Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2008

V. Singh
H. Schulzrinne
Columbia U.
P. Boni
Verizon
July 8, 2007

Vehicle Info Event Package
draft-singh-simple-vehicle-info-01.txt

Status of this Memo

Copyright Notice

Abstract

This document defines a new SIP event package for updating and retrieving status information of vehicles. The information can include the vehicle's status and diagnostic information distributed by vehicle telematics systems. This event package is useful for fleet management and vehicle tracking applications. The event package is called vehicle-info event package and is applicable to all

types of vehicles like cars, buses, ships and aircraft. However, this document focuses on automobiles.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Vehicle-info Event Package Description 4
3.1. Message Flow Diagram 5
3.2. Vehicle Status Information 6
3.3. Vehicle Location Information 7
4. Vehicle-info Event Package 7
4.1. Event Package Name 7
4.2. Event Package Parameters 7
4.3. SUBSCRIBE Bodies 7
4.4. Subscription Duration 7
4.5. NOTIFY Bodies 8
4.6. Notifier Processing of SUBSCRIBE Requests 8
4.7. Notifier Generation of NOTIFY Requests 8
4.8. Subscriber Processing of NOTIFY Requests 8
4.9. Rate of Notifications 9
5. The Vehicle-info Document 9
5.1. Document Description 9
5.2. XML Schema 10
6. Example of vehicle-info XML 13
7. Security Considerations 13
7.1. Authorization Considerations 14
8. IANA Considerations 14
9. Acknowledgements 14
10. References 14
10.1. Normative References 14
10.2. Informative References 15
Authors' Addresses 16
Intellectual Property and Copyright Statements 18

1. Introduction

The Session Initiation Protocol (SIP) events framework described in RFC3265 [1] defines subscription-based event distribution mechanism for sending notification of events. The RFC 3265 requires the events to be related to the SIP state. However, there are many benefits of using such framework for delivering non-SIP related notifications. Also, there seems to be an interest in the community for application of SIP event mechanism to events not related to SIP [10].

Today, vehicle information is processed and communicated via vehicle telematics systems, which employ a multitude of standards and are used for a number of purposes, including remote diagnostics and reporting vehicle's mechanical and electronic problems or failures. Increasingly, navigational and entertainment applications are being deployed within such frameworks such as ITSA [11], GST [12].

The vehicle information can be used for monitoring and tracking purposes. For example, location information and movement related information (speed, acceleration and direction) can be used to perform location tracking and to ensure that the vehicle is moving within acceptable speed limits. The vehicle's location and moving status information can be described using RFC 4119 [2] and moving object status tracking [13] using the presence event package.

Other information useful for monitoring includes vehicle status information containing vehicle's state (e.g., ignition status) and vehicle's diagnostic information. Some of the vehicle's information can be directly used to infer presence information of users of the vehicle whereas other information is only useful in vehicle management (e.g., by car-rental companies).

The vehicle's information can be divided into two parts: the core vehicular information, that is a vehicle status, and the location and connectivity-related information, already standardized in presence and location specifications in RFC4119 [2].

The vehicle-info event package as described in this document can be used to distribute core vehicular information. The event package aggregates vehicle information from telematics systems into an XML-based data structure. This document specifies the schema of the proposed event package. The schema contains core vehicular information as well as other information, some of which can be mapped to device characteristics as defined in the presence data model. The proposed schema should be treated as a seed document, open for contributions from vehicle telematics industry and their standards institutions.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [3].

This document uses GML Moving Vehicle terms as described in GML [4]. It also introduces terminology from the On Board Diagnostics standard's (OBD-II) [5].

3. Vehicle-info Event Package Description

This event package defines an XML-based schema for information that can be used to manage vehicles. It also shows how an application can subscribe to the information related to a vehicle or use this information to derive presence information of a user. An application will subscribe to a vehicle using vehicle-info event package.

An application may use presence event package to subscribe to the vehicle's location and moving status information.

