ECRIT
Internet-Draft
Intended status: Standards Track
Expires: September 2, 2016

B. Rosen
NeuStar, Inc.
H. Schulzrinne
Columbia U.
H. Tschofenig
ARM Limited
R. Gellens
March 1, 2016

Data-Only Emergency Calls
draft-ietf-ecrit-data-only-ea-11.txt

Abstract

RFC 6443 'Framework for Emergency Calling Using Internet Multimedia' describes how devices use the Internet to place emergency calls and how Public Safety Answering Points (PSAPs) handle Internet multimedia emergency calls natively. The exchange of multimedia traffic for emergency services involves a SIP session establishment starting with a SIP INVITE that negotiates various parameters for that session.

In some cases, however, the transmission of application data is all that is needed. Examples of such environments include alerts issued by a temperature sensor, burglar alarm, or chemical spill sensor. Often these alerts are conveyed as one-shot data transmissions. These type of interactions are called 'data-only emergency calls'. This document describes a container for the data based on the Common Alerting Protocol (CAP) and its transmission using the SIP MESSAGE transaction.

Status of This Memo

Table of Contents

1. Introduction

RFC 6443 [RFC6443] describes how devices use the Internet to place emergency calls and how Public Safety Answering Points (PSAPs) handle Internet multimedia emergency calls natively. The exchange of multimedia traffic for emergency services involves a SIP session establishment starting with a SIP INVITE that negotiates various parameters for that session.

In some cases, however, there is only application data to be conveyed from the end devices to a PSAP or an intermediary. Examples of such environments includes sensors issuing alerts, or vehicles sending crash data. These messages may be one-shot alerts to emergency authorities and do not require establishment of a session. These type of interactions are called ’data-only emergency calls’. In this document, we use the term "call" so that similarities between data-only (non-interactive) alerts and sessions with interactive media are more obvious.

Data-only emergency calls are similar to regular emergency calls in the sense that they require the emergency indications, emergency call routing functionality and may even have the same location requirements. However, the communication interaction will not lead to the exchange of interactive media, that is, Real-Time Protocol packets, such as voice, video data or real-time text.

The Common Alerting Protocol (CAP) [cap] is a document format for exchanging emergency alerts and public warnings. CAP is mainly used for conveying alerts and warnings between authorities and from authorities to citizen/individuals. This document is concerned with citizen to authority "alerts", where the alert is a call without any interactive media.

This document describes a method of including a CAP message in a SIP transaction, either by value (the CAP message is in the body of the message, using a CID) or by reference (a URI is included in the message, which when dereferenced returns the CAP message) by defining it as a block of "additional data" as defined in [I-D.ietf-ecrit-additional-data]. The additional data mechanism is also used to send alert specific data beyond that available in the CAP message. This document also describes how a SIP MESSAGE [RFC3428] transaction can be used to send a data-only call.

2. Terminology

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 RFC 2119 [RFC2119].

3. Architectural Overview

This section illustrates two envisioned usage modes: targeted and location-based emergency alert routing.

1. Emergency alerts containing only data are targeted to an intermediary recipient responsible for evaluating the next steps. These steps could include:

    1. Sending a non-interactive call containing only data toward a Public Safety Answering Point (PSAP);

    2. Establishing a third-party initiated emergency call towards a PSAP that could include audio, video, and data.

2. Emergency alerts may be targeted to a Service URN used for IP-based emergency calls where the recipient is not known to the originator. In this scenario, the alert may contain only data (e.g., a CAP, Geolocation header field and one or more Call-Info header fields containing Additional Data [I-D.ietf-ecrit-additional-data] in a SIP MESSAGE).

Figure 1 shows a deployment variant where a sensor is pre-configured (using techniques outside the scope of this document) to issue an alert to an aggregator that processes these messages and performs whatever steps are necessary to appropriately react to the alert. For example, a security firm may use different sensor inputs to dispatch their security staff to a building they protect or to initiate a third-party emergency call.

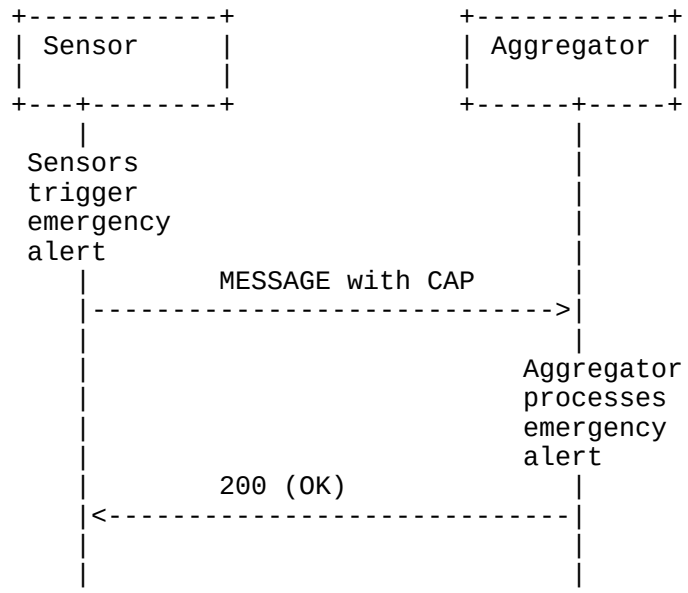

Figure 1: Targeted Emergency Alert Routing

In Figure 2 a scenario is shown whereby the alert is routed using location information and a Service URN. An emergency services routing proxy (ESRP) may use LoST to determine the next hop proxy to route the alert message to. A possible receiver is a PSAP and the recipient of the alert may be a call taker. In the generic case, there is very likely no prior relationship between the originator and the receiver, e.g., a PSAP. A PSAP, for example, is likely to receive and accept alerts from entities it cannot authorize. This scenario corresponds to the classic emergency services use case and the description in [RFC6881] is applicable. In this use case, the only difference between an emergency call and an emergency data-only call is that the former uses INVITE, creates a session, and negotiates one or more media streams, while the latter uses MESSAGE, does not create a session, and does not have media.

