GEOPRIV	B. Rosen	
Internet-Draft	Neustar	
Intended status: Standards Track	H. Schulzrinne	
Expires: September 2, 2010	Columbia U.	
	March 01, 2010	

TOC

Completing the Request Location
draft-rosen-ecrit-completed-location-00

Abstract

This document defines an extension to LoST (RFC5222) that allows a request to specify that a PIDF be returned in the response if the location was valid, but there were some missing elements. For example, an civic location that did not include a postal code, but was otherwise a complete, and unambiguous location, would receive a complete PIDF in the response that included the postal code.

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”
The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.
This Internet-Draft will expire on September 2, 2010.

Copyright Notice

Table of Contents

---

1. Introduction

TOC

This document describes an update to the LoST protocol [RFC5222] (Hardie, T., Newton, A., Schulzrinne, H., and H. Tschofenig, “LoST: A Location-to-Service Translation Protocol,” August 2008.) which allows a <findservice> request to complete the location in the response. The PIDF returned would have all of the civic address elements of the PIDF in the request, but would contain additional elements not in the request PIDF that the LoST server knew were part of the location.

When performing a validation function, the criteria to determine if a location is valid used will often be that the information provided was sufficient to uniquely identify the location proffered in the request. It may not be necessary that the request PIDF-LO contain all of the elements that the LoST server may know about for the location. So long as there were sufficient elements to uniquely identify the location, the location could be considered valid. However, when the location is subsequently used (in a LoST request, or for any other use), having those fields that the LoST server has, but the requestor does not, may be valuable.

For example, the input PIDF-LO may contain the <PCN> element (the postal community name) and a <PC> element (the postal code), but may be missing the <A3> element (city, township). Having the postal addressing information may be sufficient (along with the other information in the PIDF-LO) to determine exactly where the PIDF-LO refers to. However, the post office name may be different from the actual municipality, and the PSAP that serves it may depend on the actual municipality. To cite a specific, the Postal Code 16046, with the postal community name "Mars", in the state of Pennsylvania in the United States spans a county boundary and four communities. A street name of "Conrad" is in Allegheny County, in Pine Township. There is a Mars township, but it is in Butler

County. Allegheny County is served by the Allegheny PSAP, Butler County is served by the Butler PSAP. If the PIDF-LO in the request included:

```
<country>US</country>
<A1>PA</A1>
<PC>16046</PC>
<PCN>Mars</PCN>
<RD>Conrad</RD>
<STS>DR</STS>
<HNO>470</HNO>
```

the information in this PIDF may be enough to precisely locate the target in Pine Township, Allegheny County (because there is only one Conrad Drive in postal code 16046). The requestor may need to know that this is Pine, and not Mars. The response to this request arising from the use of the extension described in this document may include:

```
<country>US</country>
<A1>PA</A1>
<A2>Allegheny</A2>
<A3>Pine</A3>
<PC>16046</PC>
<PCN>Mars</PCN>
<RD>Conrad</RD>
<STS>DR</STS>
<HNO>470</HNO>
```

This extension allows the requester to ask the LoST server to complete the location, that is, return a complete PIDF with the elements it may have that were not in the request, provided that the location was valid. If the location was not valid, the LoST server would not know what to return.
This feature is not a 'partial location matching function' which might be used to guess at what a valid location might be given a less-than-unique set of elements. When the LoST server returns additional elements, it does so knowing that the elements it returns indeed belong to the location described by the request.

---

2. Conventions used in this document

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

TOC

3. <requestCompletedLocation> element

This document defines a new attribute to <findService> called 'requestLocation'. When the LoST server receives a <findService> request containing a 'requestLocation' attribute with values of 'true', and the PIDF in the <location> element within the request is valid, it will include a PIDF in the response.

---

4. <responseCompletedLocation> element

TOC

When the <findService> request contains a requestCompletedLocation attribute set to true, and <location> in the request is valid, then the server MUST include a <responseCompletedLocation> element in the <findServiceResponse> containing a PIDF will all the elements of the <location> PIDF, plus any additional elements that the LoST server may have that pertain to that location.
If the request contains requestLocation, but the location was invalid, the server MUST NOT send a PIDF in the response. The locationInvalid response will indicate to the requestor that no PIDF should be expected in the response. requestLocation is independent of locationValidation in the request.

---

5. Relax NG Schema

TOC

The Relax NG schema in [RFC5222] (Hardie, T., Newton, A., Schulzrinne, H., and H. Tschofenig, “LoST: A Location-to-Service Translation Protocol,” August 2008.) is modified to be:

```
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 requestCompletedLocation {
        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?, responseCompletedLocation?,
      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
    }+
}

##
##       Completed Location in Responses
##
div {
```

```
  responseCompletedLocation =
    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*
  }
```

---

6. Security Considerations

TOC

The <responseCompletedLocation> contains more information than the requestor originally had. However, the response does provide any more fine grained location ability: the request needed to have sufficient information to uniquely locate the target in order to be valid, and the response only contains additional information the LoST server may know about the location. For this reason, the privacy implications of this feature are not any more significant that those raised in RFC5222. The addition of the PIDF in the response does not have any other security implications beyond those discussed in RFC5222.

---

7. IANA Considerations

TOC

TOC

7.1. Relax NG Schema Registration

URI: urn:ietf:params:xml:schema:lost2

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

Relax NG Schema: The Relax NG schema to be registered is contained in Section 6. Its first line is

default namespace = "urn:ietf:params:xml:ns:lost2"

and its last line is

}

7.2. LoST Namespace Registration

TOC

URI: urn:ietf:params:xml:ns:lost2

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

XML:

```
BEGIN
<?xml version="2.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1"/>
  <title>LoST Namespace</title>
</head>
<body>
  <h1>Namespace for LoST</h1>
  <h2>urn:ietf:params:xml:ns:lost2</h2>
<p>See <a href="http://www.rfc-editor.org/rfc/rfc5222.txt">
   RFC5222</a>.</p>
</body>
</html>
END
```

8. Normative References

TOC

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC5222]	Hardie, T., Newton, A., Schulzrinne, H., and H. Tschofenig, "LoST: A Location-to-Service Translation Protocol," RFC 5222, August 2008 (TXT).

Authors' Addresses

TOC

	Brian Rosen
	Neustar
	470 Conrad Dr
	Mars, PA 16046
	US
EMail:	br@brianrosen.net
	Henning Schulzrinne
	Columbia University
	Department of Computer Science
	450 Computer Science Building
	New York, NY 10027
	US
Phone:	+1 212 939 7042
EMail:	hgs@cs.columbia.edu
URI:	http://www.cs.columbia.edu