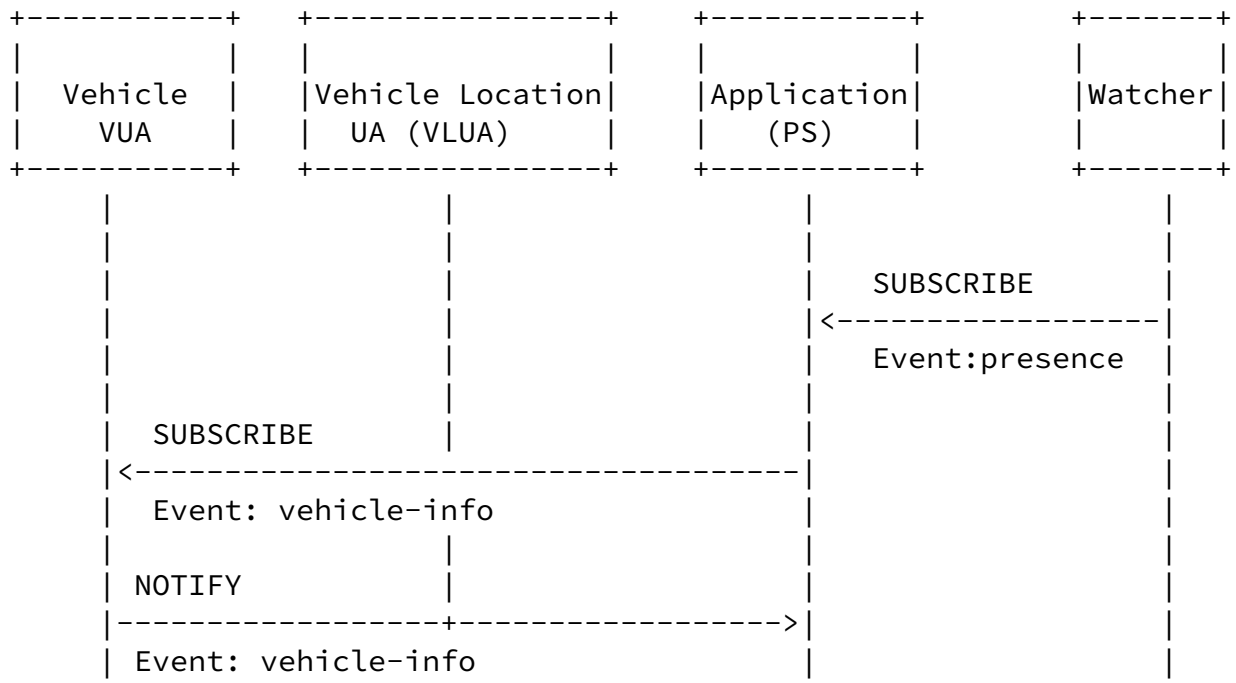
If the application is managing a vehicle (e.g., fleet management application), it sends SUBSCRIBE requests using the vehicle-info event package and presence event package to the vehicle's entity. The application will render the vehicle information in the specified XML format or in other formats such as PIDF [14], depending on the application processing functions.

Alternatively, if the application is managing user's (driver or passenger) presence data, it may compose user's presence information with vehicle as a device, which contributes to user's presence. The application sends SUBSCRIBE requests to vehicle's entity using the vehicle-info and presence event package.

A membership event package [15] can be used to track changes in membership in groups such as vehicle. If a user is a member of the vehicle group, a presence server would compose user's presentity PIDF based - among others - on the vehicle information provided. User may become a member of the vehicle group when a vehicle sensor discovers user's identity, using - for example - a Bluetooth technology or a smart card readout. However, the details of such person-vehicle association are beyond the scope of this document.

The information the vehicle-info event package provides consists of internal vehicle state data and the diagnostics.