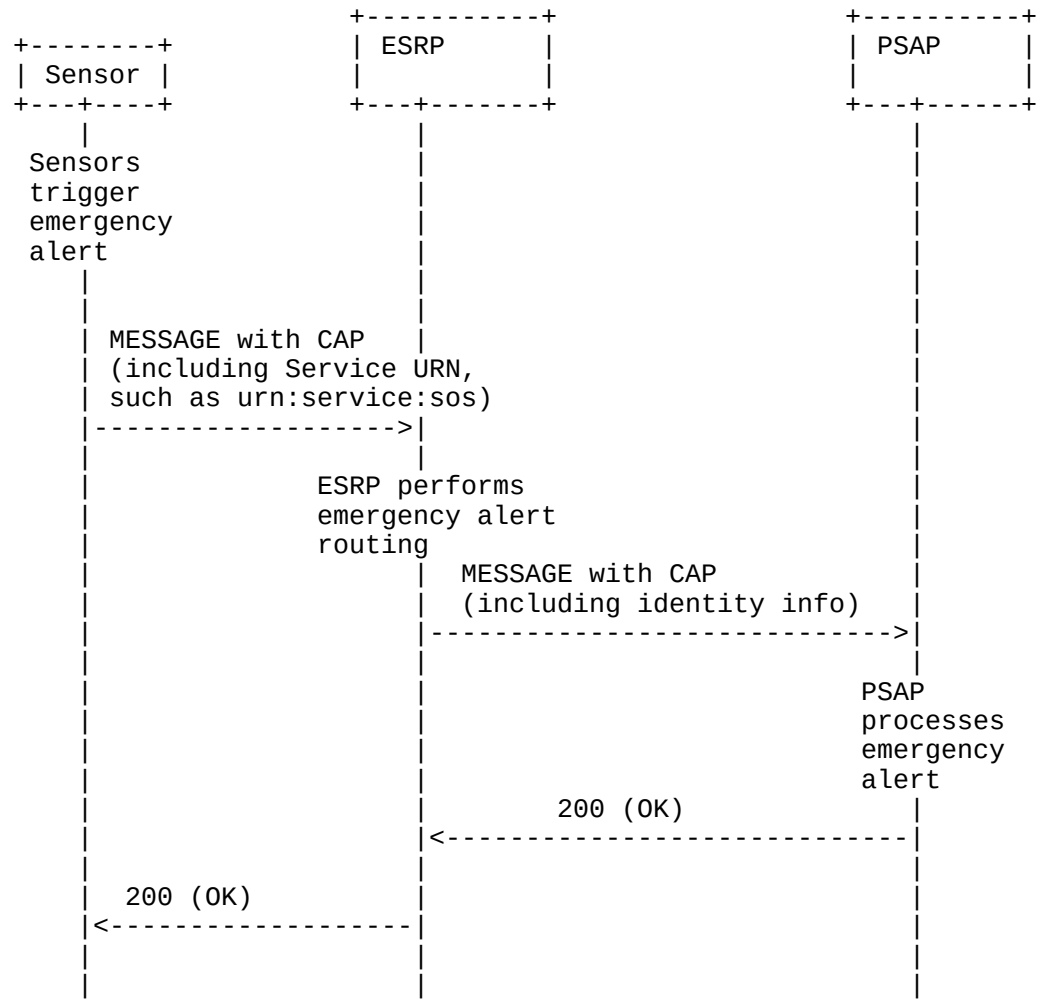

Figure 2: Location-Based Emergency Alert Routing

4. Protocol Specification

4.1. CAP Transport

A CAP message may be sent in the initial message of any SIP transaction. However, this document only addresses sending a CAP message in a SIP INVITE that initiates an emergency call, or in a SIP MESSAGE transaction for a one-shot, data-only emergency call. Behavior with other transactions is not defined.

The CAP message is included in a SIP message as an additional-data block [I-D.ietf-ecrit-additional-data]. Accordingly, it is introduced to the SIP message with a Call-Info header field with a purpose of "emergencyCall.cap". The header field may contain a URI that is used by the recipient (or in some cases, an intermediary) to obtain the CAP message. Alternative, the Call-Info header field may contain a Content Indirect url [RFC2392] and the CAP message included in the body of the message. In either case, the CAP message is located in a MIME block of the type 'application/ emergencyCall.cap+xml'.

If the SIP server does not support the functionality required to fulfill the request then a 501 Not Implemented MUST be returned as specified in RFC 3261 [RFC3261]. This is the appropriate response when a UAS does not recognize the request method and is not capable of supporting it for any user.

The 415 Unsupported Media Type error MUST be returned as specified in RFC 3261 [RFC3261] if the SIP server is refusing to service the request because the message body of the request is in a format not supported by the server for the requested method. The server MUST return a list of acceptable formats using the Accept, Accept-Encoding, or Accept-Language header fields, depending on the specific problem with the content.

4.2. Profiling of the CAP Document Content

The usage of CAP MUST conform to the specification provided with [cap]. For usage with SIP the following additional requirements are imposed:

sender: The following restrictions and conditions apply to setting the value of the <sender> element:

Originator is a SIP entity, Author indication irrelevant: When the alert was created by a SIP-based originator and it is not useful to be explicit about the author of the alert, then the <sender> element MUST be populated with the SIP URI of the user agent.

Originator is a non-SIP entity, Author indication irrelevant: When the alert was created by a non-SIP based entity and the identity of this original sender is to be preserved, then this identity MUST be placed into the <sender> element. In this situation it is not useful to be explicit about the author of the alert. The specific type of identity being used will depend on the technology used by the original originator.

Author indication relevant: When the author is different from the actual originator of the message and this distinction should be preserved, then the <sender> element MUST NOT contain the SIP URI of the user agent.

incidents: The <incidents> element MUST be present. This incident identifier MUST be chosen in such a way that it is unique for a given <sender, expires, incidents> combination. Note that the <expires> element is optional and may not be present.

scope: The value of the <scope> element MAY be set to "Private" if the alert is not meant for public consumption. The <addresses> element is, however, not used by this specification since the message routing is performed by SIP and the respective address information is already available in other SIP header fields. Populating information twice into different parts of the message may lead to inconsistency.

parameter: The <parameter> element MAY contain additional information specific to the sendor.

area: It is RECOMMENDED to omit this element when constructing a message. If the CAP message already contains an <area> element, then the specified location information SHOULD be copied into the PIDF-LO structure of the ’geolocation’ header field.

4.3. Sending a Data-Only Emergency Call

A data-only emergency call is sent using a SIP MESSAGE transaction with a CAP URI or body as described above in a manner similar to how an emergency call with interactive media is sent, as described in [RFC6881]. The MESSAGE transaction does not create a session nor send media, but otherwise, the header content of the transaction, routing, and processing of data-only calls are the same as those of other emergency calls.

5. Error Handling

This section defines a new error response code and a header field for additional information.

5.1. 425 (Bad Alert Message) Response Code

This SIP extension creates a new location-specific response code, defined as follows:

425 (Bad Alert Message)

The 425 response code is a rejection of the request due to its included alert content, indicating that it was malformed or not satisfactory for the recipient’s purpose.

A SIP intermediary can also reject an alert it receives from a UA when it understands that the provided alert is malformed.

Section 5.2 describes an AlertMsg-Error header field with more details about what was wrong with the alert message in the request. This header field MUST be included in the 425 response.

It is only appropriate to generate a 425 response when the responding entity has no other information in the request that is usable by the responder.

A 425 response code MUST NOT be sent in response to a request that lacks an alert message, as the user agent in that case may not support this extension.

A 425 response is a final response within a transaction, and MUST NOT terminate an existing dialog.

5.2. The AlertMsg-Error Header Field

The AlertMsg-Error header field provides additional information about what was wrong with the original request. In some cases the provided information will be used for debugging purposes.

The AlertMsg-Error header field has the following ABNF [RFC5234]:

```
message-header   /= AlertMsg-Error
                        ; (message-header from 3261)
AlertMsg-Error   = "AlertMsg-Error" HCOLON
                        ErrorValue
ErrorValue       =  error-code
                         *(SEMI error-params)
error-code       = 1*3DIGIT
error-params     = error-code-text
                         / generic-param ; from RFC3261
error-code-text  = "code" EQUAL quoted-string ; from RFC3261
```

HCOLON, SEMI, and EQUAL are defined in RFC3261 [RFC3261]. DIGIT is defined in RFC5234 [RFC5234].

The AlertMsg-Error header field MUST contain only one ErrorValue to indicate what was wrong with the alert payload the recipient determined was bad.

The ErrorValue contains a 3-digit error code indicating what was wrong with the alert in the request. This error code has a corresponding quoted error text string that is human understandable. The text string are OPTIONAL, but RECOMMENDED for human readability, similar to the string phrase used for SIP response codes. That said, the strings are complete enough for rendering to the user, if so desired. The strings in this document are recommendations, and are not standardized -- meaning an operator can change the strings -- but MUST NOT change the meaning of the error code. Similar to how RFC 3261 specifies, there MUST NOT be more than one string per error code.

The AlertMsg-Error header field MAY be included in any response if an alert message was in the request part of the same transaction. For example, a UA includes an alert in an MESSAGE to a PSAP. The PSAP can accept this MESSAGE, thus creating a dialog, even though its UA determined that the alert message contained in the MESSAGE was bad. The PSAP merely includes an AlertMsg-Error header field value in the 200 OK to the MESSAGE, thus informing the UA that the MESSAGE was accepted but the alert provided was bad.

If, on the other hand, the PSAP cannot accept the transaction without a suitable alert message, a 425 response is sent.

A SIP intermediary that requires the UA’s alert message in order to properly process the transaction may also sends a 425 with an AlertMsg-Error code.

This document defines an initial list of AlertMsg-Error values for any SIP response, including provisional responses (other than 100 Trying) and the new 425 response. There MUST be no more than one AlertMsg-Error code in a SIP response.

AlertMsg-Error: 100 ; code="Cannot Process the Alert Payload"

AlertMsg-Error: 101 ; code="Alert Payload was not present or could not be found"

AlertMsg-Error: 102 ; code="Not enough information to determine the purpose of the alert"

AlertMsg-Error: 103 ; code="Alert Payload was corrupted"

Additionally, if an entity cannot or chooses not to process the alert message from a SIP request, a 500 (Server Internal Error) SHOULD be used with or without a configurable Retry-After header field.

6. Updates to the CAP Message

If the sender anticipates that the content of the CAP message may need to be updated during the lifecycle of the event referred to in the message, it may include an update block as defined in [I-D.rosen-ecrit-addldata-subnot].If the sender includes an update block and does not have a globally reachable URI, then the UA must register it’s contact with a Registrar, and include a GRUU in in the update block

7. Call Backs

This document does not describe any method for the recipient to call back the sender of a data-only call. Usually, these alerts are sent by automata, which do not have a mechanism to receive calls of any kind. The identifier in the ’From’ header field may be useful to obtain more information, but any such mechanism is not defined in this document. The CAP message may contain related contact information for the sender.

8. Handling Large Amounts of Data

It is not atypical for sensors to have large quantities of data that they may wish to send. Including large amounts of data in a MESSAGE is not advisable, because SIP entities are usually not equipped to handle very large messages. In such cases, the sender SHOULD make use of the by-reference mechanisms defined in [I-D.ietf-ecrit-additional-data], which involves making the data available via HTTPS (either at the originator or at another entity), placing a URI to the data in the ’Call-Info’ header field, and the recipient using HTTPS to retrieve the data. The CAP message itself can be sent by-reference using this mechanism, as well as any or all of the Additional Data blocks that may contain sensor-specific data.

9. Example

Figure 3 shows a CAP document indicating a BURGLARY alert issued by a sensor called ’sensor1@domain.com’. The location of the sensor can be obtained from the attached location information provided via the ’geolocation’ header field contained in the SIP MESSAGE structure. Additionally, the sensor provided some data along with the alert message, using proprietary information elements intended only to be

processed by the receiver, a SIP entity acting as an aggregator. This example reflects the description in Figure 1.

```
MESSAGE sip:aggregator@domain.com SIP/2.0
Via: SIP/2.0/TCP sensor1.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
From: sip:sensor1@domain.com;tag=49583
To: sip:aggregator@domain.com
Call-ID: asd88asd77a@1.2.3.4
Geolocation: <cid:abcdef@domain.com>
  ;routing-allowed=yes
Supported: geolocation
Accept: application/pidf+xml, application/emergencyCall.cap+xml
CSeq: 1 MESSAGE
Call-Info: cid:abcdef2@domain.com;purpose=emergencyCall.cap
Content-Type: multipart/mixed; boundary=boundary1
Content-Length: ...

--boundary1

Content-Type: application/emergencyCall.cap+xml
Content-ID: <abcdef2@domain.com>
Content-Disposition: by-reference;handling=optional
<?xml version="1.0" encoding="UTF-8"?>

<alert xmlns="urn:oasis:names:tc:emergency:cap:1.1">
  <identifier>S-1</identifier>
  <sender>sip:sensor1@domain.com</sender>
  <sent>2008-11-19T14:57:00-07:00</sent>
  <status>Actual</status>
  <msgType>Alert</msgType>
  <scope>Private</scope>
  <incidents>abc1234</incidents>
  <info>
      <category>Security</category>
      <event>BURGLARY</event>
      <urgency>Expected</urgency>
      <certainty>Likely</certainty>
      <severity>Moderate</severity>
      <senderName>SENSOR 1</senderName>
      <parameter>
        <valueName>SENSOR-DATA-NAMESPACE1</valueName>
        <value>123</value>
      </parameter>
      <parameter>
        <valueName>SENSOR-DATA-NAMESPACE2</valueName>
        <value>TRUE</value>
```

```
          </parameter>
       </info>
      </alert>

      --boundary1

      Content-Type: application/pidf+xml
      Content-ID: <abcdef2@domain.com>
      Content-Disposition: by-reference;handling=optional
      <?xml version="1.0" encoding="UTF-8"?>
          <presence
             xmlns="urn:ietf:params:xml:ns:pidf"
             xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
             xmlns:gbp=
                    "urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
             xmlns:cl="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
             xmlns:gml="http://www.opengis.net/gml"
             xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
             entity="pres:alice@atlanta.example.com">
           <dm:device id="sensor">
             <gp:geopriv>
               <gp:location-info>
                 <gml:location>
                   <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:pos>32.86726 -97.16054</gml:pos>
                   </gml:Point>
                </gml:location>
               </gp:location-info>
               <gp:usage-rules>
                 <gbp:retransmission-allowed>false
                 </gbp:retransmission-allowed>
                 <gbp:retention-expiry>2010-11-14T20:00:00Z
                 </gbp:retention-expiry>
               </gp:usage-rules>
               <gp:method>802.11</gp:method>
             </gp:geopriv>
             <dm:timestamp>2010-11-04T20:57:29Z</dm:timestamp>
           </dm:device>
         </presence>
      --boundary1--
```

Figure 3: Example Message conveying an Alert to an Aggregator

Figure 4 shows the same CAP document sent as a data-only emergency call towards a PSAP.

```
MESSAGE urn:service:sos SIP/2.0
```

```
Via: SIP/2.0/TCP sip:aggreg.1.example.com;branch=z9hG4bK776absa
Max-Forwards: 70
From: sip:aggregator@example.com;tag=32336
To: 112
Call-ID: asdf33443a@example.com
Route: sip:psap1.example.gov
Geolocation: <cid:abcdef@example.com>
  ;routing-allowed=yes
Supported: geolocation
Accept: application/pidf+xml, application/emergencyCall.cap+xml
Call-info: cid:abcdef2@domain.com;purpose=emergencyCall.cap
CSeq: 1 MESSAGE
Content-Type: multipart/mixed; boundary=boundary1
Content-Length: ...

--boundary1

Content-Type: application/emergencyCall.cap+xml
Content-ID: <abcdef2@example.com>
<?xml version="1.0" encoding="UTF-8"?>

<alert xmlns="urn:oasis:names:tc:emergency:cap:1.1">
  <identifier>S-1</identifier>
  <sender>sip:sensor1@domain.com</sender>
  <sent>2008-11-19T14:57:00-07:00</sent>
  <status>Actual</status>
  <msgType>Alert</msgType>
  <scope>Private</scope>
  <incidents>abc1234</incidents>
  <info>
      <category>Security</category>
      <event>BURGLARY</event>
      <urgency>Expected</urgency>
      <certainty>Likely</certainty>
      <severity>Moderate</severity>
      <senderName>SENSOR 1</senderName>
      <parameter>
        <valueName>SENSOR-DATA-NAMESPACE1</valueName>
        <value>123</value>
      </parameter>
      <parameter>
        <valueName>SENSOR-DATA-NAMESPACE2</valueName>
        <value>TRUE</value>
      </parameter>
  </info>
</alert>

--boundary1
```

```
Content-Type: application/pidf+xml
Content-ID: <abcdef2@domain.com>
<?xml version="1.0" encoding="UTF-8"?>
    <presence
       xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gbp=
              "urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
       xmlns:cl="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
       xmlns:gml="http://www.opengis.net/gml"
       xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
       entity="pres:alice@atlanta.example.com">
     <dm:device id="sensor">
       <gp:geopriv>
         <gp:location-info>
           <gml:location>
             <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
               <gml:pos>32.86726 -97.16054</gml:pos>
             </gml:Point>
           </gml:location>
         </gp:location-info>
         <gp:usage-rules>
           <gbp:retransmission-allowed>false
           </gbp:retransmission-allowed>
           <gbp:retention-expiry>2010-11-14T20:00:00Z
           </gbp:retention-expiry>
         </gp:usage-rules>
         <gp:method>802.11</gp:method>
       </gp:geopriv>
       <dm:timestamp>2010-11-04T20:57:29Z</dm:timestamp>
     </dm:device>
   </presence>
--boundary1--
```

Figure 4: Example Message conveying an Alert to a PSAP

10. Security Considerations

This section discusses security considerations when SIP user agents issue emergency alerts utilizing MESSAGE and CAP. Location specific threats are not unique to this document and are discussed in [RFC7378] and [RFC6442].

The ECRIT emergency services architecture [RFC6443] considers classic individual-to-authority emergency calling where the identity of the emergency caller does not play a role at the time of the call establishment itself, i.e., a response to the emergency call does not depend on the identity of the caller. In the case of emergency

alerts generated by devices such as sensors, the processing may be different in order to reduce the number of falsely generated emergency alerts. Alerts may get triggered based on certain sensor input that may have been caused by factors other than the actual occurrence of an alert relevant event. For example, a sensor may simply be malfunctioning. For this reason, not all alert messages are directly sent to a PSAP, but rather may be pre-processed by a separate entity, potentially under supervision by a human, to filter alerts and potentially correlate received alerts with others to obtain a larger picture of the ongoing situation.

In any case, for alerts initiated by sensors, the identity may play an important role in deciding whether to accept or ignore an incoming alert message. With the scenario shown in Figure 1 it is very likely that only authorized sensor input will be processed. For this reason, it needs to be possible to refuse to accept alert messages from an unknown origin. Two types of information elements can be used for this purpose:

1. SIP itself provides security mechanisms that allow the verification of the originator's identity. These mechanisms can be re-used, such as P-Asserted-Identity [RFC3325] or SIP Identity [RFC4474]. The latter provides a cryptographic assurance while the former relies on a chain of trust model.

2. CAP provides additional security mechanisms and the ability to carry further information about the sender's identity. Section 3.3.2.1 of [cap] specifies the signing algorithms of CAP documents.

In addition to the desire to perform identity-based access control, the classic communication security threats need to be considered, including integrity protection to prevent forgery or replay of alert messages in transit. To deal with replay of alerts, a CAP document contains the mandatory <identifier>, <sender>, <sent> elements and an optional <expire> element. Together, these elements make the CAP document unique for a specific sender and provide time restrictions. An entity that has already received a CAP message within the indicated timeframe is able to detect a replayed message and, if the content of that message is unchanged, then no additional security vulnerability is created. Additionally, it is RECOMMENDED to make use of SIP security mechanisms, such as SIP Identity [RFC4474], to tie the CAP message to the SIP message. To provide protection of the entire SIP message exchange between neighboring SIP entities, the usage of TLS is MANDATORY.

Note that none of the security mechanism in this document protect against a compromised sensor sending crafted alerts.

11. IANA Considerations

11.1. Registration of the 'application/emergencyCall.cap+xml' MIME type

To: ietf-types@iana.org

Subject: Registration of MIME media type application/emergencyCall.cap+xml

MIME media type name: application

MIME subtype name: cap+xml

Required parameters: (none)

Optional parameters: charset; Indicates the character encoding of enclosed XML. Default is UTF-8 [RFC3629].

Encoding considerations: Uses XML, which can employ 8-bit characters, depending on the character encoding used. See RFC 3023 [RFC3023], Section 3.2.

Security considerations: This content type is designed to carry payloads of the Common Alerting Protocol (CAP). RFC XXX [Replace by the RFC number of this specification] discusses security considerations for this.

Interoperability considerations: This content type provides a way to convey CAP payloads.

Published specification: RFC XXX [Replace by the RFC number of this specification].

Applications which use this media type: Applications that convey alerts and warnings according to the CAP standard.

Additional information: OASIS has published the Common Alerting Protocol at http://www.oasis-open.org/committees/documents.php&wg_abbrev=emergency

Person and email address to contact for further information: Hannes Tschofenig, hannes.tschofenig@gmx.net

Intended usage: Limited use

Author/Change controller: IETF ECRIT working group

Other information: This media type is a specialization of application/xml RFC 3023 [RFC3023], and many of the considerations described there also apply to application/cap+xml.

11.2. IANA Registration of 'cap' Additional Data Block

This document registers a new block type in the sub-registry called 'Additional Data Blocks' defined in [I-D.ietf-ecrit-additional-data]. The token is "cap" and the reference is this document.

11.3. IANA Registration for 425 Response Code

In the SIP Response Codes registry, the following is added

Reference: RFC-XXXX (i.e., this document)

Response code: 425 (recommended number to assign)

Default reason phrase: Bad Alert Message

Registry:

Response Code	Reference
Request Failure 4xx	
425 Bad Alert Message	[this doc]

This SIP Response code is defined in Section 5.

