ecrit B. Rosen
Internet-Draft April 26, 2019
Updates: RFC5222 (if approved)
Intended status: Standards Track
Expires: October 28, 2019

Validation of Locations Around a Planned Change
draft-ietf-ecrit-lost-planned-changes-03

Abstract

This document defines an extension to LoST (RFC5222) that allows a planned change to the data in the LoST server to occur. Records that previously were valid will become invalid at a date in the future, and new locations will become valid after the date. The extension adds two elements to the <findservice> request: A URI to be used to inform the LIS that previously valid locations will be invalid after the planned change date, and add a date which requests the server to perform validation as of the date specified. It also adds an optional Time-To-Live element to the response, which informs clients about the current expected lifetime of the validation. This document also provides a conventional XML schema for LoST, as backwards compatible alternative to the RelaxNG schema in RFC5222

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes an update to the LoST protocol + which allows a <findservice> request to optionally add a URI and a date to be used with planned changes to the underlying location information in the server. The URI is retained by the LoST server, associated with the data record that was validated, and used to notify the LIS (the LoST client) when a location which was previously valid will become invalid. The date is used by the client to ask the server to perform validation as of a future date. In addition to this mechanism, the <findserviceResponse> is also extended to provide a Time-To-Live (TTL) for validation, after which the client should revalidate the location.

Validation of civic locations involves dealing with data that changes over time. A typical example is a portion of a county or province that was not part of a municipality is "annexed" to a municipality. Prior to the change, the content of the PIDF-LO A3 element would be blank, or represent some other value and after the change would be the municipality that annexed that part of the county/province. This

kind of annexation has an effective date and time (typically 00:00 on the first or last day of a month).

Records in a LIS must change around these kinds of events. The old record must be discarded, and a new, validated record must be loaded into the LIS. It is often difficult for the LIS operator to know that records must be changed around such events. There are other circumstances where locations that were previously valid become invalid, such as a street renaming or renumbering event. As RFC5222 defines validation, the only way for a LIS to discover such changes was to periodically revalidate its entire database. Of course, this would not facilitate timely changes, is not coordinated with the actual change event, and also adds significant load to the LoST server. Even if re-validation is contemplated, the server has no mechanism to control, or even suggest the time period for revalidation

This extension allows the client to provide a stable URI that is retained by the server associated with the location information used in the request. In the event of a planned change, or any other circumstance where the location becomes invalid, the server sends a notification to the URI informing it of a change. The notification contains the date and time when the location becomes invalid.

Ideally, following such a notification, the LIS will prepare a new record to be inserted in its active database, that becomes active at the precise planned event date and time, at which point it would also delete the old record. However, the new record has to be valid, and the LIS would like to validate it prior to the planned change event. If it requests validation before the planned event, the server (without this extension) would inform the client that the location was invalid. This extension includes an optional "as of" date and time in the request that allows the LoST server to provide validation as of the date and time specified, as opposed to the "as of now" implied in the current LoST protocol.

When it is not practical or advisable for the LIS to maintain stable URIs for all of its records, periodic revalidation can be still used to maintain the data in the LIS. However, the server should be able to control the rate of such revalidation. For this purpose, a new TTL element is included in the <findserviceResponse> which provides advice from the server to the LIS of when validation is suggested.

There are quite a few implementations of LoST. Experience with these implementations indicates that the RelaxNG schema is very difficult to deal with, because the tools that the implementors use don't support it, and their staff is unfamiliar with it. Alternative schemas have been circulated, which is undesirable, as they may not

be in conformance to the RelaxNG schema in [RFC5222]. This document provides an XML schema that replaces the RelaxNG schema. It can be used by any implementation interchangeably with the RelaxNG schema.

2. Conventions used in this document

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

"Server" in this document refers to the LoST server and "Client" is the LoST client, even when the server is performing an operation on the client.

