

ecrit
Internet-Draft
Updates: [RFC5222](#) (if approved)
Intended status: Standards Track
Expires: 14 April 2022

B. Rosen
11 October 2021

Validation of Locations Around a Planned Change
draft-ietf-ecrit-lost-planned-changes-05

Abstract

This document defines an extension to the Location to Service Translation (LoST) protocol ([RFC5222](#)) that allows planned changes to the data to be handled smoothly. This extension is only useful with the validation function of LoST. It is beneficial for LoST validation clients to be aware of planned changes, as records that previously were valid may become invalid at a known future date, and new locations may become valid after the date. This extension adds an element to the <findService> request: a date that allows the LoST client to request that the server perform validation as of the date specified. 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 the 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 14 April 2022.

Internet-Draft

LoST Planned Changes

October 2021

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Conventions used in this document	4
3.	Planned Change Poll Interface	4
4.	<plannedChange> element	6
5.	Time-To-Live (TTL) in Response	6
6.	Replacement XML schema	7
7.	Extension XML Schema	15
8.	plannedChangePoll Interface Description	16
9.	Security Considerations	18
10.	IANA Considerations	19
10.1.	Replacement XML Schema Registration	19
10.2.	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) 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 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. Using [\[RFC5222\]](#) validation, the only way for 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 the LoST server as well as the LIS. Even if re-validation is contemplated, the server has no mechanism to control, or even suggest the time period for revalidation.

This extension allows a LoST client to obtain from the 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 the client's location records, identify which of the clients 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. The planned change interface is a REST/JSON interface that allows the client to poll the server using the last ID that it obtained from that server. The response to the 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 for the last ChangeSet in the returned list will be used by the client for the next poll.

When 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, the LIS may use the "as of" 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 in a <FindService> request allow a LIS to be prepared for and smoothly transition to planned changes, without the overhead and delay inherent in performing periodic revalidation of each location. The notification URI allows a LIS to be alerted to planned changes, while the "as of" date allows the LIS to verify the validity of locations before they become active.

In situations where it is not practical or advisable for the LIS to maintain a stable URI for each of its records, periodic revalidation can still be used to maintain the data in the LIS. However, the LoST server should be able to influence the rate of such revalidation.

For this purpose, this extension adds a Time to Live (TTL) attribute to 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, 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", "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.](#) Planned Change Poll Interface

This document defines a new interface to the LoST server. The interface has three entry points. One, Versions, returns the current version(s) the interface supports. This allows the interface to

evolve over time. Another entry point, `PlannedChangePoll`, is a poll. The poll returns a list of `changeSetIds` which identify `ChangeSet` objects. The third, `GetChangeSet`, accepts a `changeSetId` and returns the `ChangeSet` object which contains the identifier (`changeSetId`), a date (`changeSetEffective`) and an array of partial locations. A partial location is an array of location information element name and value pairs. The client compares the location elements with its records. For each of the clients 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 a Country, A1, A2, A3 and A4 location elements, which means that any address with that 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 means any address number on the specified street.

The `changeSetId` is string, which the server maintains as an ordered list of `changeSetIds`. The id itself may not show the ordering. For example, it could be a hashed timestamp, or a hashed sequence number. Given a `changeSetId` returned by it in a prior poll, the server can identify which `ChangeSets` it has that come after, in order, after the one with the proffered `changeSetId`. A new client does not know any ids, or a client may lose the id that it had. The client would poll omitting the `changeSetId` in the poll query, and in response, the server returns all the `ChangeSets` it knows about. The effective date of a `ChangeSet` returned by the server need not always be in the future. Tardy clients may not keep up. On the other hand, the server is not obligated to keep change sets whose `changeSetEffective` date has past for more than some arbitrary time. A 12 month time period may be appropriate for a server whose service area doesn't have many changes, where a 3 month time period may be needed in a fast changing service area where many changes occur regularly. A tardy client in a fast changing environment may receive a large number of `ChangeSets` in response to a poll.