3.1. Message Flow Diagram



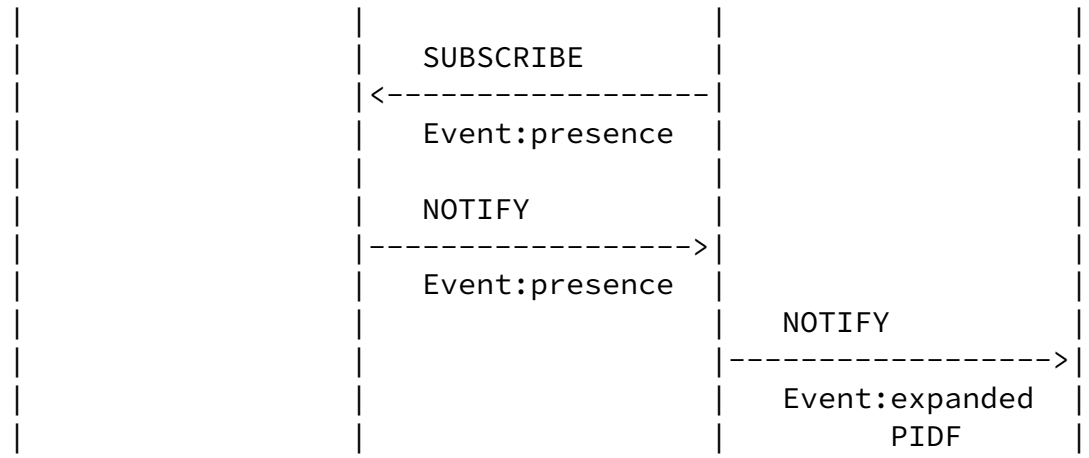


Figure 1: Vehicle-info event package to derive presence information (driver's PUA not shown)

When an application or a presence server receives a subscription request for a user currently in a vehicle, the presence server uses the membership event package to check if the user is inside the vehicle and then generates SUBSCRIBE requests. These requests are issued to presence agent of user, vehicle's presence user agent (VLUA), and vehicle's user agent for vehicle-info (VUA). VUA and VLUA are logical entities and can be collocated.

When NOTIFY messages from VUA and VLUA reach the presence server, they will be used to compose user's presence state and to send expanded PIDF to the requesting watcher. For example, the <person> element in PIDF/RPID might have an <activities> tag that indicates 'driving' if the car is moving. Also, the vehicle location information, if present, will be included in user's expanded PIDF. When user presence state already contains location, e.g., from a GPS-enabled cell phone, then there is a need to reconcile such information, obtained from two different location sources.

3.2. Vehicle Status Information

Vehicle status information is subdivided into the vehicle state information and the diagnostics data.

Vehicle status information is only partially standardized today. Telematics systems use different specifications to describe vehicle data. Manufacturers of vehicle control devices provide different static and dynamic information about the vehicle. In this document, we have focused on the information provided by OBD-II [5]. The OBD-II data is referred to as the diagnostics data. The vehicle state data may come from other, non OBD-II, systems.

OBD-II stands for the On-Board Diagnostics, generation II. OBD-II is a set of rules and regulations each car manufacturer has to follow to have their Engine Management System pass the U.S. federal emissions tests. It also allows to run a set of comprehensive diagnostics related to the engine and other parts of the vehicle using OBD-II scan tools. Most modern vehicles are OBD-II equipped. Diagnostics can be performed locally or remotely. OBD-II specification uses the Diagnostic Trouble Codes (DTC), which can be stored in the vehicle's Powertrain Control Module (PCM) and retrieved or obtained on demand by running a new diagnostic test.

The DTC is a five-byte code. The first byte is an ASCII character that identifies the part of the vehicle where the fault occurred. It can assume the values P, B, C or U. For example, "P" refers to the engine or transmission system. The second byte can take value of digit "0" for common fault code or "1" for proprietary one. The third byte indicates the vehicle subsystem at fault and can have values from 1 to 8. For example, "2" refers to a fuel system. The fourth and fifth byte indicate a specific code number for a given fault. Each DTC has also a short text associated with it.

The OBD-II based system can also provide the real time data such as engine coolant temperature, different types of fuel system data, engine RPM and speed.

Example of vehicle information data is presented in Section 6.

3.3. Vehicle Location Information

Assuming the vehicle or the agent representing it (e.g., VLUA in Figure 1) contains the location information about itself, it can convey this information using PIDF-LO [2], presence event package [16]. This requires the application (e.g., user's presence server)

to send SUBSCRIBE request using the presence event package to receive the location information and its updates.