3. <plannedChange> element

This document defines a new element to <findService> called "plannedChange". This element contains two attributes: 'uri' and 'asOf'. The 'uri' attribute MUST be a URI with a scheme of HTTPS. The URI will be stored by the server against the location in the request for subsequent use with the notification function defined below. To minimize storage requirements of at the server, the length of the URI MUST be 256 bytes or less. Each client of the server may only store one URI against a location, where "location" is defined by policy at the server, since a given unique location may have many combinations of location elements that resolve to the same location. If the server receives a 'uri' for the same location from the same client, the URI in the request replaces the URI it previously retained. Policy at the server may limit how many URIs it retains for a given location. A new warning is defined below to be used to indicate that the URI has not been stored. If the location in the request is invalid, the URI will not be stored and the warning will be returned.

The 'asOf' attribute contains a date and time. The server will validate the location in the request as of the date specified, taking into account planned changes. This allows the client to verify that it can make changes in the LIS commensurate with changes in the LoST server by validating locations in advance of a change.

4. <locationInvalidated> object

When the server needs to invalidate a location where the client provided a URI in <plannedChange>, the server executes an HTTPS POST containing <locationInvalidated> to the URI previously provided. This is the notice from the server to the client that the location may be invalid and should be revalidated. <locationInvalidated> contains an 'invalidAsOf' attribute that specifies when the location

may become invalid. If the date/time in 'invalidAsOf' is earlier than the time the <locationInvalidated> was sent, the location may already be invalid and the LIS should take immediate action. If the POST operation fails, the server MAY retry the operation immediately, and if it fails again, retry the operation at a later time.

5. URI Not Stored Warning

A new warning is added to the exceptionContainer, 'uriNotStored'. This warning MUST NOT be returned unless the <plannedChange> element was found in the corresponding request. The warning is returned when the server decides not to store the URI found in the <plannedChange> element. As discussed above, this may occur because, among other reasons, the policy at the server limits how many URIs will be stored against a specific location, the URI is not well formed or the policy at the server has some other restriction on the feature.

6. Time-To-Live (TTL) in Response

A new <ttl> element is added to the <findserviceResponse>. The <ttl> element contains a date and time after which the client may wish to revalidate the location at the server. This element MAY be added by the server if validation is requested in the response. The form of the element is the 'expires' attribute pattern, which allows explicit 'NO CACHE' and 'NO EXPIRATION' values to be returned in the <ttl> element. 'NO CACHE' has no meaning and MUST NOT be returned in TTL. 'NO EXPIRATION' means the server does not have any suggested revalidation period.

Selecting a revalidation interval is a complex balancing of timeliness, server load, stability of the underlying data, and policy of the LoST server. Too short, and load on the server may overwhelm it. Too long and invalid data may persist in the server for too long. The URI mechanism provides timely notice to coordinate changes, but even with it, it is often advisable to revalidate data eventually.

In areas that have little change in data, such as fully built out, stable communities already part of a municipality, it may be reasonable to set revalidation periods of 6 months or longer, especially if the URI mechanism is widely deployed at both the server and the clients. In areas that are quickly growing, 20-30 day revalidation may be more appropriate even though such revalidation would be the majority of the traffic on the LoST server.

When a planned change is made, typically the TTL value for the affected records is lowered, so that revalidation is forced soon after the change is implemented. It is not advisable to set the

expiration precisely at the planned change time if a large number of records will be changed, since that would cause a large spike in traffic at the change time. Rather, the expiration time should have a random additional time added to it to spread out the load.

