

ecrit
Internet-Draft
Intended status: Standards Track
Expires: 1 May 2026

B. Rosen
28 October 2025

Validation of Locations Around a Planned Change
draft-ietf-ecrit-lost-planned-changes-13

Abstract

This document defines an extension to the Location to Service Translation (LoST) protocol (RFC5222) that allows a LoST server to notify a client of planned changes to location data. This extension is only useful with the validation function of LoST. It is beneficial for LoST validation clients to be aware of planned changes, since at a known future date, previously valid records may become invalid, and new records may become valid. This extension adds an element to the <findService> request to allow a LoST client to request validation as of a specified date. It adds an optional Time-To-Live element to the response, which informs clients of the current expected lifetime of a validation. It also adds a separate interface to a LoST server that allows a client to poll for planned changes. Additionally, this document provides a conventional XML schema for LoST, as a backwards compatible alternative to the RelaxNG schema in RFC5222.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on 1 May 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Conventions used in this document	4
3. Planned Change Poll Interface	4
4. <asOf> Element	6
5. <revalidateAfter> in Response	7
6. Replacement XML schema	8
7. Extension XML Schema	16
8. plannedChangePoll Interface Description	17
9. Security Considerations	20
10. IANA Considerations	20
10.1. Application Service Tag	20
10.2. LoST XML Schema Registration	20
10.3. Planned Change Extension XML Schema Registration	20
11. Normative References	21
Author's Address	21

1. Introduction

Validation of civic locations involves dealing with data that may change over time. A typical example is when a portion of a county or district that was not part of a municipality is "annexed" to a municipality. Prior to the change, a Presence Information Data Format Location Object (PIDF-LO) [RFC5139] specifying a civic location in the affected area would have a blank A3 element, or would contain some other value; after the change, a PIDF-LO specifying the same location would contain an A3 element set to the name of the municipality that annexed that part of the county/district. This kind of annexation has an effective date and time (typically 00:00 on the first or last day of a month), known in advance. Other kinds of changes may also occur, and these will almost always also have an effective date that is known in advance.

Records in a LoST client such as a Location Information Server (LIS) must change around these kinds of events. Each affected old record must be discarded, and a new, validated record must be loaded into the client's database. It is often difficult for a 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. Using [RFC5222] validation, the only way for a LoST client such as a LIS to discover such changes is to periodically revalidate its entire database. Of course, this does not facilitate timely changes, is not coordinated with the actual change event, and also adds significant load to LoST servers as well as LoST clients such as LISSs. Even when re-validation is used, a server has no mechanism to control, or suggest the time period for revalidation.

This extension allows a LoST client to obtain from a LoST server sets of locations which may change. It makes use of "partial location information" which is a set of location elements and values that, when compared against client location records, identify which of the client records may change as a result of the planned change. A set of such partial locations is termed a "ChangeSet". ChangeSets have an ID, and a date when the change is effective. IDs are ordered so that changes can be completed in the correct order. The planned change interface is a REST/JSON interface that allows a client to poll a server using the last ID that it obtained from that server. The response to a poll is a list of all the newer planned changes the server knows about beyond the ChangeSet whose ID was included in the poll. The ID of the last ChangeSet in the returned list is used by the client in its next poll.

When a LoST client such as a LIS receives a new ChangeSet, it may prepare one or more new records so that, at the precise planned event date and time, it may insert the new records into its active database and delete the old records. As part of preparing the new records in advance of the change, a client may use the "asOf" date component of this extension to perform a LoST validation of the new record as of the effective date.

The "asOf" date component of this extension allow a LoST client such as a LIS to be prepared for and smoothly transition to planned changes. The polling mechanism allows a client to be informed of planned changes, while the "asOf" date allows it to verify the validity of locations before they become active.

Unplanned changes will occur, and periodic revalidation can still be used to maintain the data in the client's records. However, LoST servers should be able to influence the rate of such revalidation. For this purpose, this extension adds an optional "expires" element to the <findServiceResponse> to provide advice from a server to a client as to when revalidation is suggested. In a <findServiceResponse>, a LoST server may include the "expires" element to expressly state when the location should be re-validated. This avoids clients blindly revalidating every address on an arbitrary schedule.

There are quite a few implementations of LoST. Experience with these implementations indicates that the RelaxNG schema is very difficult to deal with, both because many commonly used development tools don't support it, and development staff is often unfamiliar with it. Informal alternative schemas have been circulated, which is undesirable as they may not be in conformance with 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

"Server" in this document refers to a LoST server and "Client" is a LoST client.

3. Planned Change Poll Interface

This document defines a new interface to a LoST server. The interface has three endpoints:

1. Versions, which returns the current version(s) the interface supports. This allows the interface to evolve over time.
2. PlannedChangePoll, which supports a polling mechanism. The poll returns a list of changeSetIds which identify ChangeSet objects.
3. GetChangeSet, which accepts a changeSetId and returns the identified ChangeSet object.

A ChangeSet object contains an identifier (changeSetId), a date (changeSetEffective) and an array of partial locations.