If the user is a member of the vehicle group, the location information can be directly used to determine the user's location. Otherwise, location information only refers to the position of the vehicle.

4. Vehicle-info Event Package

4.1. Event Package Name

The name of this event package is "vehicle-info". This package name is carried in the SIP Event and Allow-Events header fields, as defined in RFC 3265 [1].

4.2. Event Package Parameters

RFC 3265 [1] allows event packages to define additional parameters carried in the Event header field. This event package does not define additional parameters.

4.3. SUBSCRIBE Bodies

The SUBSCRIBE bodies may contain the watcher filters (RFC 4660) [17] to specify triggers of when and what data the watcher is interested in. The mechanism to specify the filter remains same as specified in event filter format document [18].

4.4. Subscription Duration

The default expiration time for subscription to vehicle-info event package will be one day. Normally, a vehicle will be allocated to a task at a granularity of one day. However, this may change depending on the usage scenario, in which case the alternative expiration time will be specified by a subscriber in the Expires header field.

4.5. NOTIFY Bodies

According to RFC 3265, the NOTIFY message contains bodies in a format listed in the Accept header field of the SUBSCRIBE request or a package-specific default if the Accept header field was omitted from the SUBSCRIBE request. All subscribers and notifiers MUST support the "application/vehicle-info+xml" data format described in Section 4. By default, if no Accept header field is specified in a SUBSCRIBE request, the NOTIFY request will contain a body in the "application/vehicle-info+xml" data format. If the Accept header field is present, it MUST include "application/vehicle-info+xml" and MAY include any other types.

4.6. Notifier Processing of SUBSCRIBE Requests

RFC 3265 specifies that packages should define any package-specific processing of SUBSCRIBE requests at a notifier, specifically with regards to authentication and authorization.

Vehicle dynamic state information is a sensitive data. Therefore, all subscriptions to it SHOULD be authenticated and authorized before approval. Authentication MAY be performed by the techniques available through SIP, such as digest, S/MIME, TLS. Authorization policies for access need to be administered. We assume that in most cases applications will be subscribers, in which case authorization policies will be provided ahead of time.

SUBSCRIBE requests are addressed to the vehicle ID, typically vehicle's VIN, for example, 44G44444H4444@avis.com. The notifier will verify whether vehicle id is in the scope of Vehicle UA (VUA). The VUA may be collocated with the vehicle or it can be a network-based entity collocated with other VUA's.

4.7. Notifier Generation of NOTIFY Requests

Once the subscription is accepted, the notifier MAY send a NOTIFY request with the body of the most recent vehicle information data it has. Typically, it will send the NOTIFY request when any data item in the vehicle information data has changed. The body of the NOTIFY MUST be sent using one of the types listed in the Accept header field in the most recent SUBSCRIBE request, or using the type "application/vehicle-info+xml" if no Accept header field was present.

4.8. Subscriber Processing of NOTIFY Requests

The information from the vehicle's VUA sent in the body of NOTIFY request to the watcher conveys the vehicle states, such as ignition or fuel status.

4.9. Rate of Notifications

A notifier SHOULD NOT generate notifications for a single vehicle at a rate greater than once every 180 seconds in normal driving conditions. When the vehicle's engine is turned on or off, several notifications may be issued over the short period of time (10 seconds). In collision or accident situation, several notifications may be attempted to be sent within one second.

5. The Vehicle-info Document

A vehicle-info document is an XML document [6] that MUST be well-formed and SHOULD be valid. A vehicle-info document MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying vehicle-info documents. The namespace URI for elements defined by this specification is a URN (RFC 2141) [7], using the namespace identifier 'ietf' defined by RFC 2648 [8] and extended by RFC 3688 [9]. This URN is urn:ietf:params:xml:ns:vehicle-info

5.1. Document Description

A vehicle-info document starts with the root element tag <vehicle-info>. The root element consists of two child elements, <vehicle-state> and <vehicle-diagnostics>, described below. Other elements from different namespaces MAY be present for extensions; elements or attributes from unknown namespaces MUST be ignored.

There are two attributes associated with the <vehicle-info> element, both of which MUST be present: "version" and "state". The "version" attribute is a timestamp expressed in seconds. It MUST be represented by a 32 bit integer. Synchronization of vehicle-info documents, generated by multiple publishers of the vehicle-related information, will be easier with timestamping. The "state" attribute indicates whether the document contains the full vehicle-info state, or only the information which has recently changed (partial).

The <vehicle-state> element consists of one or more child elements. These elements provide information on the vehicle internal state. Currently, these child elements may include:

The <status> element - describing whether the vehicle is accessible ("open") or not ("closed").

The <ignition> element - describing whether the engine is

running ("on/off").

The <fuel> and <temperature> elements - providing the data on fuel in the tank and the temperature. The temperature may be reported as the vehicle's inside temperature or engine temperature or other. The temperature information may be specified in more detailed form in the future. There is the "unit" attribute associated with these two elements, which MUST be present. It specifies the measurement unit.

The <passengers> element - describing how many passengers are in the car.

The <airbags> element - describing the status of the airbags ("open/closed").

The <vehicle-state> element MAY contain other elements and attributes from different namespaces for extensions; elements or attributes from unknown namespaces MUST be ignored.

The <vehicle-diagnostics> element consists of one or more child <obdii> elements described in detail in section 3.2 . There is a "DTC" attribute associated with the element. The attribute contains a five-byte Diagnostic Trouble Code (DTC) from the Powertrain Control Module. The element contains a short text related to a specific DTC. Alternatively, the <obdii> element may have a pair of the "RTData" and the "unit" attributes, which provide the description of the monitored real-time activity and the measurement unit.

The <vehicle-diagnostics> element MAY contain other elements and attributes from different namespaces for extensions; elements or attributes from unknown namespaces MUST be ignored.

5.2. XML Schema

The following is the schema definition of the vehicle-info format:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:vehicle-info"
   xmlns:car="urn:ietf:params:xml:ns:vehicle-info"
```

```
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

   <xs:include schemaLocation="vehicle-info-schema.xsd"/>

   <xs:element name="vehicle-info" type="car:vehicle-info"/>
     <xs:annotation>
        <xs:documentation>
```



```
          Root element for vehicle-info package.
        </xs:documentation>
     </xs:annotation>

     <xs:complexType name="vehicle-info">
       <xs:sequence>
             <xs:element name="vehicle-state" minOccurs="0"
             maxOccurs="1"/>
         <xs:element name="vehicle-diagnostics" minOccurs="0"
             maxOccurs="1"/>
         <xs:any namespace="##other" processContents="lax" minOccurs="0"
             maxOccurs="unbounded"/>
       </xs:sequence>
       <xs:attribute name="entity" type="xs:anyURI" use="required"/>
       <xs:attribute name="state" type="xs:string" use="required"/>
       <xs:attribute name="version" type="xs:signedInt"
             use="required"/>
     </xs:complexType>
     <xs:complexType name="vehicle-state">
           <xs:sequence>
             <xs:element name="status" type="xs:string"
                         minOccurs="0" maxOccurs="1"/>
                   <xs:annotation>
                   <xs:documentation>
                   Vehicle is powered on (with engine running or not)
                   or not
                   </xs:documentation>
                   </xs:annotation>
                 <xs:element name="ignition" type="xs:string"
                 minOccurs="0" maxOccurs="1"/>
                   <xs:annotation>
                   <xs:documentation>Engine running or not.
```

```
                  </xs:documentation>
                  </xs:annotation>
                <xs:element name="fuel"  type="xs:Double"
                        minOccurs="0" maxOccurs="1"/>
                  <xs:annotation>
                  <xs:documentation>Fuel in the fuel tank.
                          </xs:documentation>
                  </xs:annotation>
              <xs:element name="temperature" type="xs:Integer"
                        minOccurs="0" maxOccurs="1"/>
                  <xs:annotation>
                  <xs:documentation>Temperature inside the
                   vehicle.
                  </xs:documentation>
                  </xs:annotation>
                <xs:element name="passengers"
```



```
                type="xs:nonNegativeInteger" minOccurs="0"
                maxOccurs="1"/>
                  <xs:annotation>
                  <xs:documentation>Number of passengers
                  (driver included).
                  </xs:documentation>
                  </xs:annotation>
              <xs:element name="airbags" type="xs:string"
                  minOccurs="0" maxOccurs="1"/>
                  <xs:annotation>
                  <xs:documentation>Airbags status.
                  </xs:documentation>
                  </xs:annotation>
          <xs:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
                  <xs:attribute name="unit" type="xs:string"/>
                  <xs:annotation>
                  <xs:documentation>Measurement unit.
                  </xs:documentation>
                  </xs:annotation>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="vehicle-diagnostics">
            <xs:sequence>
          <xs:element name="obdii" type="xs:string" minOccurs="0"
```

```
                                maxOccurs="unbounded"/>
                  <xs:annotation>
                  <xs:documentation>OBD-II diagnostic code
 description or value of the Real Time Data measurement.
                  </xs:documentation>
                  </xs:annotation>
            <xs:any namespace="##other" processContents="lax"
                        minOccurs="0" maxOccurs="unbounded"/>
          <xs:attribute name="DTC" type="xs:string"/>
                        <xs:annotation>
                        <xs:documentation>Diagnostic Trouble Code.
                        </xs:documentation>
                        </xs:annotation>
                        <xs:attribute name="RTData" type="xs:string"/>
                          <xs:annotation>
                          <xs:documentation>Real Time Data measurement.
                          </xs:documentation>
                          </xs:annotation>
                        <xs:attribute name="unit" type="xs:string"/>
                          <xs:annotation>
                            <xs:documentation>Measurement unit.
                            </xs:documentation>
                          </xs:annotation>
```



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

 </xs:schema>
```