7. Replacement XML schema

The Relax NG schema in [RFC5222] is replaced with the following XML schema:

```
 <?xml version="1.0" encoding="UTF-8"?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns="urn:ietf:params:xml:ns:lost1"
            targetNamespace="urn:ietf:params:xml:ns:lost1"
            elementFormDefault="qualified">

   <xs:element name="findService">
     <xs:complexType>
       <xs:sequence>
         <xs:group ref="requestLocation"/>
         <xs:group ref="commonRequestPattern"/>
       </xs:sequence>
       <xs:attribute name="validateLocation" type="xs:boolean"/>
       <xs:attribute name="serviceBoundary">
         <xs:simpleType>
           <xs:restriction base="xs:token">
             <xs:enumeration value="reference"/>
             <xs:enumeration value="value"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="recursive" type="xs:boolean"/>
     </xs:complexType>
   </xs:element>

   <xs:element name="listServices">
     <xs:complexType>
       <xs:group ref="commonRequestPattern"/>
     </xs:complexType>
   </xs:element>

   <xs:element name="listServicesByLocation">
     <xs:complexType>
       <xs:sequence>
         <xs:group ref="requestLocation"/>
         <xs:group ref="commonRequestPattern"/>
       </xs:sequence>
       <xs:attribute name="recursive" type="xs:boolean"/>
```

```
     </xs:complexType>
   </xs:element>

   <xs:element name="getServiceBoundary">
     <xs:complexType>
       <xs:group ref="extensionPoint"/>
       <xs:attributeGroup ref="serviceBoundaryKey"/>
     </xs:complexType>
   </xs:element>

   <xs:element name="findServiceResponse">
     <xs:complexType>
       <xs:sequence>
         <xs:element ref="mapping" maxOccurs="unbounded"/>
         <xs:element ref="locationValidation" minOccurs="0"/>
         <xs:group ref="commonResponsePattern"/>
         <xs:group ref="locationUsed"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:element name="listServicesResponse">
     <xs:complexType>
       <xs:sequence>
         <xs:element ref="serviceList"/>
         <xs:group ref="commonResponsePattern"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:element name="listServicesByLocationResponse">
     <xs:complexType>
       <xs:sequence>
         <xs:element ref="serviceList"/>
         <xs:group ref="commonResponsePattern"/>
         <xs:group ref="locationUsed"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:element name="getServiceBoundaryResponse">
     <xs:complexType>
       <xs:sequence>
         <xs:group ref="serviceBoundary"/>
         <xs:group ref="commonResponsePattern"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>
```

```
   <xs:group name="commonRequestPattern">
     <xs:sequence>
       <xs:group ref="service"/>
       <xs:element ref="path" minOccurs="0"/>
       <xs:group ref="extensionPoint"/>
     </xs:sequence>
   </xs:group>

   <xs:group name="commonResponsePattern">
     <xs:sequence>
       <xs:element ref="warnings" minOccurs="0" maxOccurs="unbounded"/>
       <xs:element ref="path"/>
       <xs:group ref="extensionPoint"/>
     </xs:sequence>
   </xs:group>

   <xs:group name="requestLocation">
     <xs:sequence>
       <xs:element ref="location" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:group>

   <xs:element name="location">
     <xs:complexType>
       <xs:complexContent>
         <xs:extension base="locationInformation">
           <xs:attribute name="id" type="xs:token" use="required"/>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
   </xs:element>

   <xs:complexType name="locationInformation">
     <xs:group ref="extensionPoint" maxOccurs="unbounded"/>
     <xs:attribute name="profile" type="xs:NMTOKEN"/>
   </xs:complexType>

   <xs:group name="serviceBoundary">
     <xs:sequence>
       <xs:element ref="serviceBoundary" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:group>

   <xs:element name="serviceBoundary" type="locationInformation"/>

   <xs:element name="serviceBoundaryReference">
     <xs:complexType>
       <xs:group ref="extensionPoint"/>
```

```
       <xs:attributeGroup ref="source"/>
