. Specific object-level extension mechanisms are described in section 2.7.2 of the EPP core protocol specification [2].

2.6. Command-Response Extension

EPP command and response structures defined in existing object mappings MAY also be extended. For example, an object mapping that describes general functionality for the provisioning of Internet domain names can be extended to included additional command and response elements needed for the provisioning of domain names that represent E.164 telephone numbers [12]. Specific command-response extension mechanisms are described in section 2.7.3 of the EPP core protocol specification [2].

2.7. Authentication Information Extension

Some EPP object mappings, such as the Internet domain name mapping [10], include elements to associate authentication information (such as a password) with an object. The schema for any object mapping that supports authentication information SHOULD be flexible enough to specify multiple forms of authentication information. With XML Schema ([4] and [5]), this can be accomplished by offering an element choice that includes an <any> element information item:

```
<any namespace="##other"/>
```

3. Selecting an Extension Mechanism

Extensibility is a powerful feature of XML, but it also provides multiple opportunities to make poor design decisions. There are typically several different ways to accomplish a single task, and while all may "work" (for some definition of "work") one extension form will usually be more appropriate than others to complete a given task. The following sequence of steps can be followed to select an appropriate extension form to solve an extension problem:

- Command-Response Extension: Adding elements to an existing object mapping is the simplest form of extension available, and is thus the form that should be explored before any other form is considered. The first question to ask when considering an extension form is thus:

  > Can the task be accomplished by adding to an existing object mapping or changing an existing object mapping slightly?

  If the answer to this question is "yes", you should consider extending an existing object mapping to complete your task. Knowing where to find object mappings is critical to being able to answer this question; see section Section 3.1 for information describing mapping archives. If the answer to this question is "no", consider an object-level extension next.

- Object-level Extension: If there is no existing object mapping that can be extended to meet your requirements, consider developing a new object mapping. The second question to ask when considering an extension form is thus:

  > Can the task be accomplished using the existing EPP command and response structures applied to a new object?

  If the answer to this question is "yes", you should consider developing a new object mapping to complete your task. A new object mapping should differ significantly from existing object mappings; if you find that a new mapping is replicating a significant number of structures found in an existing mapping you probably answered the command-response question incorrectly. If the answer to this question is "no", consider a protocol-level extension next.

- Protocol-level Extension: If there is no existing object mapping that can be extended to meet your requirements and the existing EPP command and response structures are insufficient, consider

developing new protocol commands, responses, or other structures. The third and final question to ask when considering an extension form is thus:

> Can the task be accomplished by adding new EPP commands, responses, or other structures applied to new or existing objects?

If the answer to this question is "no", EPP can not be used directly to complete your task. If the answer to this question is "yes", extend the protocol by defining new operational structures.

The extension forms and decision points listed here are presented in order of complexity. Selecting an extension form without careful consideration of the available extension options can add complexity without any gain in functionality.

3.1. Mapping and Extension Archives

Existing object mappings are a critical resource when trying to select an appropriate extension form. Existing mappings or extensions can provide a solid basis for further extension, but designers have to know where to find them to consider them for use.

Several organizations maintain archives of XML structures that can be useful extension platforms. These include:

- The IETF: Object mappings and other extensions have been documented in RFCs and Internet-Drafts.
- IANA: Guidelines and registration procedures for an IANA XML registry used by the IETF are described in "The IETF XML Registry" [8].
- OASIS [16]: OASIS maintains an XML archive containing schema definitions for use in the business applications of XML.
- XML.org [17]: XML.org maintains an XML archive containing schema definitions for use in multiple industries.
- Other archives are likely in the future. Consult your favorite Internet search engine for additional resources.

4. Internationalization Considerations

EPP is represented in XML [3], which requires conforming parsers to recognize both UTF-8 [13] and UTF-16 [14]; support for other character encodings is also possible. EPP extensions MUST observe

both the Internationalization Considerations described in the EPP core protocol specification [2] and IETF policy on the use of character sets and languages described in RFC 2277 [9].

5. IANA Considerations

This memo has no direct impact on the IANA. Guidelines for extensions that require IANA action are described in Section 2.2.1.

6. Security Considerations

EPP extensions inherit the security services of the protocol structure that's being extended. For example, an extension of an object mapping inherits all of the security services of the object mapping. Extensions MAY specify additional security services, such as services for peer entity authentication, confidentiality, data integrity, authorization, access control, or non-repudiation. Extensions MUST NOT mandate removal of security services available in the protocol structure being extended.

Protocol designers developing EPP extensions need to be aware of the security threats to be faced in their intended operating environment so that appropriate security services can be provided. Guidelines for designers to consider and suggestions for writing an appropriate Security Considerations section can be found in RFC 3552 [15].

7. References

7.1. Normative 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 (EPP)", RFC 3730, March 2004.

[3] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (2nd ed)", W3C REC-xml, October 2000, <http://www.w3.org/TR/REC-xml>.

[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] Bray, T., Hollander, D. and A. Layman, "Namespaces in XML", W3C REC-xml-names, January 1999, <http://www.w3.org/TR/REC-xml-names>.

[7] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

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

[9] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

7.2. Informative References

[10] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", RFC 3731, March 2004.

[11] Hollenbeck, S., Rose, M. and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", BCP 70, RFC 3470, January 2003.

[12] Hollenbeck, S., "Extensible Provisioning Protocol E.164 Number Mapping", Work in Progress, February 2003.

[13] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

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

[15] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

8. URIs

[16] <http://oasis-open.org/>

[17] <http://xml.org/>

9. Author's Address

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

EMail: shollenbeck@verisign.com

10. Full Copyright Statement

Intellectual Property

Acknowledgement

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