11.4. IANA Registration of New AlertMsg-Error Header Field

The SIP AlertMsg-error header field is created by this document, with its definition and rules in Section 5, to be added to the IANA Session Initiation Protocol (SIP) Parameters registry with two actions:

1. Update the Header Fields registry with

Registry: Header Name	compact	Reference
AlertMsg-Error		[this doc]

2. In the portion titled "Header Field Parameters and Parameter Values", add

Header Field	Parameter Name	Predefined Values	Reference
AlertMsg-Error	code	yes	[this doc]

11.5. IANA Registration for the SIP AlertMsg-Error Codes

This document creates a new registry for SIP, called "AlertMsg-Error Codes". AlertMsg-Error codes provide reasons for an error discovered by a recipient, categorized by the action to be taken by the error recipient. The initial values for this registry are shown below.

Registry Name: AlertMsg-Error Codes

Reference: [this doc]

Registration Procedures: Specification Required

Code	Default Reason Phrase	Reference
100	"Cannot Process the Alert Payload"	[this doc]
101	"Alert Payload was not present or could not be found"	[this doc]
102	"Not enough information to determine the purpose of the alert"	[this doc]
103	"Alert Payload was corrupted"	[this doc]

Details of these error codes are in Section 5.

12. Acknowledgments

The authors would like to thank the participants of the Early Warning adhoc meeting at IETF#69 for their feedback. Additionally, we would like to thank the members of the NENA Long Term Direction Working Group for their feedback.

Additionally, we would like to thank Martin Thomson, James Winterbottom, Shida Schubert, Bernard Aboba, and Marc Linsner for their review comments.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

[cap] Jones, E. and A. Botterell, "Common Alerting Protocol v. 1.1", October 2005.

[RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, DOI 10.17487/RFC2392, August 1998, <http://www.rfc-editor.org/info/rfc2392>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[RFC3428] Campbell, B., Ed., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, DOI 10.17487/RFC3428, December 2002, <http://www.rfc-editor.org/info/rfc3428>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, DOI 10.17487/RFC3023, January 2001, <http://www.rfc-editor.org/info/rfc3023>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC6442] Polk, J., Rosen, B., and J. Peterson, "Location Conveyance for the Session Initiation Protocol", RFC 6442, DOI 10.17487/RFC6442, December 2011, <http://www.rfc-editor.org/info/rfc6442>.

[RFC6881] Rosen, B. and J. Polk, "Best Current Practice for Communications Services in Support of Emergency Calling", BCP 181, RFC 6881, DOI 10.17487/RFC6881, March 2013, <http://www.rfc-editor.org/info/rfc6881>.

[I-D.ietf-ecrit-additional-data] Gellens, R., Rosen, B., Tschofenig, H., Marshall, R., and J. Winterbottom, "Additional Data Related to an Emergency Call", draft-ietf-ecrit-additional-data-37 (work in progress), October 2015.

[I-D.rosen-ecrit-addldata-subnot] Rosen, B., "Updating Additional Data related to an Emergency Call using Subscribe/ Notify", draft-rosen-ecrit-addldata-subnot-01 (work in progress), November 2013.

13.2. Informative References

[RFC7378] Tschofenig, H., Schulzrinne, H., and B. Aboba, Ed., "Trustworthy Location", RFC 7378, DOI 10.17487/RFC7378, December 2014, <http://www.rfc-editor.org/info/rfc7378>.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, DOI 10.17487/RFC4474, August 2006, <http://www.rfc-editor.org/info/rfc4474>.

[RFC3325] Jennings, C., Peterson, J., and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, DOI 10.17487/RFC3325, November 2002, <http://www.rfc-editor.org/info/rfc3325>.

[RFC6443] Rosen, B., Schulzrinne, H., Polk, J., and A. Newton, "Framework for Emergency Calling Using Internet Multimedia", RFC 6443, DOI 10.17487/RFC6443, December 2011, <http://www.rfc-editor.org/info/rfc6443>.

Authors' Addresses

Brian Rosen
NeuStar, Inc.
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 7004
Email: hgs+ecrit@cs.columbia.edu
URI: http://www.cs.columbia.edu

Hannes Tschofenig
ARM Limited
Austria

Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Randall Gellens

Email: rg+ietf@randy.pensive.org

ECRIT
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

R. Marshall
J. Martin
TCS
B. Rosen
Neustar

March 21, 2016

A LoST extension to return complete and similar location info
draft-ietf-ecrit-similar-location-02

Abstract

This document introduces a new way to provide returned location information in LoST responses that is either of a completed or similar form to the original input civic location, based on whether valid or invalid civic address elements are returned within the findServiceResponse message. This document defines a new extension to the findServiceResponse message within the LoST protocol [RFC5222] that enables the LoST protocol to return a completed civic address element set for a valid location response, and one or more suggested sets of similar location information for invalid LoST responses. These two types of civic addresses are referred to as either "complete location" or "similar location", and are included as a compilation of CAtype xml elements within the existing LoST findServiceResponse message structure.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 22, 2016.

Table of Contents

1. Introduction

The LoST protcol [RFC5222] supports the validation of civic location information as input, by providing a set of validation result status indicators. The current usefulness of the supported xml elements, "valid", "invalid", and "unchecked", is limited, because while they each provide an indication of validity for any one location element as a part of the whole civic address, the mechanism is insufficient in providing either the complete set of civic address elements that the LoST server contains, or of providing alternate suggestions (hints) as to which civic address is intended for use.

Whether the input civic location is valid and missing information, or invalid due to missing or wrong information during input, this

document provides a mechanism to return a complete set of civic address elements for those valid or invalid cases.

This enhancement to the validation feature within LoST is required by systems that rely on accurate location for processing in order to increase the likelihood that the correct and/or complete form of a civic location becomes known in those cases where it is incomplete or just plain wrong. One such use case is that of location based emergency calling. The use of this protocol extension will protocol extension will facilitate the correction of errors, and allow location servers to be more easily provisioned with complete address information.

The structure of this document includes terminology, Section 2, followed by a discussion of the basic elements involved in location validation. The use of these elements, by way of example, is discussed in an overview section, Section 3, with accompanying rationale, and a brief discussion of the impacts to LoST, and its current schema.

2. Terminology

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 RFC 2119 [RFC2119].

The following terms are defined in this document:

Location: The term Location can be used to refer to either a civic location or a geodetic location.

Geodetic Location: a geographic coordinate set of values that describes a point within a defined geographic datum. For example, a WGS84 referenced lattitude, longitude coordinate pair (2D), or lattitude, longitude, and altitude (3D). Note: geodetic location is defined here for context, but is not used elsewhere within this document.

Civic Location: The term civic location applies to a set of one or more civic address elements that are used in conjunction with each other, and in accordance with a known ruleset to to designate a specific place within a region of geography, or a region of geography by itself as defined in [RFC5139].

Civic Address: The term Civic Address is used interchangeably with the term Civic Location within this document.

Civic Address Element: The term Civic Address Element is used within this document to apply to an individual CAtype data descriptor, for example, as is described in [RFC4776], [RFC5774], and [RFC6848].