       <xs:attributeGroup ref="serviceBoundaryKey"/>
     </xs:complexType>
   </xs:element>

   <xs:attributeGroup name="serviceBoundaryKey">
     <xs:attribute name="key" type="xs:token" use="required"/>
   </xs:attributeGroup>

   <xs:element name="path">
     <xs:complexType>
       <xs:sequence>
         <xs:element ref="via" maxOccurs="unbounded"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:element name="via">
     <xs:complexType>
       <xs:group ref="extensionPoint"/>
       <xs:attributeGroup ref="source"/>
     </xs:complexType>
   </xs:element>

   <xs:group name="locationUsed">
     <xs:sequence>
       <xs:element ref="locationUsed" minOccurs="0"/>
     </xs:sequence>
   </xs:group>

   <xs:element name="locationUsed">
     <xs:complexType>
       <xs:attribute name="id" type="xs:token" use="required"/>
     </xs:complexType>
   </xs:element>

   <xs:attributeGroup name="expires">
     <xs:attribute name="expires" use="required">
       <xs:simpleType>
         <xs:union memberTypes="xs:dateTime">
           <xs:simpleType>
             <xs:restriction base="xs:token">
               <xs:enumeration value="NO-CACHE"/>
             </xs:restriction>
           </xs:simpleType>
           <xs:simpleType>
             <xs:restriction base="xs:token">
               <xs:enumeration value="NO-EXPIRATION"/>
```

```
             </xs:restriction>
           </xs:simpleType>
         </xs:union>
       </xs:simpleType>
     </xs:attribute>
   </xs:attributeGroup>

   <xs:simpleType name="qnameList">
     <xs:list itemType="xs:QName"/>
   </xs:simpleType>

   <xs:element name="mapping">
     <xs:complexType>
       <xs:sequence>
         <xs:element ref="displayName"
              minOccurs="0" maxOccurs="unbounded"/>
         <xs:group ref="service"/>
         <xs:choice minOccurs="0">
           <xs:group ref="serviceBoundary"/>
           <xs:element ref="serviceBoundaryReference"/>
         </xs:choice>
         <xs:element ref="uri"
              minOccurs="0" maxOccurs="unbounded"/>
         <xs:element ref="serviceNumber" minOccurs="0"/>
         <xs:group ref="extensionPoint"/>
       </xs:sequence>
       <xs:attributeGroup ref="expires"/>
       <xs:attribute name="lastUpdated" type="xs:dateTime"
             use="required"/>
       <xs:attributeGroup ref="source"/>
       <xs:attribute name="sourceId" type="xs:token"
             use="required"/>
       <xs:attributeGroup ref="message"/>
     </xs:complexType>
   </xs:element>

   <xs:element name="displayName">
     <xs:complexType>
       <xs:simpleContent>
         <xs:extension base="xs:string">
           <xs:attribute ref="xml:lang" use="required"/>
         </xs:extension>
       </xs:simpleContent>
     </xs:complexType>
   </xs:element>

   <xs:element name="uri" type="xs:anyURI"/>
```

```
   <xs:element name="serviceNumber">
     <xs:simpleType>
       <xs:restriction base="xs:token">
         <xs:pattern value="[0-9*#]+"/>
       </xs:restriction>
     </xs:simpleType>
   </xs:element>

   <xs:element name="locationValidation">
     <xs:complexType>
       <xs:sequence>
         <xs:element ref="valid" minOccurs="0"/>
         <xs:element ref="invalid" minOccurs="0"/>
         <xs:element ref="unchecked" minOccurs="0"/>
         <xs:group ref="extensionPoint"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:element name="valid" type="qnameList"/>

   <xs:element name="invalid" type="qnameList"/>

   <xs:element name="unchecked" type="qnameList"/>

   <xs:complexType name="exceptionContainer">
     <xs:sequence>
       <xs:choice minOccurs="0" maxOccurs="unbounded">
         <xs:element ref="badRequest"/>
         <xs:element ref="internalError"/>
         <xs:element ref="serviceSubstitution"/>
         <xs:element ref="defaultMappingReturned"/>
         <xs:element ref="forbidden"/>
         <xs:element ref="notFound"/>
         <xs:element ref="loop"/>
         <xs:element ref="serviceNotImplemented"/>
         <xs:element ref="serverTimeout"/>
         <xs:element ref="serverError"/>
         <xs:element ref="locationInvalid"/>
         <xs:element ref="locationProfileUnrecognized"/>
