Network Working Group
Request for Comments: 5222
Category: Standards Track

T. Hardie
Qualcomm, Inc.
A. Newton
American Registry for Internet Numbers
H. Schulzrinne
Columbia University
H. Tschofenig
Nokia Siemens Networks

August 2008

LoST: A Location-to-Service Translation Protocol

Status of This Memo

Abstract

This document describes an XML-based protocol for mapping service identifiers and geodetic or civic location information to service contact URIs. In particular, it can be used to determine the location-appropriate Public Safety Answering Point (PSAP) for emergency services.

Table of Contents

1. Introduction

Protocols such as Naming Authority Pointer (NAPTR) records and the Service Location Protocol (SLP) can be used to discover servers offering a particular service. However, for an important class of services the appropriate specific service instance depends both on the identity of the service and the geographic location of the entity that needs to reach it. Emergency telecommunications services are an important example; here, the service instance is a Public Safety Answering Point (PSAP) that has jurisdiction over the location of the user making the call. The desired PSAP isn't necessarily the one that is topologically or even line-of-sight closest to the caller; rather, it is the one that serves the caller's location based on jurisdictional boundaries.

This document describes a protocol for mapping a service identifier and location information compatible with the Presence Information Data Format Location Object (PIDF-LO) [6] to one or more service URIs. Service identifiers take the form of the service URNs described in [9]. Location information here includes revised civic location information [10] and a subset of the PIDF-LO profile [13], which consequently includes the Geo-Shapes [12] defined for GML [11]. Example service URI schemes include sip [14], xmpp [15], and tel [16]. While the initial focus is on providing mapping functions for emergency services, it is likely that the protocol is applicable to other service URNs. For example, in the United States, the "2-1-1" and "3-1-1" service numbers follow a similar location-to-service behavior as emergency services.

This document names this protocol "LoST", for Location-to-Service Translation. LoST satisfies the requirements [18] for mapping protocols. LoST provides a number of operations, centered around mapping locations and service URNs to service URLs and associated information. LoST mapping queries can contain either civic or geodetic location information. For civic addresses, LoST can indicate which parts of the civic address are known to be valid or invalid, thus providing address validation, as described in Section 3.5 of [18]. LoST indicates errors in the location data to facilitate debugging and proper user feedback, but also provides best-effort answers.

LoST queries can be resolved recursively or iteratively. To minimize round trips and to provide robustness against network failures, LoST supports caching of individual mappings and indicates the region for which the same answer would be returned ("service region").

As defined in this document, LoST messages are carried in HTTP and HTTPS protocol exchanges, facilitating use of TLS for protecting the integrity and confidentiality of requests and responses.

This document focuses on the description of the protocol between the mapping client and the mapping server. Other functions, such as discovery of mapping servers, data replication and the overall mapping server architecture are described in a separate document [19].

The query message carries location information and a service identifier encoded as a Uniform Resource Name (URN) (see [9]) from the LoST client to the LoST server. The LoST server uses its database to map the input values to one or more Uniform Resource Identifiers (URIs) and returns those URIs along with optional information, such as hints about the service boundary, in a response message to the LoST client. If the server cannot resolve the query itself, it may in turn query another server or return the address of another LoST server, identified by a LoST server name. In addition to the mapping function described in Section 8, the protocol also allows to retrieve the service boundary (see Section 9) and to list the services available for a particular location (see Section 11) or supported by a particular server (see Section 10).

2. Terminology and Requirements Notation

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 [1].

This document uses the following terms:

Mapping:
: Mapping is a process that takes a location and a service identifier as inputs and returns one or more URIs. Those URIs can point either to a host providing that service or to a host that in turn routes the request to the final destination. This definition is a generalization of the term "mapping" as used in [18], because LoST can be used for non-emergency services.

LoST client:
: A host acts as a LoST client if it sends LoST query messages and receives LoST response messages.

LoST server:
: A host acts as a LoST server if it receives LoST query messages and sends LoST response messages. In recursive operation, the same entity may be both a client and a server.

Authoritative LoST server:
An authoritative server acts only as a server and successfully resolves the input location and service identifier to a URI or set of URIs.

Service boundary:
A service boundary circumscribes the region within which all locations map to the same service URI or set of URIs for a given service. A service boundary may consist of several non-contiguous geometric shapes.

Validation:
The term "validation" describes the behavior defined as "location validation" in Section 3.5 of [18].

Additional emergency service terminology can be found in [18].

3. Overview of Protocol Usage

The LoST protocol supports the following types of queries and responses:

<findService> and <findServiceResponse>
A LoST client retrieves contact URIs based on location information and a service identifier with this request and response. The same query type may also ask for location validation and for service numbers, either combined with a mapping request or separately. The details can be found in Section 8.

<getServiceBoundary> and <getServiceBoundaryResponse>
A LoST client obtains a service boundary with this request and response, as described in Section 9.

<listServices> and <listServicesResponse>
With this request and response, a LoST client can find out which services a LoST server supports, as described in Section 10.

<listServicesByLocation> and <listServicesByLocationResponse>
A LoST client can determine with this request and response which services are available for a specific location region. Section 11 describes the details.

LoST clients may initiate any of the above queries at any time. Among the common triggers are:

1. when the client initially starts up or attaches to a network;

2. when the client detects that its location has changed sufficiently that it is outside the bounds of the service region;

3. when a SIP message arrives at a SIP proxy performing location-based call routing;

4. when cached mapping information has expired; and

5. when invoking a particular service. At that time, a client may omit requests for service boundaries or other auxiliary information.

A service-specific Best Current Practice (BCP) document, such as [21], governs whether a client is expected to invoke the mapping service just before needing the service or whether to rely on cached answers. Cache entries expire at their expiration time (see Section 5.2), or they become invalid if the caller's device moves beyond the boundaries of the service region. Service-specific Best Current Practice documents may also provide guidance on the contact URI schemes most appropriate to the service. As a general set of guidelines, URI schemes that do not provide mechanisms for actually initiating a contact method should be avoided (examples include data, info, cid, and tag) as transforming those references into contact mechanisms requires a layer of indirection that makes the overall mechanism more fragile. Provisionally registered URI schemes should also be carefully considered before use, because they are subject to change in core semantics.

4. LoST Servers and Their Resolution

LoST servers are identified by U-NAPTR/DDDS (URI-Enabled NAPTR/ Dynamic Delegation Discovery Service) [8] application unique strings, in the form of a DNS name. An example is 'lostserver.example.com'.

Clients need to use the U-NAPTR [8] specification described below to obtain a URI (indicating host and protocol) for the applicable LoST service. In this document, only the HTTP and HTTPS URL schemes are defined. Note that the HTTP URL can be any valid HTTP URL, including those containing path elements.

The following two DNS entries show the U-NAPTR resolution for "example.com" to the HTTPS URL https://lostserv.example.com/secure or the HTTP URL http://lostserver.example.com, with the former being preferred.

```
example.com.

IN NAPTR 100  10   "u"    "LoST:https"
     "!.*!https://lostserver.example.com/secure!"  ""

IN NAPTR 200  10   "u"    "LoST:http"
     "!.*!http://lostserver.example.com!"  ""
```

Clients learn the LoST server's host name by means beyond the scope of this specification, such as SIP configuration and DHCP [25].

5. The <mapping> Element

The <mapping> element is the core data element in LoST, describing a service region and the associated service URLs. Its attributes and elements are described in subsections below.

5.1. The Mapping Data Source: 'source', 'sourceId', and 'lastUpdated' Attributes

The 'source', 'sourceId', and 'lastUpdated' attributes uniquely identify a particular mapping record. They are created by the authoritative source for a mapping and are never modified when a mapping is served from a cache. All three attributes are REQUIRED for all <mapping> elements. A receiver can replace a mapping with another one having the same 'source' and 'sourceId' and a more recent time in 'lastUpdated'.

The 'source' attribute contains a LoST application unique string identifying the authoritative generator of the mapping (Section 4).

The 'sourceId' attribute identifies a particular mapping and contains an opaque token that MUST be unique among all different mappings maintained by the authoritative source for that particular service. For example, a Universally Unique Identifier (UUID) is a suitable format.

The 'lastUpdated' attribute describes when a specific instance of mapping, identified by the combination of 'source' and 'sourceId', was last changed. The contents of this attribute has the XML data type dateTime in its timezoned form, using the canonical UTC representation with the letter 'Z' as the timezone indicator.

5.2. Mapping Validity: The 'expires' Attribute

The 'expires' attribute contains the absolute time at which the mapping becomes invalid. The contents of this attribute is a timezoned XML type dateTime, in canonical representation. The <mapping> element MUST include the 'expires' attribute.

Optionally, this attribute may contain the values of 'NO-CACHE' and 'NO-EXPIRATION' instead of a dateTime value. The value 'NO-CACHE' is an indication that the mapping should not be cached. The value of 'NO-EXPIRATION' is an indication that the mapping does not expire.

On occasion, a server may be forced to return an expired mapping if it cannot reach the authoritative server or the server fails to return a usable answer. Clients and servers MAY cache the mapping so that they have at least some information available. Caching servers that have such stale information SHOULD re-attempt the query each time a client requests a mapping. Since the expired mapping will be returned to the client as a non-error/non-warning response, the client MUST check the 'expires' attribute; if the mapping has expired, local policy at the client determines whether it discards the answer and tries again later or uses the possibly stale response.

5.3. Describing the Service with the <displayName> Element

Zero or more <displayName> elements describe the service with a string that is suitable for display to human users, each annotated with the 'xml:lang' attribute that contains a language tag to aid in the rendering of text.

5.4. The Mapped Service: The <service> Element

The mandatory <service> element identifies the service for which this mapping applies. Two cases need to be distinguished when the LoST server sets the <service> element in the response message:

1. If the requested service, identified by the service URN [9] in the <service> element of the request, exists for the location indicated, then the LoST server copies the service URN from the request into the <service> element.

2. If, however, the requested service, identified by the service URN [9] in the <service> element in the request, does not exist for the location indicated, the server either can return a <serviceNotImplemented> (Section 13.1) error or can provide an alternate service that approximates the desired service for that

location. In the latter case, the server MUST include a <service> element with the alternative service URN. The choice of service URN is left to local policy, but the alternate service should be able to satisfy the original service request.

5.5. Defining the Service Region with the <serviceBoundary> Element

A response MAY indicate the region for which the service URL returned would be the same as in the actual query, the so-called service region. The service region can be indicated by value or by reference (see Section 5.6). If a client moves outside the service area and wishes to obtain current service data, it sends a new query with its current location. The service region is described by value in one or more <serviceBoundary> elements, each formatted according to a specific location profile, identified by the 'profile' attribute (see Section 12). <serviceBoundary> elements formatted according to different location profiles are alternative representations of the same area, not additive to one another; this allows a client understanding only one of the profile types to be sure it has a complete view of the serviceBoundary. Within a serviceBoundary element there may, however, be multiple locations which are additive; this is necessary because some <serviceBoundary> areas could not be easily expressed with a single shape or civic location. If included in a response, the <serviceBoundary> element MUST contain at least one service boundary that uses the same profile as the request.

A service boundary is requested by the client, using the 'serviceBoundary' attribute in the request with the value set to "value".

5.6. Service Boundaries by Reference: The <serviceBoundaryReference> Element

Since geodetic service boundaries may contain thousands of points and can thus be quite large, clients may wish to conserve bandwidth by requesting a reference to the service boundary instead of the value described in Section 5.5. The identifier of the service boundary is returned as an attribute of the <serviceBoundaryReference> element, along with a LoST application unique string (see Section 4) identifying the server from where it can be retrieved. The actual value of the service boundary is then retrieved with the getServiceBoundary (Section 9) request.

A reference to a service boundary is requested by the client using the 'serviceBoundary' attribute in the request with the value set to "reference". A LoST server may decide, based on local policy, to return the service boundary by value or to omit the <serviceBoundaryReference> element in the response.

The identifier is a random token with at least 128 bits of entropy and can be assumed to be globally unique. It uniquely references a particular boundary. If the boundary changes, a new identifier MUST be chosen. Because of these properties, a client receiving a mapping response can simply check if it already has a copy of the boundary with that identifier. If so, it can skip checking with the server whether the boundary has been updated. Since service boundaries are likely to remain unchanged for extended periods of time, possibly exceeding the normal lifetime of the service URL, this approach avoids unnecessarily refreshing the boundary information just because the remainder of the mapping has become invalid.

5.7. The Service Number: The <serviceNumber> Element

The service number is returned in the optional <serviceNumber> element. It contains a string of digits, * and # that a user on a device with a 12-key dial pad could use to reach that particular service.

5.8. Service URLs: The <uri> Element

The response returns the service URLs in one or more <uri> elements. The URLs MUST be absolute URLs. The ordering of the URLs has no particular significance. Each URL scheme MUST only appear at most once, but it is permissible to include both secured and regular versions of a protocol, such as both 'http' and 'https' or 'sip' and 'sips'.

6. Path of a Request: The <path> Element

To prevent loops and to allow tracing of request and response paths, all requests that allow recursion include a <path> element that contains one or more <via> elements, each possessing an attribute containing a LoST application unique string (see Section 4). The order of <via> elements corresponds to the order of LoST servers, i.e., the first <via> element identifies the server that initially received the request from the client issuing the request. Every server in a recursive query operation is included in the <path> element, including the first server to receive it.

The server that answers the request instead of forwarding it, such as the authoritative server, copies the <path> element verbatim into the response. The <path> element is not modified in responses as the responses traverses the server chain back to the querying client.

If a query is answered iteratively, the querier includes all servers that it has already contacted.

When a cached mapping is returned, then the <path> element cached together with the mapping is returned.

The example in Figure 4 indicates that the answer was given to the client by the LoST server at esgw.ueber-110.de.example, which got the answer from the (authoritative) LoST server at polizei.muenchen.de.example.

7. Identifying the Location Element Used for Mapping: <locationUsed>

Several of the requests can provide one or more <location> elements, among which the server gets to choose. It is useful for the client to be able to determine which one was actually used in producing the result. For that purpose, the <location> tag MUST contain an 'id' attribute that uniquely identifies the <location> element. The format of the identifier is left to the client; it could, for example, use a hash of the location information. The server returns the identifier for the <location> element it used in the <locationUsed> tag.

8. Mapping a Location and Service to URLs: <findService>

8.1. Overview

The <findService> query constitutes the core of the LoST functionality, mapping civic or geodetic locations to URLs and associated data. After giving an example, we enumerate the elements of the query and response.

8.2. Examples

8.2.1. Example Using Geodetic Coordinates

The following is an example of mapping a service to a location using geodetic coordinates, for the service associated with the police (urn:service:sos.police).

```
<?xml version="1.0" encoding="UTF-8"?>
<findService
  xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:p2="http://www.opengis.net/gml"
  serviceBoundary="value"
  recursive="true">

  <location id="6020688f1ce1896d" profile="geodetic-2d">
    <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326">
       <p2:pos>37.775 -122.422</p2:pos>
    </p2:Point>
  </location>
  <service>urn:service:sos.police</service>

</findService>
```

Figure 1: A <findService> geodetic query

Given the query above, a server would respond with a service, and information related to that service. In the example below, the server has mapped the location given by the client for a police service to the New York City Police Department, instructing the client that it may contact them via the URIs "sip:nypd@example.com" and "xmpp:nypd@example.com". The server has also given the client a geodetic, two-dimensional boundary for this service. The mapping was last updated on November 1, 2006 and expires on January 1, 2007. If the client's location changes beyond the given service boundary or the expiration time has been reached, it may want to requery for this information, depending on the usage environment of LoST.

```
<?xml version="1.0" encoding="UTF-8"?>
<findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:p2="http://www.opengis.net/gml">
  <mapping
    expires="2007-01-01T01:44:33Z"
    lastUpdated="2006-11-01T01:00:00Z"
    source="authoritative.example"
    sourceId="7e3f40b098c711dbb6060800200c9a66">
    <displayName xml:lang="en">
      New York City Police Department
    </displayName>
    <service>urn:service:sos.police</service>
    <serviceBoundary profile="geodetic-2d">
      <p2:Polygon srsName="urn:ogc:def::crs:EPSG::4326">
        <p2:exterior>
          <p2:LinearRing>
            <p2:pos>37.775 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4194</p2:pos>
          </p2:LinearRing>
        </p2:exterior>
      </p2:Polygon>
    </serviceBoundary>
    <uri>sip:nypd@example.com</uri>
    <uri>xmpp:nypd@example.com</uri>
    <serviceNumber>911</serviceNumber>
  </mapping>
  <path>
    <via source="resolver.example"/>
    <via source="authoritative.example"/>
  </path>
  <locationUsed id="6020688f1ce1896d"/>
</findServiceResponse>
```

Figure 2: A <findServiceResponse> geodetic answer

8.2.2. Civic Address Mapping Example

The example below shows how to map a service to a location much like the example in Section 8.2.1, but using civic address location information. In this example, the client requests the service associated with police (urn:service:sos.police) along with a specific civic address (house number 6 on a street named Otto-Hahn-Ring in Munich, Germany).

```
<?xml version="1.0" encoding="UTF-8"?>
<findService xmlns="urn:ietf:params:xml:ns:lost1"
  recursive="true" serviceBoundary="value">
  <location id="627b8bf819d0bad4d" profile="civic">
    <civicAddress
      xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">
      <country>DE</country>
      <A1>Bavaria</A1>
      <A3>Munich</A3>
      <A6>Otto-Hahn-Ring</A6>
      <HNO>6</HNO>
      <PC>81675</PC>
    </civicAddress>
  </location>
  <service>urn:service:sos.police</service>
</findService>
```

Figure 3: A <findService> civic address query

Given the query above, a server would respond with a service, and information related to that service. In the example below, the server has mapped the location given by the client for a police service to the Muenchen Polizei-Abteilung, instructing the client that it may contact them via the URIs sip:munich-police@example.com and xmpp:munich-police@example.com. The server has also given the client a civic address boundary (the city of Munich) for this service. The mapping was last updated on November 1, 2006 by the authoritative source "polizei.muenchen.de.example" and expires on January 1, 2007. This instructs the client to requery for the information if its location changes beyond the given service boundary (i.e., beyond the indicated district of Munich) or after January 1, 2007.

```
<?xml version="1.0" encoding="UTF-8"?>
<findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1">
  <mapping
    expires="2007-01-01T01:44:33Z"
    lastUpdated="2006-11-01T01:00:00Z"
    source="esgw.ueber-110.de.example"
    sourceId="e8b05a41d8d1415b80f2cdbb96ccf109">
    <displayName xml:lang="de">
      Muenchen Polizei-Abteilung
    </displayName>
    <service>urn:service:sos.police</service>
    <serviceBoundary
      profile="civic">
      <civicAddress
        xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">
        <country>DE</country>
        <A1>Bavaria</A1>
        <A3>Munich</A3>
        <PC>81675</PC>
      </civicAddress>
    </serviceBoundary>
    <uri>sip:munich-police@example.com</uri>
    <uri>xmpp:munich-police@example.com</uri>
    <serviceNumber>110</serviceNumber>
  </mapping>
  <path>
    <via source="esgw.ueber-110.de.example"/>
    <via source="polizei.muenchen.de.example"/>
  </path>
  <locationUsed id="627b8bf819d0bad4d"/>
</findServiceResponse>
```

Figure 4: A <findServiceResponse> civic address answer

8.3. Components of the <findService> Request

The <findService> request includes attributes and elements that govern whether the request is handled iteratively or recursively, whether location validation is performed, and which elements may be contained in the response.

8.3.1. The <location> Element

The <findService> query communicates location information using one or more <location> elements, which MUST conform to a location profile (see Section 12). There MUST NOT be more than one location element

for each distinct location profile. The order of location elements is significant; the server uses the first location element where it understands the location profile.

8.3.2. Identifying the Service: The <service> Element

The type of service desired is specified by the <service> element. It contains service URNs from the registry established in [9].

8.3.3. Recursion and Iteration

LoST can operate in either recursive or iterative mode, on a request-by-request basis. In recursive mode, the LoST server initiates queries on behalf of the requester and returns the result to the requester.

In iterative mode, the server contacted returns a redirection response indicating the next server to be queried if the server contacted cannot provide an answer itself.

For the queries defined in this document, only the LoST <findService> and <listServicesByLocation> queries can be recursive, as indicated by the 'recursive' attribute. A value of "true" indicates a recursive query, with the default being "false" when the attribute is omitted. Regardless of the attribute, a server MAY always answer a query by providing a LoST application unique string (see Section 4), i.e., indirection; however, it MUST NOT recurse if the attribute is "false".

8.3.4. Service Boundary

LoST <mapping> elements can describe the service boundary either by value or by reference. Returning a service boundary reference is generally more space-efficient for geospatial (polygon) boundaries and if the boundaries change rarely, but does incur an additional <getServiceBoundary> request. The querier can express a preference for one or the other modality with the 'serviceBoundary' attribute in the <findService> request, but the server makes the final decision as to whether to return a reference or a value.

8.3.5. Requesting Civic Location Validation

Civic address validation is requested by setting the optional attribute 'validateLocation' to true. If the attribute is omitted, it is assumed to be false. The response is described in Section 8.4.2. The example in Figure 5 demonstrates address validation. If the server chooses a geodetic location among the locations provided in a request, the attribute is ignored.

```
<?xml version="1.0" encoding="UTF-8"?>
<findService
  xmlns="urn:ietf:params:xml:ns:lost1"
  recursive="true"
  validateLocation="true"
  serviceBoundary="value">
  <location id="627b8bf819d0bad4d" profile="civic">
    <civicAddress
      xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">
      <country>DE</country>
      <A1>Bavaria</A1>
      <A3>Munich</A3>
      <A6>Otto-Hahn-Ring</A6>
      <HNO>6</HNO>
      <PC>81675</PC>
    </civicAddress>
  </location>
  <service>urn:service:sos.police</service>
</findService>
```

Figure 5: A <findService> query with address validation request

```
<?xml version="1.0" encoding="UTF-8"?>
<findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1">
  <mapping
    expires="2007-01-01T01:44:33Z"
    lastUpdated="2006-11-01T01:00:00Z"
    source="authoritative.example"
    sourceId="4db898df52b84edfa9b6445ea8a0328e">
    <displayName xml:lang="de">
      Muenchen Polizei-Abteilung
    </displayName>
    <service>urn:service:sos.police</service>
    <serviceBoundary profile="civic">
      <civicAddress
        xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">
        <country>DE</country>
        <A1>Bavaria</A1>
        <A3>Munich</A3>
        <PC>81675</PC>
      </civicAddress>
    </serviceBoundary>
    <uri>sip:munich-police@example.com</uri>
    <uri>xmpp:munich-police@example.com</uri>
    <serviceNumber>110</serviceNumber>
  </mapping>
  <locationValidation>
    <valid>country A1 A3 A6</valid>
    <invalid>PC</invalid>
    <unchecked>HNO</unchecked>
  </locationValidation>
  <path>
    <via source="resolver.example"/>
    <via source="authoritative.example"/>
  </path>
  <locationUsed id="627b8bf819d0bad4d"/>
</findServiceResponse>
```

Figure 6: A <findServiceResponse> message with address validation information

8.4. Components of the Mapping Response <findServiceResponse>

8.4.1. Overview

Mapping responses consist of the <mapping> element (Section 5) describing the mapping itself, possibly followed by warnings (Section 13.2), location validation information (Section 8.4.2), and an indication of the path (Section 6) the response has taken.

8.4.2. Civic Address Validation: The <locationValidation> Element

A server can indicate in its response which civic address elements it has recognized as valid, which ones it has ignored, and which ones it has checked and found to be invalid. The server SHOULD include this information if the 'validateLocation' attribute in the request was true, but local policy at the server may allow this information to be omitted. Each element contains a list of tokens separated by whitespace, enumerating the civic location labels used in child elements of the <civicAddress> element. The <valid> element enumerates those civic address elements that have been recognized as valid by the LoST server and that have been used to determine the mapping. The <unchecked> elements enumerates the civic address elements that the server did not check and that were not used in determining the response. The <invalid> element enumerate civic address elements that the server attempted to check, but that did not match the other civic address elements found in the <valid> list. Civic location tokens that are not listed in either the <valid>, <invalid>, or <unchecked> element belong to the class of unchecked tokens.

Note that the same address can yield different responses if parts of the civic address contradict each other. For example, if the postal code does not match the city, local server policy determines whether the postal code or the city is considered valid. The mapping naturally corresponds to the valid elements.

The example shown in Figure 5 and in Figure 6 indicates that the tokens 'country', 'A1', 'A3', and 'A6' have been validated by the LoST server. The server considered the postal code 81675 in the <PC> element as not valid for this location. The 'HNO' token belongs to the class of unchecked location tokens.

9. Retrieving the Service Boundary via <getServiceBoundary>

As discussed in Section 5.5, the <findServiceResponse> can return a globally unique identifier in the 'serviceBoundary' attribute that can be used to retrieve the service boundary, rather than returning the boundary by value. This is shown in the example in Figure 7 and Figure 8. The client can then retrieve the boundary using the <getServiceBoundary> request and obtains the boundary in the <getServiceBoundaryResponse>, illustrated in the example in Figure 9 and Figure 10. The client issues the request to the server identified in the 'server' attribute of the <serviceBoundaryReference> element. These requests are always directed to the authoritative server and do not recurse.

```
<?xml version="1.0" encoding="UTF-8"?>
<findService
  xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:p2="http://www.opengis.net/gml"
  recursive="true"
  serviceBoundary="reference">
  <location id="6020688f1ce1896d" profile="geodetic-2d">
    <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326">
       <p2:pos>37.775 -122.422</p2:pos>
    </p2:Point>
  </location>
  <service>urn:service:sos.police</service>
</findService>
```

Figure 7: <findService> request and response with service boundary reference

```
<?xml version="1.0" encoding="UTF-8"?>
<findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:p2="http://www.opengis.net/gml">
  <mapping
    expires="2007-01-01T01:44:33Z"
    lastUpdated="2006-11-01T01:00:00Z"
    source="authoritative.example"
    sourceId="7e3f40b098c711dbb6060800200c9a66">
    <displayName xml:lang="en">
      New York City Police Department
    </displayName>
    <service>urn:service:sos.police</service>
    <serviceBoundaryReference
      source="authoritative.example"
      key="7214148E0433AFE2FA2D48003D31172E"/>
    <uri>sip:nypd@example.com</uri>
    <uri>xmpp:nypd@example.com</uri>
    <serviceNumber>911</serviceNumber>
  </mapping>
  <path>
    <via source="resolver.example"/>
    <via source="authoritative.example"/>
  </path>
  <locationUsed id="6020688f1ce1896d"/>
</findServiceResponse>
```

Figure 8: <findServiceResponse> message with service boundary reference

```
<?xml version="1.0" encoding="UTF-8"?>
<getServiceBoundary xmlns="urn:ietf:params:xml:ns:lost1"
    key="7214148E0433AFE2FA2D48003D31172E"/>
```

Figure 9: Requesting a service boundary with <getServiceBoundary>

```
<?xml version="1.0" encoding="UTF-8"?>
<getServiceBoundaryResponse
  xmlns="urn:ietf:params:xml:ns:lost1">
    <serviceBoundary profile="geodetic-2d">
      <p2:Polygon srsName="urn:ogc:def::crs:EPSG::4326">
        <p2:exterior>
          <p2:LinearRing>
            <p2:pos>37.775 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4194</p2:pos>
          </p2:LinearRing>
        </p2:exterior>
      </p2:Polygon>
    </serviceBoundary>
  <path>
    <via source="resolver.example"/>
    <via source="authoritative.example"/>
  </path>
</getServiceBoundaryResponse>
```

Figure 10: Geodetic service boundary response

10. List Services: <listServices>

A LoST client can ask a LoST server for the list of services that it understands, primarily for diagnostic purposes. The query does not contain location information, as it simply provides an indication of which services the server can look up, not whether a particular service is offered for a particular area. Typically, only top-level services are included in the answer, implying support for all sub-services. Since the query is answered by the queried server, there is no notion of recursion or indirection. The <listServicesByLocation> (Section 11) query below can be used to find out whether a particular service is offered for a specific location. An example request and response are shown in Figure 11.

```
<?xml version="1.0" encoding="UTF-8"?>
<listServices
  xmlns="urn:ietf:params:xml:ns:lost1">
  <service>urn:service:sos</service>
</listServices>
```

Figure 11: Example of <ListServices> query

```
<?xml version="1.0" encoding="UTF-8"?>
<listServicesResponse
 xmlns="urn:ietf:params:xml:ns:lost1">
 <serviceList>
  urn:service:sos.ambulance
  urn:service:sos.animal-control
  urn:service:sos.fire
  urn:service:sos.gas
  urn:service:sos.mountain
  urn:service:sos.marine
  urn:service:sos.physician
  urn:service:sos.poison
  urn:service:sos.police
 </serviceList>
 <path>
  <via source="authoritative.example"/>
 </path>
</listServicesResponse>
```

Figure 12: Example of <ListServicesResponse>

11. List Services By Location: <listServicesByLocation>

A LoST client can ask a LoST server for the list of services it knows about for a particular area. The <listServicesByLocation> query contains one or more <location> elements, each from a different location profile (Section 12), and may contain the <service> element. As for <findService>, the server selects the first location element that has a profile the server understands and it can operate either recursively or iteratively; <via> elements track the progress of the request. The query indicates the services that the server can enumerate from within the forest structure of which it is a part. Because LoST does not presume a single, overarching organization of all potential service types, there may be services available within a geographic area that could be described by other LoST servers connected to other forest structures. As an example, the emergency services forest for a region may be distinct from the forests that locate commercial services within the same region.

If the query contains the <service> element, the LoST server returns only immediate child services of the queried service that are available for the provided location. If the <service> element is absent, the LoST service returns all top-level services available for the provided location that it knows about.

A server responds to this query with a <listServicesByLocationResponse> response. This response MAY contain <via> elements (see Section 6) and MUST contain a <serviceList> element, consisting of a whitespace-separated list of service URNs. The query and response are illustrated in Figure 13 and in Figure 14, respectively.

```
<?xml version="1.0" encoding="UTF-8"?>
<listServicesByLocation
  xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:p2="http://www.opengis.net/gml"
  recursive="true">
  <location id="3e19dfb3b9828c3" profile="geodetic-2d">
    <p2:Point srsName="urn:ogc:def:crs:EPSG::4326">
      <p2:pos>-34.407 150.883</p2:pos>
    </p2:Point>
  </location>
  <service>urn:service:sos</service>
</listServicesByLocation>
```

Figure 13: Example of <ListServicesbyLocation> query

```
<?xml version="1.0" encoding="UTF-8"?>
<listServicesByLocationResponse
 xmlns="urn:ietf:params:xml:ns:lost1">
 <serviceList>
  urn:service:sos.ambulance
  urn:service:sos.animal-control
  urn:service:sos.fire
  urn:service:sos.gas
  urn:service:sos.mountain
  urn:service:sos.marine
  urn:service:sos.physician
  urn:service:sos.poison
  urn:service:sos.police
 </serviceList>
 <path>
  <via source="resolver.example"/>
  <via source="authoritative.example"/>
 </path>
 <locationUsed id="3e19dfb3b9828c3"/>
</listServicesByLocationResponse>
```

Figure 14: Example of <ListServicesByLocationResponse> response

12. Location Profiles

LoST uses location information in <location> elements in requests and <serviceBoundary> elements in responses. Such location information may be expressed in a variety of ways. This variety can cause interoperability problems where a request or response contains location information in a format not understood by the server or the client, respectively. To achieve interoperability, this document defines two mandatory-to-implement baseline location profiles to define the manner in which location information is transmitted. It is possible to standardize other profiles in the future. The baseline profiles are:

geodetic-2d:
: a profile for two-dimensional geodetic location information, as described in Section 12.2;.

civic:
: a profile consisting of civic address location information, as described in Section 12.3.

Requests and responses containing <location> or <serviceBoundary> elements MUST contain location information in exactly one of the two baseline profiles, in addition to zero or more additional profiles. The ordering of location information indicates a preference on the part of the sender.

Standards action is required for defining new profiles. A location profile MUST define:

1. The token identifying it in the LoST location profile registry.

2. The formal definition of the XML to be used in requests, i.e., an enumeration and definition of the XML child elements of the <location> element.

3. The formal definition of the XML to be used in responses, i.e., an enumeration and definition of the XML child elements of the <serviceBoundary> element.

4. The declaration of whether geodetic-2d or civic is to be used as the baseline profile. It is necessary to explicitly declare the baseline profile as future profiles may be combinations of geodetic and civic location information.

12.1. Location Profile Usage

A location profile is identified by a token in an IANA-maintained registry (Section 17.5). Clients send location information compliant with a location profile, and servers respond with location information compliant with that same location profile.

When a LoST client sends a <findService> request that provides location information, it includes one or more <location> elements. A <location> element carries an optional 'profile' attribute that indicates the location format of the child elements. A client may obtain location information that does not conform to a profile it recognizes, or it may not have the capability to map XML to profiles. In that case, a client MAY omit the profile attribute and the server should interpret the XML location data to the best of its ability, returning a "locationProfileUnrecognized" error if it is unable to do so.

The concept of location profiles is described in Section 12. With the ability to specify more than one <location> element, the client is able to convey location information for multiple location profiles in the same request.

When a LoST server sends a response that contains location information, it uses the <serviceBoundary> elements much like the client uses the <location> elements. Each <serviceBoundary> element contains location information conforming to the location profile specified in the 'profile' attribute. A response MAY contain multiple mappings or boundaries for the different <location> elements, subject to the restrictions below.

Using the location profiles defined in this document, the following rules ensure interoperability between clients and servers:

1. A client MUST be capable of understanding the response for the baseline profiles it used in the request.

2. If a client sends location information conformant to any location profile other than the ones described in this document, it MUST also send, in the same request, location information conformant to one of the baseline profiles. Otherwise, the server might not be able to understand the request.

3. A client MUST NOT send multiple <location> objects that are derived from different baseline profiles. In other words, a client MUST only send location objects according to the same baseline profile in a query, but it MAY contain a location element following a baseline profile in addition to some other profile.

4. If a client has both location information primarily of geodetic nature and location information primarily of a civic nature, it MUST send separate requests containing each type of location information.

5. There can only be one instance of each location profile in a query.

6. Servers MUST implement all profiles described in this document.

7. A server uses the first-listed location profile that it understands and ignores the others.

8. If a server receives a request that only contains location information using profiles it does not understand, the server responds with a <locationProfileError> (Section 13.1).

9. The <serviceBoundary> element MUST use the same location profile that was used to retrieve the answer and indicates which profile has been used with the 'profile' attribute.

These rules enable the use of location profiles not yet specified, while ensuring baseline interoperability. Take, for example, this scenario illustrated in Figure 15 and 16. Client X has had its firmware upgraded to support the 'not-yet-standardized-prism-profile' location profile. Client X sends location information to Server Y, which does not understand the 'not-yet-standardized-prism-profile' location profile. If Client X also sends location information using the geodetic-2D baseline profile, then Server Y will still be able to understand the request and provide an understandable response, though with location information that might not be as precise or expressive as desired. This is possible because both Client X and Server Y understand the baseline profile.

```
<?xml version="1.0" encoding="UTF-8"?>
<findService
  xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:gs="http://www.opengis.net/pidflo/1.0"
  recursive="true"
  serviceBoundary="value">
  <location id="ABC 123"
      profile="not-yet-standardized-prism-profile">
    <gs:Prism srsName="urn:ogc:def:crs:EPSG::4979">
      <gs:base>
        <gml:Polygon>
          <gml:exterior>
            <gml:LinearRing>
              <gml:posList>
                42.556844 -73.248157 36.6
                42.656844 -73.248157 36.6
                42.656844 -73.348157 36.6
                42.556844 -73.348157 36.6
                42.556844 -73.248157 36.6
              </gml:posList>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
      </gs:base>
      <gs:height uom="urn:ogc:def:uom:EPSG::9001">
        2.4
      </gs:height>
    </gs:Prism>
  </location>
  <location id="DEF 345" profile="geodetic-2d">
    <gml:Point id="point1" srsName="urn:ogc:def:crs:EPSG:4326">
      <gml:pos>42.656844 -73.348157</gml:pos>
    </gml:Point>
  </location>
  <service>urn:service:sos.police</service>
</findService>
```

Figure 15: Example of a <findServices> query with baseline profile interoperability

```
<?xml version="1.0" encoding="UTF-8"?>
<findServiceResponse
  xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:p2="http://www.opengis.net/">
  <mapping
    expires="2007-01-01T01:44:33Z"
    lastUpdated="2006-11-01T01:00:00Z"
    source="authoritative.example"
    sourceId="cf19bbb038fb4ade95852795f045387d">
    <displayName xml:lang="en">
      New York City Police Department
    </displayName>
    <service>urn:service:sos.police</service>
    <serviceBoundary profile="geodetic-2d">
      <p2:Polygon srsName="urn:ogc:def::crs:EPSG::4326">
        <p2:exterior>
          <p2:LinearRing>
            <p2:pos>37.775 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4194</p2:pos>
          </p2:LinearRing>
        </p2:exterior>
      </p2:Polygon>
    </serviceBoundary>
    <uri>sip:nypd@example.com</uri>
    <serviceNumber>911</serviceNumber>
  </mapping>
  <path>
    <via source="resolver.example"/>
    <via source="authoritative.example"/>
  </path>
  <locationUsed id="DEF 345"/>
</findServiceResponse>
```

Figure 16: Example of a <findServiceResponse> message with baseline profile interoperability

12.2. Two-Dimensional Geodetic Profile

The "geodetic-2d" location profile is identified by the token "geodetic-2d". Clients and servers use this profile by placing the following location shapes into the <serviceBoundary> or into the <location> element (unless indicated otherwise):

Point:
The <Point> element is described in Section 5.2.1 of [13]. Section 5.2.1 of [13] shows also the specification of a <Point> with either a two-dimensional position (latitude and longitude) or three-dimensional position (latitude, longitude, and altitude). A client MAY use the three-dimensional position, and servers MAY interpret a three-dimensional position as a two-dimensional position by ignoring the altitude value. A <Point> element is not placed into a <serviceBoundary> element.

Polygon:
The <Polygon> element is described in Section 5.2.2 of [13]. The restriction to 16 points for a polygon contained in Section 7.2.2 of [12] is not applicable to this document.

Circle:
The <Circle> element is described in Section 5.2.3 of [13].

Ellipse:
The <Ellipse> element is described in Section 5.2.4 of [13].

ArcBand:
The <ArcBand> element is described in Section 5.2.5 of [13].

When a client uses a <Polygon>, <Circle>, <Ellipse>, or <ArcBand> element within the <location> element, it is indicating that it will be satisfied by query results appropriate to any portion of the shape. It is left to the server to select an appropriate matching algorithm. A server MAY return multiple <mapping> elements if the shape extends across multiple service areas. Servers are not required to return all possible <mapping> elements to avoid denial-of-service attacks in which clients present queries that span a very large number of service boundaries (e.g., presenting a shape covering all of the United States).

In the case where the server does not return multiple <mapping> elements, but the shape extends across a service boundary, it is possible that the matching algorithm selected by the LoST server will return results that match a portion of the shape but do not match those specific to a particular point. A client may always select a point from within the shape to avoid this condition. The cases where

it does not are generally those where it knows its own position only within the shape given. In emergency service use cases, that may result in the PSAP contacted at the URI provided by LoST being required to forward a call to one of its neighbors; this is an expected part of the overall emergency response system. In non-emergency service use cases, the service deployment model should take into account this issue as part of the provisioning model, as the combination of the data in the LoST server and the algorithm used for mapping determine which contact URIs are returned when shapes are used that overlap multiple service areas.