A partial location is an array of location information element namespace/name and value pairs. A LoST client (such as a LIS) compares the location elements with its records. For each of the client's records where all of the location elements provided in the partial location have the same values as those in the partial location, that client record may be affected by the planned change. The client's records may have other location elements, but those are not considered in the comparison. So, for example, a partial location may have Country, A1, A2, A3 and A4 location elements, which means that any address with those Country, A1, A2, A3 and A4 values may be affected by the planned change, regardless of street name and number. As another example, a partial location with Country, A1, A2, A3, A4, RD and POD but not HN applies to any address number on the specified street.

Servers supporting this extension maintain an ordered list of changeSetIds. A changeSetId is a string, which might not show the ordering. For example, the ID could be a hashed timestamp, or a hashed sequence number, among other things. Given a changeSetId returned in a prior poll, a server can identify which ChangeSets it has that come next, in order, after the specified changeSetId. The effective date of a ChangeSet returned by a server need not be in the future. Tardy clients might not keep up. On the other hand, a server is not obligated to keep ChangeSets whose changeSetEffective date has passed by more than some arbitrary time. A 12 month time period may be appropriate for a server whose service area doesn't have many changes, while a three month time period may be needed in a service area with frequent changes. A tardy client in a fast-changing environment might receive a large number of ChangeSetIds in response to a poll.

Polls are expected every few minutes. A new client (or one that has lost its last changeSetID) performs a poll without specifying an ID, and the server responds with all the changeSetIds that it knows about. Thereafter, the client retains the last changeSetId from its most recent poll and uses that in the next poll. If the response to that poll contains no changeSetIds, it means the changeSetId the client has is the latest change the server knows about; the client uses the same changeSetId in subsequent polls until the server returns a new changeSetId.

The version mechanism returns a list of versions of the web service the server supports. This document specifies version 1.0. Versions are described as a major version and a minor version, where major and minor versions are integers, and a version description is a string consisting of the major version, a dot, and the minor version. A backwards-compatible change within a major version MAY increment only the minor version number. A non-backwards-compatible change MUST

increment the major version number. To achieve backwards compatibility, implementations MUST ignore any object members they do not recognize. Minor version specifications SHALL only add objects, non-required members of existing objects, and non-mandatory-to-use functions and SHALL NOT delete any objects, members of objects or functions. This means an implementation of a specific major version and minor version is backwards compatible with all minor versions of the major version. The versions mechanism returns an array of supported versions, one for each major version supported, with the minor version listed being the highest supported minor version. When versions are written or used as a string, the major version is first and separated from the minor version with a period. For example major version 3, minor version 2 is written as "3.2".

The normative definition of the web interface for these endpoints is contained in Section 8

Discovering the Planned Change Poll interface uses the "lost+plannedChange" application tag with U-NAPTR using the name (application unique string) of the LoST server that provided validation as input along with this new service tag. Similarly to LoST, valid protocol tags for use in combination with "lost+plannedChange" are "http" and "https". The U-NAPTR responds with the base URI of the interface and the client must use the value to replace "localhost" in the yaml in Section 8.

4. <asOf> Element

This document defines a new element in the <findService> request and <findServiceResponse> called <asOf>, which contains a date and time in dateTime format with a required timezone. A server supporting this extension validates the location in the request as of the date and time specified, taking into account planned changes. This allows a client to verify in advance that it can make changes in its records in accordance with changes at the LoST server.

A server responds with what it knows (as of the moment of the request) will be in effect on or before the <asOf> date

The LoST server is not expected to retain a history of changes. Therefore, an `<asOf>` date in the past will return the same results as omitting it (meaning current data). There are no constraints on how far in advance planned changes may occur, and changes to the data may occur at any time. Therefore, two queries with the same future `<asOf>` value placed at different times can return different results. The server responds with its current understanding of planned changes (including any planned or unplanned changes that have already occurred) and makes no guarantees about future queries of the same location.

When a server responds with a result `<asOf>` a time other than the time of the query, the `<findServiceResponse>` MUST include `<asOf>` containing the time used and each `<mapping>` included MUST have the 'expires' attribute set to "NO-CACHE". A server SHOULD NOT include `<asOf>` in the result if `<asOf>` was not requested or if it otherwise used the current time to formulate the response. A client receiving a response containing `<asOf>` MUST NOT assume that any mappings returned will remain unchanged and be in effect in the future; a client wishing to contact a service is still expected to use LoST contemporaneously with that need.

5. `<revalidateAfter>` in Response

This extension defines a new element `<revalidateAfter>` to be added to the extension point of the `<locationValidation>` element appearing in a `<findServiceResponse>`. The element contains a date and time after which the client may wish to revalidate the location with the server. Alternatively, the element may contain the value "NO-EXPIRATION" which is an assertion that the server is not currently aware of any planned changes that would affect the future validation result, but makes no suggestion as to when the client should revalidate. A server MAY add this element to a response when validation is requested. A server is not expected to add this element when the client is asking for validation `<asOf>` a specified time significantly into the future.