Figure 2: XML schema

6. Example of vehicle-info XML

As mentioned earlier, the vehicle-info XML-based data should be treated as a data structure, open for contributions from vehicle telematics industry and standards bodies. An example is given below:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<vehicle-info xmlns="urn:ietf:params:xml:ns:vehicle-info"
    entity="sip:44G44444H4444@avis.com"
    state="full"
    version="1111111111" >
  <vehicle-state>
    <status>open</status>
    <ignition>on</ignition>
    <fuel unit="gallon">3.0</fuel>
    <temperature unit="F">68</temperature>
    <passengers>3</passengers>
    <airbags>closed</airbags>
  </vehicle-state>
  <vehicle-diagnostics>
    <obdii DTC="P0120">Throttle Switch Malfunction</obdii>
    <obdii DTC="P1390">Timing Belt Skipped a Tooth</obdii>
    <obdii RTData="EngineCoolantTemp" unit="F">20</obdii>
    <obdii RTData="VehicleSpeed" unit="Miles">55</obdii>
    <obdii RTData="EngineRPM" unit="RPM">3257</obdii>
  </vehicle-diagnostics>
</vehicle-info>
```

Figure 3: XML example

7. Security Considerations

7.1. Authorization Considerations

There are obvious similarities between this section and the Security Considerations section of the membership event package document [15].

The group vehicle-info data contains privacy sensitive information as it can be used to deduce more detailed presence information of the user from multiple event packages of different entities. Consequently, access to the vehicle-info data and to the extended presence information MUST be controlled and be unavailable to unauthorized entities.

A vehicle management company may be authorized to obtain the vehicle information using vehicle-info event package. A vehicle management server may allow the vehicle-info data to be passed to a user presentity only if the user is inside ths vehicle.

In the car rental scenario example, apart from car rental company, only the presentity associated with the car is authorized by the car rental company to get vehicle-info data for the car. The same applies to the vehicle location data. The presentity may have this data aggregated into its extended presence information.

In many cases, other users may get the vehicle-info data indirectly. Watchers, who want to get presentity's extended presence information can obtain it by subscribing to presentity using the presence event package. The PA would send presence information based on presentity's privacy preferences as described in the common policy specification [19].

8. IANA Considerations

A future version of this document will provide IANA considerations.

9. Acknowledgements

10. References

10.1. Normative References

[1] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

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

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

[4] "Geographic information - Geography Markup Language (GML),

OpenGIS 03-105r1, available at:
http://portal.opengeospatial.org/files/?artifact_id=4700",
April 2004.

[5] "On Board Diagnostic (OBD), available at: http://obdii.com",
April 2004.

[6] "Extensible Markup Language (XML) 1.0 (Second Edition), World
Wide Web Consortium FirstEdition REC-xml-20001006, available
at: http://www.w3.org/TR/2000/REC-xml-20001006", October 2000.

[7] Moats, R., "URN Syntax", RFC 2141, May 1997.

[8] Moats, R., "A URN Namespace for IETF Documents", RFC 2648,
August 1999.

[9] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
January 2004.

10.2. Informative References

[10] Garcia-Martin, M. and H. Schulzrinne, "Notification of General
Events Using the Session Initiation Protocol (SIP) Event
Notification Framework", draft-garcia-sipping-general-events-00
(work in progress), May 2007.

[11] "Intelligent Transportation Society of America, available at:
http://www.itsa.org/", April 2004.

[12] "Global System for Telematics Forum, at:
http://www.gstforum.org/en/home.htm", April 2004.

[13] Singh, V., "Dynamic Feature Extensions to the Presence
Information Data Format Location Object (PIDF-LO)",
draft-singh-geopriv-pidf-lo-dynamic-01 (work in progress),
March 2007.

[14] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and
J. Peterson, "Presence Information Data Format (PIDF)",
RFC 3863, August 2004.

[15] Singh, V., "Membership Event Package",
draft-singh-simple-membership-00 (work in progress), July 2007.

[16] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[17] Khartabil, H., Leppanen, E., Lonnfors, M., and J. Costa-Requena, "Functional Description of Event Notification Filtering", RFC 4660, September 2006.

[18] Khartabil, H., Leppanen, E., Lonnfors, M., and J. Costa-Requena, "An Extensible Markup Language (XML)-Based Format for Event Notification Filtering", RFC 4661, September 2006.

[19] Schulzrinne, H., Tschofenig, H., Morris, J., Cuellar, J., Polk, J., and J. Rosenberg, "Common Policy: A Document Format for Expressing Privacy Preferences", RFC 4745, February 2007.

[20] Mahy, R., "A Document Format for Filtering and Reporting Location Notications in the Presence Information Document Format Location Object (PIDF-LO)", draft-ietf-geopriv-loc-filters-01 (work in progress), March 2007.

[21] Polk, J., Schnizlein, J., and M. Linsner, "Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information", RFC 3825, July 2004.

[22] Schulzrinne, H., "Timed Presence Extensions to the Presence Information Data Format (PIDF) to Indicate Status Information for Past and Future Time Intervals", RFC 4481, July 2006.

[23] Cuellar, J., Morris, J., Mulligan, D., Peterson, J., and J. Polk, "Geopriv Requirements", RFC 3693, February 2004.

[24] Polk, J. and B. Rosen, "Session Initiation Protocol Location Conveyance", draft-ietf-sip-location-conveyance-07 (work in progress), February 2007.

[25] Thomson, M., "Geodetic Shapes for the Representation of Uncertainty in PIDF-LO", draft-thomson-geopriv-geo-shape-03 (work in progress), December 2006.

[26] Schulzrinne, H., "Geolocation Policy: A Document Format for Expressing Privacy Preferences for Location Information", draft-ietf-geopriv-policy-12 (work in progress), May 2007.

Authors' Addresses

Vishal Singh
Columbia University
Department of Computer Science
450 Computer Science Building
New York, NY 10027
US

Email: vs2140@cs.columbia.edu
URI: http://www.cs.columbia.edu/~vs2140

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

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

Piotr Boni
Verizon Communications
40 Sylvan Rd
Waltham, MA 02451
US

Phone: +1 781 466 2903
Email: piotr.boni@verizon.com

Full Copyright Statement

Intellectual Property

http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).