As a general guideline, any deployed matching algorithm should ensure that the algorithm used does not needlessly return no results if there are valid results for any portion of the shape. If an authoritative server receives a query for which the area in the query overlaps the area for which the server has mapping information, then it MUST return either a mapping whose coverage area intersects the query area or a redirect to another server whose coverage area is a subset of the server's coverage area.

When geodetic location information of this location profile is placed in the <serviceBoundary> element, then the elements with geospatial coordinates are alternative descriptions of the same service region, not additive geometries.

12.3. Basic Civic Profile

The basic civic location profile is identified by the token 'civic'. Clients use this profile by placing a <civicAddress> element, defined in [10], within the <location> element.

Servers use this profile by placing a <civicAddress> element, defined in [10], within the <serviceBoundary> element.

A response MAY contain more than one <serviceBoundary> element with profile 'civic'. Each <serviceBoundary> element describes a set of civic addresses that fall within the service boundary, namely, all addresses that textually match the civic address elements provided, regardless of the value of other address elements. A location falls within the mapping's service boundary if it matches any of the <serviceBoundary> elements. Hence, a response may contain multiple <serviceBoundary> elements with civic and/or geodetic location profiles.

13. Errors, Warnings, and Redirects

When a LoST server cannot fulfill a request completely, it can return either an error or a warning, depending on the severity of the problem. It returns an <errors> element if no useful response can be returned for the query. It returns a <warnings> element as part of another response element if it was able to respond in part, but the response may not be quite what the client had desired. For both elements, the 'source' attribute names the server that originally generated the error or warning, such as the authoritative server. Unless otherwise noted, all elements below can be either an error or a warning, depending on whether a default response, such as a mapping, is included.

13.1. Errors

LoST defines a pattern for errors, defined as <errors> elements in the Relax NG schema. This pattern defines a 'message' attribute containing human-readable text and an 'xml:lang' attribute denoting the language of the human-readable text. One or more such error elements are contained in the <errors> element.

The following errors follow this basic pattern:

badRequest
: The server could not parse or otherwise understand a request, e.g., because the XML was malformed.

forbidden
: The server refused to send an answer. This generally only occurs for recursive queries, namely, if the client tried to contact the authoritative server and was refused.

internalError
: The server could not satisfy a request due to misconfiguration or other operational and non-protocol-related reasons.

locationProfileUnrecognized
: None of the profiles in the request were recognized by the server (see Section 12).

locationInvalid
: The geodetic or civic location in the request was invalid. For example, the longitude or latitude values fall outside the acceptable ranges.

SRSInvalid
The spatial reference system (SRS) contained in the location element was not recognized or does not match the location profile.

loop
During a recursive query, the server was about to visit a server that was already in the server list in the <path> element, indicating a request loop.

notFound
The server could not find an answer to the query.

serverError
An answer was received from another LoST server, but it could not be parsed or otherwise understood. This error occurs only for recursive queries.

serverTimeout
A time out occurred before an answer was received.

serviceNotImplemented
The requested service URN is not implemented and no substitution was available.

An example is below:

```
<?xml version="1.0" encoding="UTF-8"?>
<errors xmlns="urn:ietf:params:xml:ns:lost1"
  source="resolver.example">
   <internalError message="Software bug." xml:lang="en"/>
</errors>
```

Figure 17: Example of an error response

13.2. Warnings

A response MAY contain zero or more warnings. This pattern defines a 'message' attribute containing human-readable text and an 'xml:lang' attribute denoting the language of the human-readable text. One or more such warning elements are contained in the <warnings> element. To provide human-readable text in an appropriate language, the HTTP content negotiation capabilities (see Section 14) MAY be utilized by a server.

This version of the specification defines the following warnings:

locationValidationUnavailable
: The <locationValidationUnavailable> element MAY be returned when a server wishes to notify a client that it cannot fulfill a location validation request. This warning allows a server to return mapping information while signaling this exception state.

serviceSubstitution
: The <serviceSubstitution> element MAY be returned when a server was not able to fulfill a <findService> request for a given service URN. For example, a <findService> request with the 'urn:service:sos.police' service URN for a location in Uruguay may cause the LoST service to return a mapping for the 'urn:service:sos' service URN since Uruguay does not make use of the sub-services police, fire, and ambulance. If this warning is returned, then the <service> element in the response provides information about the service URN that refers to the mapping.

defaultMappingReturned
: The <defaultMappingReturned> element MAY be returned when a server was not able to fulfill a <findService> request for a given location but is able to respond with a default URI. For example, a nearby PSAP may be returned.

An example of a warning is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:p2="http://www.opengis.net/">
  <mapping
    expires="2007-01-01T01:44:33Z"
    lastUpdated="2006-11-01T01:00:00Z"
    source="authoritative.example"
    sourceId="fb8ed888433343b7b27865aeb38f3a99">
    <displayName xml:lang="en">
      New York City Police Department
    </displayName>
    <service>urn:service:sos.police</service>
    <serviceBoundary profile="geodetic-2d">
      <p2:Polygon srsName="urn:ogc:def::crs:EPSG::4326">
        <p2:exterior>
          <p2:LinearRing>
            <p2:pos>37.775 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4194</p2:pos>
            <p2:pos>37.555 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4264</p2:pos>
            <p2:pos>37.775 -122.4194</p2:pos>
          </p2:LinearRing>
        </p2:exterior>
      </p2:Polygon>
    </serviceBoundary>
    <uri>sip:nypd@example.com</uri>
    <serviceNumber>911</serviceNumber>
  </mapping>
  <warnings source="authoritative.example">
    <defaultMappingReturned
        message="Unable to determine PSAP for the given location;
            using default PSAP"
        xml:lang="en"/>
  </warnings>
  <path>
    <via source="resolver.example"/>
    <via source="authoritative.example"/>
  </path>
</findServiceResponse>
```

Figure 18: Example of a warning response

13.3. Redirects

A LoST server can respond indicating that the querier should redirect the query to another server, using the <redirect> element. The element includes a 'target' attribute indicating the LoST application unique string (see Section 4) that the client SHOULD be contacting next, as well as the 'source' attribute indicating the server that generated the redirect response and a 'message' attribute explaining the reason for the redirect response. During a recursive query, a server receiving a <redirect> response can decide whether it wants to follow the redirection or simply return the response to its upstream querier. The "expires" value in the response returned by the server handling the redirected query indicates the earliest time at which a new query might be needed (see Section 5.2). The query for the same tuple of location and service SHOULD NOT be directed to the server that gave redirect prior to that time.

An example is below:

```
<?xml version="1.0" encoding="UTF-8"?>
<redirect xmlns="urn:ietf:params:xml:ns:lost1"
  target="eastpsap.example"
  source="westpsap.example"
  message="We have temporarily failed over." xml:lang="en"/>
```

Figure 19: Example of a redirect response

14. LoST Transport: HTTP

LoST needs an underlying protocol transport mechanism to carry requests and responses. This document defines the use of LoST over HTTP and LoST over HTTP-over-TLS. Client and server developers are reminded that full support of RFC 2616 HTTP facilities is expected. If LoST clients or servers re-implement HTTP, rather than using available servers or client code as a base, careful attention must be paid to full interoperability. Other transport mechanisms are left to future documents. The available transport mechanisms are determined through the use of the LoST U-NAPTR application. In protocols that support content type indication, LoST uses the media type application/lost+xml.

When using HTTP [3] and HTTP-over-TLS [4], LoST requests use the HTTP POST method. The HTTP request MUST use the Cache-Control response directive "no-cache" to disable HTTP-level caching even by caches that have been configured to return stale responses to client requests.

All LoST responses, including those indicating a LoST warning or error, are carried in 2xx responses, typically 200 (OK). Other 2xx responses, in particular 203 (Non-authoritative information), may be returned by HTTP caches that disregard the caching instructions. 3xx, 4xx, and 5xx HTTP response codes indicate that the HTTP request itself failed or was redirected; these responses do not contain any LoST XML elements. The 3xx responses are distinct from the redirects that are described in Section 13.3; the redirect operation in Section 13.3 occur after a LoST server processes the request. Where an HTTP-layer redirect will be general, a LoST server redirect as described in Section 13.3 might be specific to a specific service or be the result of other processing by the LoST server.

The HTTP URL is derived from the LoST server name via U-NAPTR application, as discussed above.

15. Relax NG Schema

This section provides the Relax NG schema used by the LoST protocol in the compact form. The verbose form is included in Appendix A.

```
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
default namespace ns1 = "urn:ietf:params:xml:ns:lost1"

##
##       Location-to-Service Translation (LoST) Protocol

##
##       A LoST XML instance has three request types, each with
##       a corresponding response type: find service, list services,
##       and get service boundary.
##
start =
  findService
  | listServices
  | listServicesByLocation
  | getServiceBoundary
  | findServiceResponse
  | listServicesResponse
  | listServicesByLocationResponse
  | getServiceBoundaryResponse
  | errors
  | redirect

##
##       The queries.
##
div {
```

```
  findService =
    element findService {
      requestLocation,
      commonRequestPattern,
      attribute validateLocation {
        xsd:boolean >> a:defaultValue [ "false" ]
      }?,
      attribute serviceBoundary {
        ("reference" | "value") >> a:defaultValue [ "reference" ]
      }?,
      attribute recursive { xsd:boolean >> a:defaultValue [ "false" ] }?
    }
  listServices = element listServices { commonRequestPattern }
  listServicesByLocation =
    element listServicesByLocation {
      requestLocation,
      commonRequestPattern,
      attribute recursive { xsd:boolean >> a:defaultValue [ "true" ] }?
    }
  getServiceBoundary =
    element getServiceBoundary { serviceBoundaryKey, extensionPoint }
}