       </xs:choice>
       <xs:group ref="extensionPoint"/>
     </xs:sequence>
     <xs:attributeGroup ref="source"/>
   </xs:complexType>

   <xs:element name="errors" type="exceptionContainer"/>
```

```
   <xs:element name="warnings" type="exceptionContainer"/>

   <xs:complexType name="basicException">
     <xs:annotation>
       <xs:documentation>
         Exception pattern.
       </xs:documentation>
     </xs:annotation>
     <xs:group ref="extensionPoint"/>
     <xs:attributeGroup ref="message"/>
   </xs:complexType>

   <xs:element name="badRequest" type="basicException"/>

   <xs:element name="internalError" type="basicException"/>

   <xs:element name="serviceSubstitution" type="basicException"/>

   <xs:element name="defaultMappingReturned" type="basicException"/>

   <xs:element name="forbidden" type="basicException"/>

   <xs:element name="notFound" type="basicException"/>

   <xs:element name="loop" type="basicException"/>

   <xs:element name="serviceNotImplemented" type="basicException"/>

   <xs:element name="serverTimeout" type="basicException"/>

   <xs:element name="serverError" type="basicException"/>

   <xs:element name="locationInvalid" type="basicException"/>

   <xs:element name="locationValidationUnavailable"
           type="basicException"/>

   <xs:element name="locationProfileUnrecognized">
     <xs:complexType>
       <xs:complexContent>
         <xs:extension base="basicException">
           <xs:attribute name="unsupportedProfiles"
               type="xs:NMTOKENS" use="required"/>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
   </xs:element>
```

```
   <xs:element name="redirect">
     <xs:complexType>
       <xs:group ref="extensionPoint"/>
       <xs:attribute name="target" type="appUniqueString"
           use="required"/>
       <xs:attributeGroup ref="source"/>
       <xs:attributeGroup ref="message"/>
     </xs:complexType>
   </xs:element>

   <xs:attributeGroup name="message">
     <xs:attribute name="message" type="xs:token"/>
     <xs:attribute ref="xml:lang"/>
   </xs:attributeGroup>

   <xs:group name="service">
     <xs:sequence>
       <xs:element ref="service" minOccurs="0"/>
     </xs:sequence>
   </xs:group>

   <xs:element name="service" type="xs:anyURI"/>

   <xs:simpleType name="appUniqueString">
     <xs:restriction base="xs:token">
       <xs:pattern value="([a-zA-Z0-9\-]+\.)+[a-zA-Z0-9]+"/>
     </xs:restriction>
   </xs:simpleType>

   <xs:attributeGroup name="source">
     <xs:attribute name="source" type="appUniqueString" use="required"/>
   </xs:attributeGroup>

   <xs:element name="serviceList">
     <xs:simpleType>
       <xs:list itemType="xs:anyURI"/>
     </xs:simpleType>
   </xs:element>

   <xs:group name="notLost">
     <xs:annotation>
       <xs:documentation>
         Any element not in the LoST namespace.
       </xs:documentation>
     </xs:annotation>
     <xs:choice>
       <xs:any namespace="##other" processContents="skip"/>
       <xs:any namespace="##local" processContents="skip"/>
```

```
     </xs:choice>
   </xs:group>

   <xs:group name="anyElement">
     <xs:annotation>
       <xs:documentation>
         A wildcard pattern for including any element
         from any other namespace.
       </xs:documentation>
     </xs:annotation>
     <xs:sequence>
       <xs:any processContents="skip"
           minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:group>

   <xs:attributeGroup name="anyElement">
     <xs:annotation>
       <xs:documentation>
         A wildcard pattern for including any element
         from any other namespace.
       </xs:documentation>
     </xs:annotation>
     <xs:anyAttribute processContents="skip"/>
   </xs:attributeGroup>

   <xs:group name="extensionPoint">
     <xs:annotation>
       <xs:documentation>
         A point where future extensions
         (elements from other namespaces)
         can be added.
       </xs:documentation>
     </xs:annotation>
     <xs:sequence>
       <xs:group ref="notLost"
            minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:group>

 </xs:schema>
```