Invalid: The status result of the unsuccessful attempt to match an individual input data as part of a larger set of data that has already been successfully matched and as shown by the [RFC5222] defined xml named element.

Valid: The status result of the successful attempt to match an individual input data as part of a larger set of data that has already also been successfully matched and shown by the [RFC5222] defined xml named element.

Invalid Location: A Civic Location that was included in a LoST request and subsequently returned with one or more civic address elements marked as invalid.

Valid Location: A Civic Location that was included in a LoST request and subsequently returned with all civic address elements marked as valid.

Complete Location: An expanded civic location that includes other civic address elements in addition to the existing validated civic address elements provided as input to a LoST server.

Similar Location: A suggested civic location that is comparatively close to the civic location which was input, but which had one or more invalid civic address elements returned by the LoST server.

Returned Location Information: A set of standard civic address elements returned in a LoST response.

3. Overview of Returned Location Information

This document describes an extension to LoST [RFC5222] to allow additional location information to be returned in a findServiceResponse for two different use cases.

When a LoST server is asked to validate a civic location, its goal is to take the set of civic address elements provided as the location information in the LoST request, and find a unique location in its database that matches the information in the request. Uniqueness might not require values for all possible elements in the civic address that the database might hold. Further, the input location information might not represent the form of location the users of the LoST service prefer to have. As an example, there are LoST civic

address elements that could be used to define a postal location, suitable for delivery of mail as well as a municipal location suitable for responding to an emergency call. While the LoST server might be able to determine the location from the postal elements provided, the emergency services would prefer that the municipal location be used for any subsequent emergency call. Since validation is often performed well in advance of an end-user placing an emergency call, if the LoST server could return the preferred form of location (or more properly in this example, the municipal elements in addition to the postal elements), those elements could be stored in a LIS and used in a later emergency call.

Since a LoST server often contains more data than what is included within a findService request, it is expected that this additional location information, if present, SHOULD only be returned within response messages that contain only valid civic address elements in the corresponding request, and where the set of valid civic address elements in the request identify a unique location. Where a LoST server contains additional location information relating to that civic address, the findServiceResponse message MAY return additional location information along with the original validated civic address elements in order to form a complete location based on local implementation policy.

In addition, this document describes the reuse of the same mechanism, but for a different purpose: to supply similar location information in the case where a LoST server response includes one or more civic address elements marked as invalid, constituting an invalid location response. In this case, the response contains one or more suggested alternative, but valid locations.

Clients MAY ignore the location information this extension defines in the response. The information is optional to send, and optional to use. In the case where the location information in the request was valid, this extension does not change the validity. In the case where the location information in the request is invalid, but alternate location information is returned, the original location remains invalid, and the LoST server does not change the mapping response other than optionally including the information defined by this extension.

In a valid location response, a LoST server returns a response to a findService request that contains a set of civic address elements marked valid. The location information in the findServiceResponse message MAY be extended to include additional location information specific for that location. As an example, the query might contain a HNO (house number), RD (road name) and A3 (city) and a few more CAtype elements, but might not contain A1 (state) or PC (Postal Code)

CAtypes. The HNO, RD, STS, POD, and A3 civic address elements might be sufficient enough to the LoST server to uniquely locate the address specified in the request and thus be considered valid. Yet, downstream entities might find it helpful to have the additional country, A1 (state), and PC, (Postal Code), civic address elements that are present within the LoST server, be included as part of a complete location response. Since [RFC5222] currently does not have a way for this additional location information to be returned in the findServiceResponse, this document extends the LoST protocol so that it can include a completeLocation element within the findServiceResponse message, allowing for the representation of complete location information.

An example showing complete location information supplied:

input address: 6000 15th Ave NW Seattle

complete location: 6000 15th Ave NW Seattle, WA 98105 US

By contrast, when invalid location is received from the LoST server, with this extension, the same mechanism works as follows: if a LoST server returns a response to a findService request that contains a set of civic address elements with one or more labeled as invalid, the location information in the findServiceResponse is extended to include additional location information that it suspects might be the location desired.

In the example cited above, policy at the LoST server might deem a missing A3 element as invalid, even if the location information in the request was sufficient to identify a unique address. In that case, the missing element would be listed in the invalid list, and similarLocation could be returned in the response showing the missing elements including A3, the same as the above example.

As another example of the use of similarLocation, consider the results based on a similar data set as used above, where the HNO, RD, STS, A1, and A3 civic address elements are not sufficient to locate a unique address, which leads to an invalid location result. This is the case, despite the fact that the LoST server typically contains additional civic address elements which could have resulted in a uniquely identifiable location if additional data had been supplied with the query. Since [RFC5222] currently does not have a way for this additional location information to be returned in the findServiceResponse, this document extends [RFC5222] so that the LoST findServiceResponse message can include one or more similarLocation elements within the findServiceResponse message representing similar civic locations.

To show this, suppose that a slightly modified address as above is inserted within a Lost findService request:

input address: 6000 15th Ave N Seattle, WA.

This time we make the assumption that the address is deemed "invalid" by the LoST server because there is no such thing as "15th Ave N" within the LoST server's data for the city of Seattle. However, we also happen to know for this example that there are two addresses within the address dataset that are "similar", when all parts of the address are taken as a whole. These similar addresses that could be suggested to the user are as follows:

similar address #1: 6000 15th Ave NW Seattle, WA 98107

similar address #2: 6000 15th Ave NE Seattle, WA 98105

This extension would allow the LoST server to include the above similar addresses as civicAddress elements in the response to locationValidation. The next section shows examples of the LoST request and response xml message fragments for the above valid and invalid scenarios, returning the complete or similar addresses, respectively.

4. Returned Location Information

The LoST server implementing this extension MAY include completeLocation or similarLocation in the findService response. completeLocation and similarLocation contain a list of civic address elements identical to the elements used in the location element with the "civic profile".

The LoST server MAY include more than one similarLocation elements in the response, but SHOULD NOT return more than a few possible similar locations. If there are too many possible locations, the server MAY return none, or it MAY return the few it considers most likely. The definition of "few" is left to the implementation of the LoST server. The server is unable to know what the intended location information was suppose to be; it is guessing. Therefore the correct location may or may not be one of the similarLocation elements the server provides, and the client cannot assume that any of them are the correct one.