##
##       The responses.
##
div {
  findServiceResponse =
    element findServiceResponse {
      mapping+, locationValidation?, commonResponsePattern, locationUsed
    }
  listServicesResponse =
    element listServicesResponse { serviceList, commonResponsePattern }
  listServicesByLocationResponse =
    element listServicesByLocationResponse {
      serviceList, commonResponsePattern, locationUsed
    }
  getServiceBoundaryResponse =
    element getServiceBoundaryResponse {
      serviceBoundary, commonResponsePattern
    }
}

##
##       A pattern common to some of the queries.
##
div {
  commonRequestPattern = service, path?, extensionPoint
```

```
}

##
##       A pattern common to responses.
##
div {
  commonResponsePattern = warnings*, path, extensionPoint
}

##
##       Location in Requests
##
div {
  requestLocation =
    element location {
      attribute id { xsd:token },
      locationInformation
    }+
}

##
##       Location Information
##
div {
  locationInformation =
    extensionPoint+,
    attribute profile { xsd:NMTOKEN }?
}

##
##       Service Boundary
##
div {
  serviceBoundary = element serviceBoundary { locationInformation }+
}

##
##       Service Boundary Reference
##
div {
  serviceBoundaryReference =
    element serviceBoundaryReference {
      source, serviceBoundaryKey, extensionPoint
    }
  serviceBoundaryKey = attribute key { xsd:token }
}

##
```

```
##       Path -
##       Contains a list of via elements -
##       places through which information flowed
##
div {
  path =
    element path {
      element via { source, extensionPoint }+
    }
}

##
##       Location Used
##
div {
  locationUsed =
    element locationUsed {
      attribute id { xsd:token }
    }?
}

##
##       Expires pattern
##
div {
  expires =
    attribute expires { xsd:dateTime | "NO-CACHE" | "NO-EXPIRATION" }
}

##
##       A QName list
##
div {
  qnameList = list { xsd:QName* }
}

##
##       A location-to-service mapping.
##
div {
  mapping =
    element mapping {
      element displayName {
        xsd:string,
        attribute xml:lang { xsd:language }
      }*,
      service,
      (serviceBoundary | serviceBoundaryReference)?,
```

```
      element uri { xsd:anyURI }*,
      element serviceNumber {
        xsd:token { pattern = "[0-9*#]+" }
      }?,
      extensionPoint,
      expires,
      attribute lastUpdated { xsd:dateTime },
      source,
      attribute sourceId { xsd:token },
      message
    }
}

##
##       Location validation
##
div {
  locationValidation =
    element locationValidation {
      element valid { qnameList }?,
      element invalid { qnameList }?,
      element unchecked { qnameList }?,
      extensionPoint
    }
}

##
##       Errors and Warnings Container.
##
div {
  exceptionContainer =
    (badRequest?
     & internalError?
     & serviceSubstitution?
     & defaultMappingReturned?
     & forbidden?
     & notFound?
     & loop?
     & serviceNotImplemented?
     & serverTimeout?
     & serverError?
     & locationInvalid?
     & locationProfileUnrecognized?),
    extensionPoint,
    source
  errors = element errors { exceptionContainer }
  warnings = element warnings { exceptionContainer }
}
```

```
##
##       Basic Exceptions
##
div {

  ##
  ##         Exception pattern.
  ##
  basicException = message, extensionPoint
  badRequest = element badRequest { basicException }
  internalError = element internalError { basicException }
  serviceSubstitution = element serviceSubstitution { basicException }
  defaultMappingReturned =
    element defaultMappingReturned { basicException }
  forbidden = element forbidden { basicException }
  notFound = element notFound { basicException }
  loop = element loop { basicException }
  serviceNotImplemented =
    element serviceNotImplemented { basicException }
  serverTimeout = element serverTimeout { basicException }
  serverError = element serverError { basicException }
  locationInvalid = element locationInvalid { basicException }
  locationValidationUnavailable =
    element locationValidationUnavailable { basicException }
  locationProfileUnrecognized =
    element locationProfileUnrecognized {
      attribute unsupportedProfiles { xsd:NMTOKENS },
      basicException
    }
}

##
##       Redirect.
##
div {

  ##
  ##         Redirect pattern
  ##
  redirect =
    element redirect {
      attribute target { appUniqueString },
      source,
      message,
      extensionPoint
    }
}
```

```
##
##       Some common patterns.
##
div {
  message =
    (attribute message { xsd:token },
     attribute xml:lang { xsd:language })?
  service = element service { xsd:anyURI }?
  appUniqueString =
    xsd:token { pattern = "([a-zA-Z0-9\-]+\.)+[a-zA-Z0-9]+" }
  source = attribute source { appUniqueString }
  serviceList =
    element serviceList {
      list { xsd:anyURI* }
    }
}

##
##       Patterns for inclusion of elements from schemas in
##       other namespaces.
##
div {

  ##
  ##         Any element not in the LoST namespace.
  ##
  notLost = element * - (ns1:* | ns1:*) { anyElement }

  ##
  ##         A wildcard pattern for including any element
  ##         from any other namespace.
  ##
  anyElement =
    (element * { anyElement }
     | attribute * { text }
     | text)*

  ##
  ##         A point where future extensions
  ##         (elements from other namespaces)
  ##         can be added.
  ##
  extensionPoint = notLost*
}
```

Figure 20: RelaxNG schema

16. Internationalization Considerations

The LoST protocol is mostly meant for machine-to-machine communications; as such, most of its elements are tokens not meant for direct human consumption. If these tokens are presented to the end user, some localization may need to occur. The content of the <displayName> element and the 'message' attributes may be displayed to the end user, and they are thus complex types designed for this purpose.

LoST exchanges information using XML. All XML processors are required to understand UTF-8 and UTF-16 encodings, and therefore all LoST clients and servers MUST understand UTF-8 and UTF-16 encoded XML. Additionally, LoST servers and clients MUST NOT encode XML with encodings other than UTF-8 or UTF-16.

17. IANA Considerations

17.1. U-NAPTR Registrations

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

Application Service Tag: LoST

Defining Publication: The specification contained within this document.

This document registers the following U-NAPTR application protocol tags:

- Application Protocol Tag: http

  Defining Publication: RFC 2616 [3]

- Application Protocol Tag: https

  Defining Publication: RFC 2818 [4]

17.2. Content-Type Registration for 'application/lost+xml'

This specification requests the registration of a new MIME type according to the procedures of RFC 4288 [7] and guidelines in RFC 3023 [5].

MIME media type name: application

MIME subtype name: lost+xml

Mandatory parameters: none

Optional parameters: charset
Indicates the character encoding of enclosed XML.

Encoding considerations: Uses XML, which can employ 8-bit characters, depending on the character encoding used. See RFC 3023 [5], Section 3.2.

Security considerations: This content type is designed to carry LoST protocol payloads.

Interoperability considerations: None

Published specification: RFC 5222

Applications that use this media type: Emergency and location-based systems

Additional information:

Magic Number: None

File Extension: .lostxml

Macintosh file type code: 'TEXT'

Personal and email address for further information:
Hannes Tschofenig, Hannes.Tschofenig@nsn.com

Intended usage: LIMITED USE

Author:
This specification is a work item of the IETF ECRIT working group, with mailing list address <ecrit@ietf.org>.

Change controller:
The IESG <iesg@ietf.org>

17.3. LoST Relax NG Schema Registration

URI: urn:ietf:params:xml:schema:lost1

Registrant Contact: IETF ECRIT Working Group, Hannes Tschofenig (Hannes.Tschofenig@nsn.com).

Relax NG Schema: The Relax NG schema to be registered is contained in Section 15. Its first line is

default namespace = "urn:ietf:params:xml:ns:lost1"

and its last line is

}

17.4. LoST Namespace Registration

URI: urn:ietf:params:xml:ns:lost1

Registrant Contact: IETF ECRIT Working Group, Hannes Tschofenig (Hannes.Tschofenig@nsn.com).

XML:

```
BEGIN
<?xml version="1.0"?>
<!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:lost1</h2>
<p>See <a href="http://www.rfc-editor.org/rfc/rfc5222.txt">
   RFC5222</a>.</p>
</body>
</html>
END
```

17.5. LoST Location Profile Registry

This document creates a registry of location profile names for the LoST protocol. Profile names are XML tokens. This registry will operate in accordance with RFC 5226 [2], Standards Action.

geodetic-2d:
Defined in Section 12.2.

civic:
Defined in Section 12.3.

18. Security Considerations

There are several threats to the overall system of which service mapping forms a part. An attacker that can obtain service contact URIs can use those URIs to attempt to disrupt those services. An attacker that can prevent the lookup of contact URIs can impair the reachability of such services. An attacker that can eavesdrop on the communication requesting this lookup can surmise the existence of an emergency and possibly its nature, and may be able to use this to launch a physical attack on the caller.

To avoid an attacker modifying the query or its result, Transport Layer Security (TLS) MUST be implemented and SHOULD be used. Use is RECOMMENDED both for clients' queries to servers and for queries among servers; this latter recommendation is to help avoid LoST cache poisoning attacks by replacing answers given to caching LoST servers.

The use of server identity checks with TLS, as described in Section 3.1 of [4], is also RECOMMENDED. Omitting the server identity check allows an attacker to masquerade as a LoST server, so this approach should be used only when getting any answer, even from a potentially malicious LoST server, is preferred over closing the connection (and thus not getting any answer at all). The host name compared against the server certificate is the host name in the URI, not the DNS name used as input to NAPTR resolution.

Note that the security considerations in [22] recommend comparing the input of NAPTR resolution to the certificate, not the output (host name in the URI). This approach was not chosen because in emergency service use cases, it is likely that deployments will see a large number of inputs to the U-NAPTR algorithm resolving to a single server, typically run by a local emergency services authority. In this case, checking the input to the NAPTR resolution against the certificates provided by the LoST server would be impractical, as the list of organizations using it would be large, subject to rapid change, and unknown to the LoST server operator.

The use of server identity does leave open the possibility of DNS-based attacks, as the NAPTR records may be altered by an attacker. The attacks include, for example, interception of DNS packets between the client and the recursive name server, DNS cache poisoning, and intentional modifications by the recursive name server; see [23] for more comprehensive discussion.

DNS Security (DNSSEC) [20] can be used to protect against these threats. While DNSSEC is incompletely deployed, users should be aware of the risk, particularly when they are requesting NAPTR records in environments where the local recursive name server, or the network between the client and the local recursive name server, is not considered trustworthy.

LoST deployments that are unable to use DNSSEC and unwilling to trust DNS resolution without DNSSEC cannot use the NATPR-based discovery of LoST servers as is. When suitable configuration mechanisms are available, one possibility is to configure the LoST server URIs (instead of the domain name to be used for NAPTR resolution) directly. Future specifications for applying LoST in non-emergency services may also specify additional discovery mechanisms and name matching semantics.

Generally, LoST servers will not need to authenticate or authorize clients presenting mapping queries. If they do, an authentication of the underlying transport mechanism, such as HTTP basic and digest authentication, MAY be used. Basic authentication SHOULD only be used in combination with TLS.

A more detailed description of threats and security requirements is provided in [17]. The threats and security requirements in non-emergency service uses of LoST may be considerably different from those described here. For example, an attacker might seek monetary benefit by returning service mapping information that directed users to specific service providers. Before deploying LoST in new contexts, a thorough analysis of the threats and requirements specific to that context should be undertaken and decisions made on the appropriate mitigations.

19. Acknowledgments

We would like to the thank the following working group members for the detailed review of previous LoST document versions:

- Martin Thomson (Review July 2006)
- Jonathan Rosenberg (Review July 2006)

- Leslie Daigle (Review September 2006)
- Shida Schubert (Review November 2006)
- Martin Thomson (Review December 2006)
- Barbara Stark (Review January 2007)
- Patrik Faltstrom (Review January 2007)
- Shida Schubert (Review January 2007 as a designated expert reviewer)
- Jonathan Rosenberg (Review February 2007)
- Tom Taylor (Review February 2007)
- Theresa Reese (Review February 2007)
- Shida Schubert (Review February 2007)
- James Winterbottom (Review July 2007)
- Karl Heinz Wolf (Review May and June 2007)

We would also like to thank the following working group members for their input to selected design aspects of the LoST protocol:

- Leslie Daigle and Martin Thomson (DNS-based LoST discovery procedure)
- John Schnizlein (authoritive LoST answers)
- Rohan Mahy (display names)
- James Polk (error handling)
- Ron Watro and Richard Barnes (expiry of cached data)
- Stephen Edge, Keith Drage, Tom Taylor, Martin Thomson, and James Winterbottom (indication of PSAP confidence level)
- Martin Thomson (service boundary references)
- Martin Thomson (service URN in LoST response message)
- Clive D.W. Feather, Martin Thomson (validation functionality)

- Roger Marshall (PSAP preference in LoST response)

- James Winterbottom, Marc Linsner, Keith Drage, Tom Taylor, Martin Thomson, John Schnizlein, Shida Schubert, Clive D.W. Feather, Richard Stastny, John Hearty, Roger Marshall, Jean-Francois Mule, Pierre Desjardins (location profiles)

- Michael Hammer, Patrik Faltstrom, Richard Stastny, Martin Thomson, Roger Marshall, Tom Taylor, Spencer Dawkins, Keith Drage (list services functionality)

- Martin Thomson, Michael Hammer (mapping of services)

- Shida Schubert, James Winterbottom, Keith Drage (default service URN)

- Otmar Lendl (LoST aggregation)

- Tom Taylor (terminology)

Klaus Darilion and Marc Linsner provided miscellaneous input to the design of the protocol. Finally, we would like to thank Brian Rosen, who participated in almost every discussion thread.

Early implementation efforts led to good feedback by two open source implementation groups. We would like to thank the implementers for their work and for helping us to improve the quality of the specification:

- Wonsang Song

- Jong-Yul Kim

- Anna Makarowska

- Krzysztof Rzecki

- Blaszczyk Piotr

We would like to thank Jon Peterson, Dan Romascanu, Lisa Dusseault, and Tim Polk for their IESG review comments. Blocking IESG comments were also received from Pasi Eronen (succeeding Sam Hartman's review) and Cullen Jennings. Adjustments have been made to several pieces of text to satisfy these requests for changes, most notably in the Security Considerations and in the discussion of redirection in the presence of overlapping coverage areas.

20. References

20.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

[4] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[5] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[6] Peterson, J., "A Presence-based GEOPRIV Location Object Format", RFC 4119, December 2005.

[7] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[8] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.

[9] Schulzrinne, H., "A Uniform Resource Name (URN) for Emergency and Other Well-Known Services", RFC 5031, January 2008.

[10] Thomson, M. and J. Winterbottom, "Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO)", RFC 5139, February 2008.

[11] Cox, S., Daisey, P., Lake, R., Portele, C., and A. Whiteside, "Geographic information - Geography Markup Language (GML)", OGC Standard OpenGIS 03-105r1, April 2004.

[12] Reed, C. and M. Thomson, "GML 3.1.1 PIDF-LO Shape Application Schema for use by the Internet Engineering Task Force (IETF)", Candidate OpenGIS Implementation Specification , December 2006.

20.2. Informative References

[13] Winterbottom, J., Thomson, M., and H. Tschofenig, "GEOPRIV PIDF-LO Usage Clarification, Considerations and Recommendations", Work in Progress, February 2008.

[14] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[15] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 3921, October 2004.

[16] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004.

[17] Taylor, T., Tschofenig, H., Schulzrinne, H., and M. Shanmugam, "Security Threats and Requirements for Emergency Call Marking and Mapping", RFC 5069, January 2008.

[18] Schulzrinne, H. and R. Marshall, "Requirements for Emergency Context Resolution with Internet Technologies", RFC 5012, January 2008.

[19] Schulzrinne, H., "Location-to-URL Mapping Architecture and Framework", Work in Progress, September 2007.

[20] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[21] Rosen, B. and J. Polk, "Best Current Practice for Communications Services in support of Emergency Calling", Work in Progress, February 2008.

[22] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.

[23] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, August 2004.

[24] <http://www.tschofenig.priv.at/svn/draft-ietf-ecrit-lost/RelaxNG>.

[25] Schulzrinne, H., Polk, J., and H. Tschofenig, "Discovering Location-to-Service Translation (LoST) Servers Using the Dynamic Host Configuration Protocol (DHCP)", RFC 5223, August 2008.

Appendix A. Non-Normative RELAX NG Schema in XML Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar ns="urn:ietf:params:xml:ns:lost1"
        xmlns="http://relaxng.org/ns/structure/1.0"
        xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
        datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

        <start>
    <a:documentation>
      Location-to-Service Translation (LoST) Protocol

      A LoST XML instance has three request types, each with
      a corresponding response type: find service, list services,
      and get service boundary.
    </a:documentation>
    <choice>
      <ref name="findService"/>
      <ref name="listServices"/>
      <ref name="listServicesByLocation"/>
      <ref name="getServiceBoundary"/>
      <ref name="findServiceResponse"/>
      <ref name="listServicesResponse"/>
      <ref name="listServicesByLocationResponse"/>
      <ref name="getServiceBoundaryResponse"/>
      <ref name="errors"/>
      <ref name="redirect"/>
    </choice>
        </start>

  <div>
    <a:documentation>
      The queries.
    </a:documentation>

    <define name="findService">
      <element name="findService">
        <ref name="requestLocation"/>
        <ref name="commonRequestPattern"/>
        <optional>
          <attribute name="validateLocation">
            <data type="boolean"/>
            <a:defaultValue>false</a:defaultValue>
          </attribute>
        </optional>
        <optional>
          <attribute name="serviceBoundary">
```

```
              <choice>
                <value>reference</value>
                <value>value</value>
              </choice>
              <a:defaultValue>reference</a:defaultValue>
            </attribute>
          </optional>
          <optional>
            <attribute name="recursive">
              <data type="boolean"/>
                <a:defaultValue>false</a:defaultValue>
            </attribute>
          </optional>
        </element>
      </define>

      <define name="listServices">
        <element name="listServices">
          <ref name="commonRequestPattern"/>
        </element>
      </define>

      <define name="listServicesByLocation">
        <element name="listServicesByLocation">
          <ref name="requestLocation"/>
          <ref name="commonRequestPattern"/>
          <optional>
            <attribute name="recursive">
              <data type="boolean"/>
              <a:defaultValue>true</a:defaultValue>
            </attribute>
          </optional>
        </element>
      </define>

      <define name="getServiceBoundary">
        <element name="getServiceBoundary">
          <ref name="serviceBoundaryKey"/>
          <ref name="extensionPoint"/>
        </element>
      </define>
    </div>

    <div>
      <a:documentation>
        The responses.
      </a:documentation>
```

```
      <define name="findServiceResponse">
        <element name="findServiceResponse">
          <oneOrMore>
            <ref name="mapping"/>
          </oneOrMore>
          <optional>
            <ref name="locationValidation"/>
          </optional>
          <ref name="commonResponsePattern"/>
          <ref name="locationUsed"/>
        </element>
      </define>

      <define name="listServicesResponse">
        <element name="listServicesResponse">
          <ref name="serviceList"/>
          <ref name="commonResponsePattern"/>
        </element>