Selecting a revalidation interval is a complex balancing of timeliness, server load, stability of the underlying data, and policy at the LoST server. Too short, and load on the server may be undesirably large. Too long, and invalid data may persist in clients for unacceptable lengths of time. The polling mechanism provides timely notice to synchronize changes, but even with it, it is advisable for clients to periodically revalidate their records. 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 six months or longer. In areas that are quickly growing, 20-30 day revalidation may be more appropriate, even though these revalidations might be a majority of the traffic at the LoST server.

6. Replacement XML schema

=====

This schema is an alternative to The Relax NG schema in [RFC5222]. Future extensions to LoST are expected to use this schema as the base for the extensions, rather than the Relax NG schema. Existing extensions described using the Relax NG schema continue to be valid.

```
<?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:import namespace="http://www.w3.org/XML/1998/namespace" />

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

```
        <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: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: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:choice>
  </xs:sequence>
</xs:complexType>
```

```
<xs:element ref="notFound"/>
<xs:element ref="loop"/>
<xs:element ref="serviceNotImplemented"/>
<xs:element ref="serverTimeout"/>
<xs:element ref="serverError"/>
<xs:element ref="SRSInvalid"/>
<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="SRSInvalid" type="basicException"/>

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

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

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

<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="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>
```

7. 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"
  targetNamespace="urn:ietf:params:xml:ns:lostPlannedChange1"
  elementFormDefault="qualified">

  <!-- extend the extensionPoint of commonRequestPattern of findService
        and findServiceResponse to include: -->
    <xs:element name="asOf" type="xs:dateTime" />

  <!-- extend the extensionPoint of locationValidation to include: -->
    <xs:element name="revalidateAfter">
      <xs:simpleType>
        <xs:union memberTypes="xs:dateTime">
          <xs:simpleType>
            <xs:restriction base="xs:token">
              <xs:enumeration value="NO-EXPIRATION"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:element>
</xs:schema>

```

8. plannedChangePoll Interface Description

```

openapi: 3.0.1
info:
  title: LoST plannedChange
  version: "1.0"
servers:
  - url: https://localhost/LoST/v1
variables:
  baseUrl:
    default: https://localhost
    description: The base URI that is the output
                  of U-NAPTR resolution
paths:
  /PlannedChangePoll:
    get:
      summary: Get a list of planned changeSetIds
      operationId: getPlannedChangeIds
      responses:
        '200':
          description: Planned Changes
          content:
            application/json:
              schema:
                $ref:

```

```
        '#/components/schemas/PlannedChangeIdList'
/GetChangeSet:
  get:
    summary: Get a ChangeSet
    operationId: getChangeSet
    parameters:
      - in: query
        name: changeSetId
        schema:
          type: string
        description: ID of a ChangeSet
    responses:
      '200':
        description: return ChangeSet object
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ChangeSet'
/Versions:
  servers:
    - url: {baseUri}/LoST
      variables:
        baseUri:
          default: https://localhost
          description: The base URI that is the output
            of U-NAPTR resolution
  get:
    summary: Retrieves all supported versions
    operationId: RetrieveVersions
    responses:
      '200':
        description: Versions supported
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/VersionsArray'
components:
  schemas:
    PlannedChangeIdList:
      type: array
      items:
        type: string
    ChangeSet:
      type: object
      properties:
        changeSetId:
          type: string
          description: ID of the ChangeSet
```

```
changeSetEffective:
  type: string
  format: datetime
  description: date and time change will come into
    effect in dateTime format with a required
    timezone
partialLocationList:
  type: array
  items:
    type: object
    properties:
      namespace:
        type: string
        description: namespace of CAtype name from IANA registry
      caType:
        type: string
        description: CAtype name from IANA registry
      value:
        type: string
        description: value for this caType
VersionsArray:
  type: object
  required:
    - versions
  properties:
    versions:
      type: array
      items:
        type: object
        required:
          - major
          - minor
        properties:
          major:
            type: integer
            format: int32
            description: Version major number
          minor:
            type: integer
            format: int32
            description: Version minor number
```

9. Security Considerations

As an extension to LoST, this document inherits the security issues raised in [RFC5222]. Clients could poll at a rate which could affect other processing at the LoST server. This is not unlike any other type of denial of service attack at an HTTP server and similar mitigations are necessary.

10. IANA Considerations

10.1. Application Service Tag

This document registers the following U-NAPTR application service tag:

Application Service Tag: LoST-PlannedChange

Defining Publication: The specification contained within this document.

10.2. LoST XML Schema Registration

IANA is requested to add this document to the list of references for the lost1 entry in the schema subregistry of the IETF XML Registry

10.3. Planned Change Extension XML Schema Registration

```
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 Changes</h1>
  <h2>urn:ietf:params:xml:ns:lostPlannedChange1</h2>
  <p>See <a href="http://www.rfc-editor.org/rfc/rfc?????.txt">
    RFC?????</a>.</p>
</body>
</html>
END
```

The XML Schema is found in Section 7.

11. Normative References

- [RFC5139] Thomson, M. and J. Winterbottom, "Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO)", RFC 5139, DOI 10.17487/RFC5139, February 2008, <<https://www.rfc-editor.org/rfc/rfc5139>>.
- [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/rfc/rfc5222>>.
- [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/rfc/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.

Author's Address

Brian Rosen
Email: br@brianrosen.net