4.1. Complete Location returned for Valid Location response

Based on the example input request, returned location information is provided in a findServiceResponse message when the original input

address is considered valid, but is missing some additional data that the LoST server has.

```
<!-- =====Request================================== -->

<findService xmlns="urn:ietf:params:xml:ns:lost1"
  validateLocation="true">

  <location id="587cd3880" profile="civic">
    <civicAddress
      xmlns="urn:ietf:params:mxl:ns:pidf:geopriv10:civicAddr">

      <A1>WA</A1>
      <A3>Seattle</A3>
      <RD>15th</RD>
      <STS>Ave</STS>
      <POD>NW</POD>
      <HNO>6000</HNO>

    </civicAddress>
  </location>

  <service>urn:service:sos</service>

</findService>

<!-- =====Response================================= -->

<findServiceResponse >
  xmlns="urn:ietf:params:xml:ns:lost1"
  xmlns:rli="urn:ietf:params:xml:ns:lost-rli1">
  xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">

  <mapping
    expires="NO-CACHE"
    lastUpdated="2006-11-01T01:00:00Z"
    source="authoritative.example"
    sourceId="8799e346000098aa3e">

    <displayName xml:lang="en">Seattle 911</displayName>
    <service>urn:service:sos</service>
    <uri>sip:seattle-911@example.com</uri>
    <serviceNumber>911</serviceNumber>
```

```
        </mapping>

        <locationValidation

          <valid>ca:A3 ca:RD ca:STS ca:POD ca:HNO</valid>
          <invalid></invalid>
          <unchecked></unchecked>

          <rli:completeLocation>  <!-- completed address -->
            <ca:civicAddress>
              <ca:country>US</ca:country>
              <ca:A1>WA</ca:A1>
              <ca:A3>SEATTLE</ca:A3>
              <ca:RD>15TH</ca:RD>
              <ca:STS>AVE</ca:STS>
              <ca:POD>NW</ca:POD>
              <ca:HNO>6000</ca:HNO>
              <ca:PC>98106</ca:PC>
              <ca:PCN>SEATTLE</ca:PCN>
            </ca:civicAddress>

          </rli:completeLocation>

        </locationValidation>

          <path>
            <via source="authoritative.example"/>
          </path>

          <locationUsed id="587cd3880"/>

        </findServiceResponse>


        <!-- ============================================== -->
```

4.2. Similar Location returned for Invalid Location response

The following example shows returned location information provided in a findServiceResponse message when the original input address is considered invalid, because of the unmatchable POD data (in this example) that the LoST server needs to provide a unique mapping.

```
<!-- =====Request=================================== -->
```

```
      <findService xmlns="urn:ietf:params:xml:ns:lost1"
        validateLocation="true">

        <location id="587cd3880" profile="civic">
          <civicAddress
            xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">

            <country>US</country>
            <A1>WA</A1>
            <A3>Seattle</A3>
            <RD>15th</RD>
            <STS>Ave</STS>
            <POD>N</POD>
            <HNO>6000</HNO>

          </civicAddress>
        </location>

        <service>urn:service:sos</service>

      </findService>

      <!-- =====Response=================================== -->


      <findServiceResponse>
        xmlns="urn:ietf:params:xml:ns:lost1"
        xmlns:rli="urn:ietf:params:xml:ns:lost-rli1">
        xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">

        <mapping
          expires="NO-CACHE"
          lastUpdated="2006-11-01T01:00:00Z"
          source="authoritative.example"
          sourceId="8799e346000098aa3e">

          <displayName xml:lang="en">Seattle 911</displayName>
          <service>urn:service:sos</service>
          <uri>sip:seattle-911@example.com</uri>
          <serviceNumber>911</serviceNumber>

        </mapping>

        <locationValidation

          <valid>ca:country ca:A1 ca:A3 ca:STS ca:RD</valid>
          <invalid>ca:POD</invalid>
          <unchecked>ca:HNO</unchecked>
```

```
          <rli:similarLocation>  <!-- similar location info -->
            <ca:civicAddress>  <!-- similar address #1 -->
              <ca:country>US</ca:country>
              <ca:A1>WA</ca:A1>
              <ca:A3>SEATTLE</ca:A3>
              <ca:RD>15TH</ca:RD>
              <ca:STS>AVE</ca:STS>
              <ca:POD>NW</ca:POD>
              <ca:HNO>6000</ca:HNO>
              <ca:PC>98106</ca:PC>
              <ca:PCN>SEATTLE</ca:PCN>
            </ca:civicAddress>

            <ca:civicAddress>  <!-- similar address #2 -->
              <ca:country>US</ca:country>
              <ca:A1>WA</ca:A1>
              <ca:A3>SEATTLE</ca:A3>
              <ca:RD>15TH</ca:RD>
              <ca:STS>AVE</ca:STS>
              <ca:POD>NE</ca:POD>
              <ca:HNO>6000</ca:HNO>
              <ca:PC>98105</ca:PC>
              <ca:PCN>SEATTLE</ca:PCN>
            </ca:civicAddress>
          </rli:similarLocation>

        </locationValidation>

          <path>
            <via source="authoritative.example"/>
          </path>

          <locationUsed id="587cd3880"/>

        </findServiceResponse>


        <!-- ================================================ -->
```

5. Relax NG schema

This section provides the Relax NG schema of LoST extensions in the compact form. The verbose form is included in a later section [to be supplied in a later version of this draft].

```
   namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
   default namespace ns1 = "urn:ietf:params:xml:ns:lost-rli1"

   ##
   ##       Extension to LoST to support returned location information
   ##
   start =
     returnedLocation

   div {
     returnedLocationResponse =
       element returnedLocationResponse {
         completeLocation, similarLocation, extensionPoint
       }
   }

   ##
   ##       completeLocation
   ##
   div {
     completeLocation =
       element location {
         attribute id { xsd:token },
         locationInformation
       }+
   }

   ##
   ##       similarLocation
   ##
   div {
     similarLocation =
       element location {
         attribute id { xsd:token },
         locationInformation
       }+
   }
   ##
   ##       Location Information
   ##
   div {
     locationInformation =
       extensionPoint+,
       attribute profile { xsd:NMTOKEN }?
   }

   ##
   ##       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 = notRLI*
}
```

6. Security Considerations

Whether the input to the LoST server is valid or invalid, the LoST server ultimately determines what it considers to be valid. Even in the case where the input location is valid, the requester still might not actually understand where that location is. For this kind of valid location use case, this described extension would typically return more location information than the requester started with, which might reveal more about the location. While this might be very desirable in some scenarios including, for example, supporting an emergency call, it might not be as desirable for other services. Individual LoST server implementations SHOULD consider the risk of releasing more detail verses the value in doing so. Generally, it is not expected that this would be a significant problem as the requester must have enough location information to be considered valid, which in most cases is enough to uniquely locate the address. Providing more CAtypes generally doesn’t actually reveal anything more. For invalid locations that are submitted, this extension would allow the LoST response to include location information which is

similar to what was input, again resulting in more information provided in the response than was known during input. LoST server implementations SHOULD evaluate the particular use cases where this extension is supported, and weigh the risks around its use. Many similar database services available today via the Internet offer similar features, such as "did you mean", and address completion, so this capability is not introducing any fundamentally new threat.

7. IANA Considerations

7.1. Relax NG Schema Registration

URI: urn:ietf:params:xml:schema:lost-rli1

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 7. Its first line is

default namespace = "urn:ietf:params:xml:ns:lost-rli1

and its last line is

}

7.2. LoST Namespace Registration

URI: urn:ietf:params:xml:ns:lost-rli1

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

XML:

```
BEGIN
<?xml version="2.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1"/>
  <title>LoST Planned Change Namespace</title>
</head>
<body>
  <h1>Namespace for LoST Returned Location Information extension</h1>
  <h2>urn:ietf:params:xml:ns:lost-rli1</h2>
<p>See <a href="http://www.rfc-editor.org/rfc/rfc????.txt">
   RFC????</a>.</p>
</body>
</html>
END
```

8. References

8.1. Normative References

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

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

8.2. Informative References

[RFC4776] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information", RFC 4776, DOI 10.17487/RFC4776, November 2006, <http://www.rfc-editor.org/info/rfc4776>.