      </define>

      <define name="listServicesByLocationResponse">
        <element name="listServicesByLocationResponse">
          <ref name="serviceList"/>
          <ref name="commonResponsePattern"/>
          <ref name="locationUsed"/>
        </element>
      </define>

      <define name="getServiceBoundaryResponse">
        <element name="getServiceBoundaryResponse">
          <ref name="serviceBoundary"/>
          <ref name="commonResponsePattern"/>
        </element>
      </define>
    </div>

    <div>
      <a:documentation>
        A pattern common to some of the queries.
      </a:documentation>

      <define name="commonRequestPattern">
        <ref name="service"/>
        <optional>
          <ref name="path"/>
```

```
        </optional>
        <ref name="extensionPoint"/>
      </define>
    </div>

    <div>
      <a:documentation>
        A pattern common to responses.
      </a:documentation>

      <define name="commonResponsePattern">
        <zeroOrMore>
          <ref name="warnings"/>
        </zeroOrMore>
        <ref name="path"/>
        <ref name="extensionPoint"/>
      </define>
    </div>

    <div>
      <a:documentation>
        Location in Requests
      </a:documentation>

      <define name="requestLocation">
        <oneOrMore>
          <element name="location">
            <attribute name="id">
              <data type="token"/>
            </attribute>
            <ref name="locationInformation"/>
          </element>
        </oneOrMore>
      </define>
    </div>

    <div>
      <a:documentation>
        Location Information
      </a:documentation>

      <define name="locationInformation">
        <oneOrMore>
          <ref name="extensionPoint"/>
        </oneOrMore>
        <optional>
          <attribute name="profile">
            <data type="NMTOKEN"/>
```

```
        </attribute>
      </optional>
    </define>
  </div>

  <div>
    <a:documentation>
      Service Boundary
    </a:documentation>

    <define name="serviceBoundary">
      <oneOrMore>
        <element name="serviceBoundary">
          <ref name="locationInformation"/>
        </element>
      </oneOrMore>
    </define>
  </div>

  <div>
    <a:documentation>
      Service Boundary Reference
    </a:documentation>

    <define name="serviceBoundaryReference">

      <element name="serviceBoundaryReference">
        <ref name="source"/>
        <ref name="serviceBoundaryKey"/>
        <ref name="extensionPoint"/>
      </element>
    </define>

    <define name="serviceBoundaryKey">
      <attribute name="key">
        <data type="token"/>
      </attribute>
    </define>
  </div>

  <div>
    <a:documentation>
      Path -
      Contains a list of via elements -
      places through which information flowed
    </a:documentation>
```

```
      <define name="path">
        <element name="path">
          <oneOrMore>
            <element name="via">
              <ref name="source"/>
              <ref name="extensionPoint"/>
            </element>
          </oneOrMore>
        </element>
      </define>
    </div>

    <div>
      <a:documentation>
        Location Used
      </a:documentation>

      <define name="locationUsed">
        <optional>
          <element name="locationUsed">
            <attribute name="id">
              <data type="token"/>
            </attribute>
          </element>
        </optional>
      </define>
    </div>

    <div>
      <a:documentation>
        Expires pattern
      </a:documentation>

      <define name="expires">
        <attribute name="expires">
          <choice>
            <data type="dateTime"/>
            <value>NO-CACHE</value>
            <value>NO-EXPIRATION</value>
          </choice>
        </attribute>
      </define>
    </div>

    <div>
      <a:documentation>
        A QName list
      </a:documentation>
```

```
      <define name="qnameList">
        <list>
          <zeroOrMore>
            <data type="QName"/>
          </zeroOrMore>
        </list>
      </define>
    </div>

    <div>
      <a:documentation>
        A location-to-service mapping.
      </a:documentation>

      <define name="mapping">
        <element name="mapping">
          <zeroOrMore>
            <element name="displayName">
              <data type="string"/>
              <attribute name="xml:lang">
                <data type="language"/>
              </attribute>
            </element>
          </zeroOrMore>
          <ref name="service"/>
          <optional>
            <choice>
              <ref name="serviceBoundary"/>
              <ref name="serviceBoundaryReference"/>
            </choice>
          </optional>
          <zeroOrMore>
            <element name="uri">
              <data type="anyURI"/>
            </element>
          </zeroOrMore>
          <optional>
            <element name="serviceNumber">
              <data type="token">
                <param name="pattern">[0-9*#]+</param>
              </data>
            </element>
          </optional>
          <ref name="extensionPoint"/>
          <ref name="expires"/>
          <attribute name="lastUpdated">
            <data type="dateTime"/>
          </attribute>
```

```
          <ref name="source"/>
          <attribute name="sourceId">
            <data type="token"/>
          </attribute>
          <ref name="message"/>
        </element>
      </define>
    </div>

    <div>
      <a:documentation>
        Location validation
      </a:documentation>

      <define name="locationValidation">
        <element name="locationValidation">
          <optional>
            <element name="valid">
              <ref name="qnameList"/>
            </element>
          </optional>
          <optional>
            <element name="invalid">
              <ref name="qnameList"/>
            </element>
          </optional>
          <optional>
            <element name="unchecked">
              <ref name="qnameList"/>
            </element>
          </optional>
          <ref name="extensionPoint"/>
        </element>
      </define>
    </div>

    <div>
      <a:documentation>
        Errors and Warnings Container.
      </a:documentation>

      <define name="exceptionContainer">
        <interleave>
          <optional>
            <ref name="badRequest"/>
          </optional>
          <optional>
```

```
            <ref name="internalError"/>
          </optional>
          <optional>
            <ref name="serviceSubstitution"/>
          </optional>
          <optional>
            <ref name="defaultMappingReturned"/>
          </optional>
          <optional>
            <ref name="forbidden"/>
          </optional>
          <optional>
            <ref name="notFound"/>
          </optional>
          <optional>
            <ref name="loop"/>
          </optional>
          <optional>
            <ref name="serviceNotImplemented"/>
          </optional>
          <optional>
            <ref name="serverTimeout"/>
          </optional>
          <optional>
            <ref name="serverError"/>
          </optional>
          <optional>
            <ref name="locationInvalid"/>
          </optional>
          <optional>
            <ref name="locationProfileUnrecognized"/>
          </optional>
        </interleave>
        <ref name="extensionPoint"/>
        <ref name="source"/>
      </define>

      <define name="errors">
        <element name="errors">
          <ref name="exceptionContainer"/>
        </element>
      </define>

      <define name="warnings">
        <element name="warnings">
          <ref name="exceptionContainer"/>
        </element>
      </define>
```

```
    </div>

    <div>
      <a:documentation>
        Basic Exceptions
      </a:documentation>

      <define name="basicException">
        <a:documentation>
          Exception pattern.
        </a:documentation>
        <ref name="message"/>
        <ref name="extensionPoint"/>
      </define>

      <define name="badRequest">
        <element name="badRequest">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="internalError">
        <element name="internalError">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="serviceSubstitution">
        <element name="serviceSubstitution">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="defaultMappingReturned">
        <element name="defaultMappingReturned">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="forbidden">
        <element name="forbidden">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="notFound">
        <element name="notFound">
          <ref name="basicException"/>
```

```
        </element>
      </define>

      <define name="loop">
        <element name="loop">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="serviceNotImplemented">
        <element name="serviceNotImplemented">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="serverTimeout">
        <element name="serverTimeout">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="serverError">
        <element name="serverError">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="locationInvalid">
        <element name="locationInvalid">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="locationValidationUnavailable">
        <element name="locationValidationUnavailable">
          <ref name="basicException"/>
        </element>
      </define>

      <define name="locationProfileUnrecognized">
        <element name="locationProfileUnrecognized">
          <attribute name="unsupportedProfiles">
            <data type="NMTOKENS"/>
          </attribute>
          <ref name="basicException"/>
        </element>
      </define>
    </div>
```

```
    <div>
      <a:documentation>
        Redirect.
      </a:documentation>

      <define name="redirect">
        <a:documentation>
          Redirect pattern
        </a:documentation>
        <element name="redirect">
          <attribute name="target">
            <ref name="appUniqueString"/>
          </attribute>
          <ref name="source"/>
          <ref name="message"/>
          <ref name="extensionPoint"/>
        </element>
      </define>
    </div>

    <div>
      <a:documentation>
        Some common patterns.
      </a:documentation>

      <define name="message">
        <optional>
          <group>
            <attribute name="message">
              <data type="token"/>
            </attribute>
            <attribute name="xml:lang">
              <data type="language"/>
            </attribute>
          </group>
        </optional>
      </define>

      <define name="service">
        <optional>
          <element name="service">
            <data type="anyURI"/>
          </element>
        </optional>
      </define>
```

```
      <define name="appUniqueString">
        <data type="token">
          <param name="pattern">([a-zA-Z0-9\-]+\.)+[a-zA-Z0-9]+</param>
        </data>
      </define>

      <define name="source">
        <attribute name="source">
          <ref name="appUniqueString"/>
        </attribute>
      </define>

      <define name="serviceList">
        <element name="serviceList">
          <list>
            <zeroOrMore>
              <data type="anyURI"/>
            </zeroOrMore>
          </list>
        </element>
      </define>
    </div>

    <div>
      <a:documentation>
        Patterns for inclusion of elements from schemas in
        other namespaces.
      </a:documentation>

      <define name="notLost">
        <a:documentation>
          Any element not in the LoST namespace.
        </a:documentation>
        <element>
          <anyName>
            <except>
              <nsName ns="urn:ietf:params:xml:ns:lost1"/>
              <nsName/>
            </except>
          </anyName>
          <ref name="anyElement"/>
        </element>
      </define>

      <define name="anyElement">
        <a:documentation>
```

```
          A wildcard pattern for including any element
          from any other namespace.
        </a:documentation>
        <zeroOrMore>
          <choice>
            <element>
              <anyName/>
              <ref name="anyElement"/>
            </element>
            <attribute>
              <anyName/>
            </attribute>
            <text/>
          </choice>
        </zeroOrMore>
      </define>

      <define name="extensionPoint">
        <a:documentation>
          A point where future extensions
          (elements from other namespaces)
          can be added.
        </a:documentation>
        <zeroOrMore>
          <ref name="notLost"/>
        </zeroOrMore>
      </define>
    </div>

  </grammar>
```

Figure 21

Appendix B. Examples Online

The XML examples and Relax NG schemas may be found online [24].

Authors' Addresses

Ted Hardie
Qualcomm, Inc.

EMail: hardie@qualcomm.com

Andrew Newton
American Registry for Internet Numbers
3635 Concorde Parkway, Suite 200
Chantilly, VA 20151
US

Phone: +1 703 227 9894
EMail: andy@hxr.us

Henning Schulzrinne
Columbia University
Department of Computer Science
450 Computer Science Building
New York, NY 10027
US

Phone: +1 212 939 7004
EMail: hgs+ecrit@cs.columbia.edu
URI: http://www.cs.columbia.edu

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
EMail: Hannes.Tschofenig@nsn.com
URI: http://www.tschofenig.priv.at

Full Copyright Statement

Intellectual Property