Polls are expected every few minutes. A new client omits the ID in its first poll, and the server responds with all the `changeSetIds` that it knows about. Thereafter, the client retains the last

changeSetId in its most recent poll and uses that in the next poll. If the response to that poll is no changeSetIds, it means the changeSetId the client has is the latest change the server knows about, and that same changeSetId will be used in subsequent polls until the server returns a new one.

The version mechanism returns a list of versions of the web service it supports. This document describes version 1.0. Versions are described as a major version, the period "." and a minor version, where major and minor versions are integers. 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 implement. Minor version definitions 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.

[4.](#) <plannedChange> element

This document defines a new element in the <findService> request called "plannedChange". This element contains an attribute: 'asOf'. The 'asOf' attribute contains a date and time in dateTime format with a required timezone. The server validates the location in the request as of the date and time specified, taking into account planned changes. This allows a 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.

[5.](#) Time-To-Live (TTL) in Response

This extension adds the 'ttl' element to the <findServiceResponse>. The 'ttl'; element contains a date and time after which the client

may wish to revalidate the location at the server. A server MAY add this attribute to the response if validation is requested. This element takes the form of the 'expires' attribute pattern of [\[RFC5222\]](#), which allows the values "NO-CACHE" or "NO-EXPIRATION" to be returned instead of a dateTime value. However, for the 'ttl' attribute "NO-CACHE" has no meaning and MUST NOT be returned. "NO-EXPIRATION" means the server does not have a 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 unacceptable lengths of time. The URI notification 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.

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

```
<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>

```

Rosen

Expires 14 April 2022

[Page 10]

Internet-Draft

LoST Planned Changes

October 2021

```

    </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>

```

Rosen

Expires 14 April 2022

[Page 11]

Internet-Draft

LoST Planned Changes

October 2021

```

  <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="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="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"

```

```

    </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>

```

[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"
            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="asOf" type="xs:dateTime" minOccurs="1"/>
        <xs:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:group>

  <!-- extend the extensionPoint of commonResponsePattern in
                                findServiceResponse to include: -->
    <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>
```

8. plannedChangePoll Interface Description

```
openapi: 3.0.1
info:
  title: LoST plannedChange
  version: "1.0"
servers:
  - url: http://localhost/LoST/v1
paths:
  /PlannedChangePoll:
    get:
      summary: Get a list of planned changeSetIds
      operationId: getPlannedChangeIds
```

responses:

Rosen

Expires 14 April 2022

[Page 16]

Internet-Draft

LoST Planned Changes

October 2021

```
    '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: changeSet object
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ChangeSet'
  /Versions:
    servers:
      - url: https://api.example.com/rum
        description: Override base path for Versions query
    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:
        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](#)]. The server could be tricked into storing a malicious URI which, when sent the `revalidateLocation` object could trigger something untoward. The server **MUST NOT** accept any data from the client in response to POSTing the `revalidateLocation`.

Rosen

Expires 14 April 2022

[Page 18]

Internet-Draft

LoST Planned Changes

October 2021

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 `<revalidateLocation>` 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 `<findServiceRequest>` operation that requests storing the URI and the sending of the `<revalidateLocation>` 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. Clients may be operated by local ISPs or other service providers that can reasonably obtain a good credential to use for the server side of the LoST server's POST transaction using the URI. Where the LoST server does not recognize the client, its policy **MAY** limit the use of this feature beyond what it would limit a client it recognizes.

[10.](#) IANA Considerations

10.1. Replacement XML Schema Registration

Rosen

Expires 14 April 2022

[Page 19]

Internet-Draft

LoST Planned Changes

October 2021

URI: urn:ietf:params:xml:schema:lost3

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 Namespace</title>

</head>

<body>

<h1>Namespace for LoST</h1>

<h2>urn:ietf:params:xml:ns:lost3</h2>

<p>See
RFC?????.</p>

```
</body>
</html>
END
```

The XML Schema is found in [Section 6](#).

[10.2](#). Planned Change Extension XML Schema Registration

Rosen

Expires 14 April 2022

[Page 20]

Internet-Draft

LoST Planned Changes

October 2021

URI: urn:ietf:params:xml:schema:lostPlannedChange1
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 </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

- [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
United States of America

Email: br@brianrosen.net