[RFC5139] Thomson, M. and J. Winterbottom, "Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO)", RFC 5139, DOI 10.17487/RFC5139, February 2008, <http://www.rfc-editor.org/info/rfc5139>.

[RFC5774] Wolf, K. and A. Mayrhofer, "Considerations for Civic Addresses in the Presence Information Data Format Location Object (PIDF-LO): Guidelines and IANA Registry Definition", BCP 154, RFC 5774, DOI 10.17487/RFC5774, March 2010, <http://www.rfc-editor.org/info/rfc5774>.

[RFC6848] Winterbottom, J., Thomson, M., Barnes, R., Rosen, B., and R. George, "Specifying Civic Address Extensions in the Presence Information Data Format Location Object (PIDF-LO)", RFC 6848, DOI 10.17487/RFC6848, January 2013, <http://www.rfc-editor.org/info/rfc6848>.

Authors' Addresses

Roger Marshall
TeleCommunication Systems, Inc.
2401 Elliott Avenue
2nd Floor
Seattle, WA 98121
US

Email: rmarshall@telecomsys.com
URI: http://www.telecomsys.com

Jeff Martin
TeleCommunication Systems, Inc.
2401 Elliott Avenue
2nd Floor
Seattle, WA 98121
US

Email: jmartin@telecomsys.com
URI: http://www.telecomsys.com

Brian Rosen
Neustar
470 Conrad Dr
Mars, PA 16046
US

Email: br@brianrosen.net

ecrit
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

B. Rosen
Neustar

October 19, 2015

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

Abstract

This document defines an extension to LoST (RFC5222) that allows a planned change to the data in the LoST server to occur. Records that previously were valid will become invalid at a date in the future, and new locations will become valid after the date. The extension adds two elements to the <findservice> request: a URI to be used to inform the LIS that previously valid locations will be invalid after the planned change date, and add a date which requests the server to perform validation as of the date specified. It also adds an optional TTL element to the response, which informs all queriers the current expected lifetime of the validation.

Table of Contents

1. Introduction

This document describes an update to the LoST protocol [RFC5222] which allows a <findservice> request to optionally add a URI and a date to be used with planned changes to the underlying location information in the server. The URI is retained by the LoST server, associated with the data record that was validated, and used to notify the LIS (the LoST client) when a location which was previously valid will become invalid. The date is used by the client to ask the server to perform validation as of a future date. In addition to this mechanism, the lt;findserviceResponse> is also extended to provide a TTL for validation, after which the client should revalidate the location.

Validation of civic locations involves dealing with data that changes over time. A typical example is a portion of a county or province that was not part of a municipality is "annexed" to a municipality. Prior to the change, the content of the PIDF A3 element would be blank, or represent some other value and after the change would be the municipality that annexed that part of the county/province. This kind of annexation has an effectivity date and time (typically 00:00 on the first or last day of a month).

Records in a LIS must change around these kinds of events. The old record must be discarded, and a new, validated record must be loaded into the LIS. It is often difficult for the LIS operator to know

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

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

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

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

2. Conventions used in this document

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

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

3. <plannedChange> element

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

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

4. <locationInvalidated> object

When the server needs to invalidate a location where the client provided a URI in <plannedChange>, the server executes an HTTPS POST containing <locationInvalidated> to the URI previously provided. This is the notice from the server to the client that the location may be invalid and should be revalidated. <locationInvalidated> contains an asOf attribute that specifies when the location may become invalid. If the date/time in asOf is earlier than the time the <locationInvalidated> was sent, the location may already be invalid and the LIS should take immediate action. If the POST operation fails, the server MAY retry the operation immediately, and if it fails again, retry the operation at a later time.

5. uri Not Stored Warning

A new warning is added to the exceptionContainer, 'uriNotStored'. This warning MUST NOT be returned unless the plannedChange element was found in the corresponding request. The warning is returned when the server decides not to store the URI found in the plannedChange element. As discussed above, this may occur because, among other reasons, the policy at the server limits how many URIs will be stored

against a specific location, the uri is not well formed or the policy at the server has some other restriction on the feature.

6. TTL in Response

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

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

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

When a planned change is made, typically the TTL for the affected records is lowered, so that revalidation is forced soon after the change is implemented. It is not advisable to set the expiration precisely at the planned change time if a large number of records will be changed, since that would cause a large spike in traffic at the change time. Rather, the expiration time should have a random additional time added to it to spread out the load.

7. Relax NG Schema

The Relax NG schema in [RFC5222] is extended to include:

```
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
default namespace ns1 = "urn:ietf:params:xml:ns:lost-plannedChange1"

##
##       Extension to Location-to-Service Translation (LoST) Protocol
##       to support a planned change to location data
```

```
   ##
   ##       plannedChange is used in the extensionPoint of
   ##       commonRequestPattern in a findService request
   ##
   ##       locationInvalidated is used by the LoST server to notify a
   ##       LIS that a previously valid location may be (or will become)
   ##       invalid
   ##
   ##       ttl is used in the extensionPoint of
   ##       commonResponsePattern in a findService response
   ##
   ##       uriNotStored is a new warning to be used in a
   ##       exceptionContainer in the warnings element of a
   ##       findServiceResponse
   ##
   start =
     plannedChange
     | locationInvalidated
     | uriNotStored
   ##
   ##       plannedChange
   ##
   div {
     plannedChange =
       element plannedChange {
         attribute uri {
           xsd:anyURI }?,
         attribute asOf {
           xsd:dateTime }?,
         extensionPoint+
       }
   }

   ##
   ##       locationInvalidated
   ##
   div {
     locationInvalidated =
       element locationInvalidated {
         attribute asOf {
           xsd:dateTime }?,
         extensionPoint+
       }
   }

   ##
   ##       ttl
   ##
```

```
   div {
     ttl =
       element ttl {
         expires,
         extensionPoint+
       }
   }

   ##
   ##       uriNotStored
   ##
   div {
     uriNotStored =
       element uriNotStored { basicException }
   }

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

     ##
     ##         Any element not in the LoST namespace.
     ##
     notLostChange = 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 = notLostChanged*
   }
```

8. Security Considerations

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

The server is subject to abuse by clients because it is being asked to store something and may need to send data to an uncontrolled URI. Clients could request many URIs for the same location for example. The server MUST have policy that limits use of this mechanism by a given client. If the policy is exceeded, the server returns the uriNotStored warning. The server MUST validate that the content of the uri sent is syntactically valid and meets the 256 byte limit. When sending the locationInvalidated object to the uri stored, the server MUST protect itself against common http vulnerabilities.

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

9. IANA Considerations

9.1. Relax NG Schema Registration

URI: urn:ietf:params:xml:schema:lost-planedChange1

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 5. Its first line is

default namespace = "urn:ietf:params:xml:ns:lost-PlannedChange1

and its last line is

}

9.2. LoST Namespace Registration

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

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

XML:

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

10. Normative References

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

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

Author’s Address

Brian Rosen
Neustar
470 Conrad Dr
Mars, PA 16046
US

EMail: br@brianrosen.net