8. Extension XML Schema

This schema provides the extension to the prior section schema for planned changes:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="urn:ietf:params:xml:ns:lostPlannedChange1"
           targetNamespace="urn:ietf:params:xml:ns:lostPlannedChange1"
           elementFormDefault="qualified">
     <!-- Import base Lost -->
     <xs:import namespace="urn:ietf:params:xml:ns:lost1"/>
<!-- extend the extensionPoint of commonRequestPattern of findService
                                                       to include:  -->
        xs:group ref="plannedChange" minOccurs="1"/>

<!-- where -->
       <xs:group name="plannedChange">
           <xs:sequence>
              <xs:element ref="uri" type="xs:anyURI" />
              <xs:element ref="invalidAsOf" type="xs:dateTime" />
              <xs:any namespace="##other" processContents="lax"
                 minOccurs="0" maxOccurs="unbounded"/>
           </xs:sequence>
        </xs:group>

<!-- extend the extensionPoint of commonResponsePattern in
                          findServiceResponse to include: -->
           <xs:atttribute name="asOf" type="xs:dateTime" minOccurs="0"/>
           <xs:element ref="ttl" type="expires" minOccurs="0" />

<!-- extend the extensionPoint of extensionContainer to include: -->
           <xs:element ref="uriNotStored"/>
<!-- where -->
           <xs:element name="uriNotStored" type="basicException"/>

</xs:schema>
```

9. Security Considerations

As an extension to LoST, this document inherits the security issues raised in [RFC5222]. The server could be tricked into storing a malicious URI which, when sent the locationInvalidated object could trigger something untoward. The server MUST NOT accept any data from the client in response to POSTing the locationInvalidated.

The server is subject to abuse by clients because it is being asked to store something and may need to send data to an uncontrolled URI. Clients could request many URIs for the same location for example. The server MUST have policy that limits use of this mechanism by a given client. If the policy is exceeded, the server returns the 'uriNotStored' warning. The server MUST validate that the content of the 'uri' attribute sent is syntactically valid and meets the 256

bytes limit. When sending the locationInvalidated object to the URI stored, the server MUST protect itself against common HTTP vulnerabilities.

The mutual authentication between client and server is RECOMMENDED for both the initial <findService> operation that requests storing the URI and the sending of the 'locationInvalidated' object. The server should be well known to the client, and its credential should be learned in a reliable way. For example, a public safety system operating the LoST server may have a credential traceable to a well known Certificate Authority known to provide credentials for public safety agencies. Many of the clients will be operated by local ISPs or other service providers where the server operator can reasonably obtain a good credential to use for the URI. Where the server does not recognize the client, its policy MAY limit the use of this feature beyond what it would limit a client it recognized.

10. IANA Considerations

10.1. Replacement XML Schema Registration

URI: urn:ietf:params:xml:schema:lost3
Registrant Contact: IETF ECRIT Working Group, Brian Rosen
(br@brianrosen.net).

The XML Schema is found in Section 7.

10.2. Extension XML Schema Registration

URI: urn:ietf:params:xml:schema:lostPlannedChange1
Registrant Contact: IETF ECRIT Working Group, Brian Rosen
(br@brianrosen.net).

The XML Schema is found in Section 8.

10.3. LoST Namespace Registration

URI: urn:ietf:params:xml:ns:lost-plannedChange1

Registrant Contact: IETF ECRIT Working Group, Brian Rosen (br@brianrosen.net).

XML:

```
BEGIN
<?xml version="2.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1"/>
  <title>LoST Planned Change Namespace</title>
</head>
<body>
  <h1>Namespace for LoST Planned Change extension</h1>
  <h2>urn:ietf:params:xml:ns:lost-plannedChange1</h2>
<p>See <a href="http://www.rfc-editor.org/rfc/rfc????.txt">
   RFC????</a>.</p>
</body>
</html>
END
```

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5222] Hardie, T., Newton, A., Schulzrinne, H., and H. Tschofenig, "LoST: A Location-to-Service Translation Protocol", RFC 5222, DOI 10.17487/RFC5222, August 2008, <https://www.rfc-editor.org/info/rfc5222>.

Author's Address

Brian Rosen
470 Conrad Dr
Mars, PA 16046
US

EMail: br@brianrosen.net