Application
Internet-Draft
Intended status: Informational
Expires: August 29, 2013

Y. Doi
TOSHIBA Corporation
February 25, 2013

# EXI Messaging Requirements
## draft-doi-exi-messaging-requirements-01

Abstract

EXI (Efficient XML Interchange) is a specification on efficient encoding of XML. EXI is useful if an application requires XML based message exchange but no sufficient resource is available. However, schema-informed mode of EXI needs some out-of-band coordination between communicating nodes. This document discusses requirement on use of schema-informed EXI as a message exchange format on the Internet systems.

Status of this Memo



Copyright Notice

Table of Contents

## 1. Introduction

EXI[W3C.REC-exi-20110310] (Efficient XML Interchange) is a specification on efficient encoding of XML. EXI is useful if an application requires XML based message exchange but no sufficient resource is available, such as environments discussed in [I-D.shelby-core-coap-req]. However, EXI may need some out-of-band coordination between communicating nodes.

The target of this document is not to discuss EXI spec itself. This document discusses how to use it as a message exchange format (a presentation layer) on the Internet systems to support development and deployment of EXI systems in the Internet including constrained nodes.

## 2. Schema Update and Data Type Derivation

In communication use cases of XML/EXI, XML schema (or equivalents) is often used to define a standard message format. A schema defines a message format, and such message format is expected to be able to extend. There are at least two ways of extension.

First way is to update the schema. Brand-new devices with a new functionality may have updated schema to support extended message. In this scenario, a system consists of multiple versions of schema. As schema-informed EXI requires communicating nodes to use identical schema, this scenario requires schema negotiation.

Second way is to use derived data types from the schema. Built-in grammar or non-strict schema-informed grammar allow derived XML instances from the definition in the XML schema. To accommodate resource-constrained nodes, an application spec may specify a parameter set with EXI Profile[W3C.WD-exi-profile-20120731].

Schema update and data derivation are not exclusive. Application designers may choose one or both approaches. This is tradeoff between extensiblity and interoperability.

## 3. Schema Negotiation for Strict Schema-Informed EXI Messaging

In short, EXI has two grammar modes: Schema-informed and Built-in. Built-in grammar uses dynamic state machines that learn document structure on-the fly. On the other hand, Schema-informed grammar makes a set of state machines from a schema and the state machines are used to encode/decode document structure. Strict mode of schema-informed grammar uses static state machines for XML elements and

attributes defined in the XML schema. Wildcard elements are handled in the built-in grammar (dynamic state machines). Non-strict mode allows XML data to be derived from that defined in the XML schema.

Because schema-informed grammars can make smallest messages in most cases, some applications may want to make use of schema-informed grammar as its message format.

To decode an EXI message, the sender and receiver must have exactly same schema. However, the way to negotiate and match schema between communicating nodes is not yet well defined.

To use EXI as application message encodings, clients and servers should have a way to coordinate the schema used in the communication. This is similar to content negotiation defined in HTTP[RFC2616]. This section describes schema negotiation cases based on common communication pattern.

## 3.1. Content-Type and Schema Identification

To negotiate schema, an application must have a way to identify a schema.

A content-type may use schema-informed EXI as its encoding. Each content-type should define how to identify a schema used in a communication. The identifier (schemaId) may have internal structure to indicate backward compatibility.

A good practice is to have schema version number (Major.Minor) as a schema ID. Between minor modifications, schemas should have backward compatibility (a node with schema 4.3 shall have schema 4.0, 4.1 and 4.2). Between major modifications, schema should not have it (a node with schema 4.3 may not have schema 1.x, 2.x and 3.x). Note that schemaId is local identifier space that belongs to a content-type. There is no need to have global schema ID registry.

On schema negotiation, a receiver of a message declares a set of acceptable schema IDs and a sender selects a schema ID among the given set. The selected schema ID should be in schemaId field of EXI option header.

## 3.2. Client-Driven Schema Negotiation

Client-driven schema negotiation is the way that a client decides actual schema version used in a communication. This happens in POST or PUT style communications. In [RFC2616], try-and-redirect style of client-driven content negotiation is described. Similar way should be possible in schema negotiation. However, it may be simpler to

have a way to declare a server's acceptable schema set.

As an alternative, a server (or a resource on a server) may declare its available schema set via some service discovery mechanisms. Candidates are such as DNS-SD[I-D.cheshire-dnsext-dns-sd] TXT resource records or media type in link format[I-D.nottingham-http-link-header] that represents a resource. If an application can assume a client does service discovery before using the service, it may assume the client knows server's schema set.

### 3.3. Server-Driven Schema Negotiation

Server-driven schema negotiation is the way that a server decides actual schema version used in a communication. In HTTP, schema negotiation in GET requests should do server-driven negotiation. In [RFC2616], Accept: header is defined to make server-driven content negotiation. Schema negotiation can be piggybacked on it by using some content type parameter to carry acceptable schema ID set.

### 3.4. Publisher-Driven Schema Negotiation

CoAP[I-D.ietf-core-coap] and some other protocols may have publish-subscribe (observer) pattern in communication. In this case, a subscriber should give its acceptable schema ID set to a publisher as it registers its subscription request.

## 4. Security Considerations

No particular security concern is raised by this document. Applications should be able to detect malformed input as usual.

## 5. IANA Considerations

This document has no actions for IANA.

## 6. Normative References

[I-D.cheshire-dnsext-dns-sd]
Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", draft-cheshire-dnsext-dns-sd-11 (work in progress), December 2011.

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank,

"Constrained Application Protocol (CoAP)", draft-ietf-core-coap-11 (work in progress), July 2012.

[I-D.nottingham-http-link-header]
Nottingham, M., "Web Linking", draft-nottingham-http-link-header-10 (work in progress), May 2010.

[I-D.shelby-core-coap-req]
Shelby, Z., Stuber, M., Sturek, D., Frank, B., and R. Kelsey, "CoAP Requirements and Features", draft-shelby-core-coap-req-02 (work in progress), October 2010.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[W3C.REC-exi-20110310]
Kamiya, T. and J. Schneider, "Efficient XML Interchange (EXI) Format 1.0", World Wide Web Consortium Recommendation REC-exi-20110310, March 2011, <http://www.w3.org/TR/2011/REC-exi-20110310>.

[W3C.WD-exi-profile-20120731]
Fablet, Y. and D. Peintner, "Efficient XML Interchange (EXI) Profile", World Wide Web Consortium LastCall WD-exi-profile-20120731, July 2012, <http://www.w3.org/TR/2012/WD-exi-profile-20120731>.

# Author's Address

Yusuke Doi
TOSHIBA Corporation
Komukai Toshiba Cho 1
Saiwai-Ku
Kawasaki, Kanagawa 2128582
JAPAN

Phone: +81-45-342-7230
Email: yusuke.doi@toshiba.co.jp
URI: