Dispatch Working Group
Internet-Draft
Intended status: Informational
Expires: August 21, 2015

A. Allen, Ed.
Blackberry

February 17, 2015

Indicating that out of dialog requests related to an existing dialog must be routed via an intermediary
draft-allen-dispatch-routing-out-of-dialog-request-01

Abstract

This document discusses the problems for out of dialog requests that are related to another dialog, caused by B2BUA intermediaries that modify SIP parameters or terminate dialogs and proposes some possible solutions.

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 August 21, 2015.

Table of Contents

1. Introduction

In RFC 3265 [1] and RFC 3515 [4] SUBCRIBE requests and REFER requests were allowed to reuse a dialog created by another SIP method (e.g. INVITE). RFC 6665 [3] has deprecated such dialog reuse due to all the problems that dialog reuse caused. However some B2BUA intermediaries change parameters in SIP requests or terminate dialogs and need to receive the SUBSCRIBE and REFER requests that relate to an existing dialog that is record routed via the B2BUA. While draft-ietf-sipcore-refer-explicit-subscription [6] defines a means for the sending of REFER requests to ensure that no subscription is created by the REFER recipient and thus it is safe to send the REFER request on a existing dialog the cases where notifications are needed still require the SUBSCRIBE and REFER request to be sent on a new dialog.

2. Problem statement

SIP sessions often involve intermediaries acting as B2BUAs that in addition to forwarding SIP requests and responses also act as UAs to perform more complex manipulations of the session. Such manipulations include modifying URIs in the Request-URI, Contact address or other header fields and even terminating the dialog for some mid session requests (for example performing a session transfer when receiving a REFER request rather than forwarding the REFER request to the remote UAS).

Typically such functionality has been achieved by sending REFER and SUBSCRIBE requests within the established dialog for the session, with the intermediary then intercepting the REFER or SUBSCRIBE request and then either modifying to conform to the expected view of the remote UAS or processing the REFER or SUBSCRIBE request rather

than forwarding it to the remote UAS. However such dialog reuse has been problematic and RFC 6665 [3] has deprecated dialog reuse (except for legacy interoperability).

However, if REFER and SUBSCRIBE requests are sent outside the related existing dialog then the requests will not be routed by the manipulating B2BUA and thus will either to fail to arrive at the remote UA due to URI manipulations or fail at the remote UA because parameters in the request (e.g Target-Dialog, Replaces, Refer-To URI, etc) do not match the values at the remote UAS. draft-kaplan-dispatch-gruu-problematic-00 [7] further describes some of the problems if a GRUU is used as the Request-URI of a related out of dialog request.

One way B2BUAs have have addressed this problem is by acting as two UAs back-to-back with the Contact URI being overwritten to be the URI of the B2BUA. However this means that GRUU of the UA is overwritten by the B2BUA and the meaning of the Contact header field parameters becomes obscure. Do the Contact header field parameters reflect the capabilities of the Contact address (i.e the B2BUA) or do they reflect the capabilities of the remote UA? If they relect the capabilities of the B2BUA then the identfication of the capabilities of the remote UAS has been lost. If they reflect the capabilities of the remote UA then they falsely identify that the B2BUA contact address has the capabilities of the remote UA. While some have advocated that a B2BUA should only indicate the capabilities that it understands and supports in the Contact, in the opinion of the author this is not desirable behavior because the feature tags may indicate many kinds of capabilities which do not require the support of the intermediary. For an intermediary only to indicate those capabilities that it understands and supports is a big barrier to UAs mutually exchanging feature capabilities. In the opinion of the author the feature capability indicator mechanism defined in RFC 6809 [2] is the appropriate means for an intermediary to indicate the capabilities that it supports and will allow. It also should be recognised that UAs may store Contact addresses especially if they are GRUUs for use later for originating sessions (e.g. stored in the address book) or for filtering incoming sessions (e.g. incoming sessions addressed to temporary GRUUs). So if the Contact address is overwritten then this information is lost or not valid as a contact outside the lifetime of the current dialog. Additionally the mechanism defined in RFC 6665 [3] depends on the UA receiving a GRUU as the remote target in order to avoid dialog reuse, so overwriting the Contact Address breaks this mechanism.

What is needed is a way for intermediaries that need to receive and manipulate or process mid session requests to indicate that mid session out of dialog requests that relate to the dialog of the

session being established, to indicate a URI to be included in the Route header of such out of dialog requests so that the request will route by the intermediary.

3. Potential solutions

3.1. New SIP header field

A new SIP header field (e.g. OOD-Record-Route) could be defined which contains the URI of the intermediary for routing out of dialog requests that relate to another dialog. The intermediary would include the new header field containing the URI that the intermediary requires related out of dialog requests to be routed to in the requests and responses at dialog establishment. The UA would then include a Route header field containing the URI received in the new header field in any related out of dialog requests it sends.

3.2. New rr-param

A new rr-param(e.g. OOD-RR) could be defined which indicates that this is the URI of the intermediary for routing out of dialog requests that relate to another dialog. The intermediary would include the new rr-param when including its URI in the Record-Route header field. The UA would then include a Route header field containing the URI with the associated new rr-param received in the Record-Route header field in any related out of dialog requests it sends.

3.3. New URI parameter

A new URI parameter (e.g. OOD-RR) could be defined which indicates that this is the URI of the intermediary for routing out of dialog requests that relate to another dialog. The intermediary would include the new URI parameter when including its URI in the Record-Route header field. The UA would then include a Route header field containing the URI with the new URI parameter received in the Record-Route header field in any related out of dialog requests it sends.

3.4. New Feature Capability Indicator

RFC 6809 [2] defines the Feature-Caps header field for intermediaries to include Feature-Capability indicators indicating the capabilities they support. A new feature-capability indicator (e.g. sip.ood-route) could be defined which contains the URI of the intermediary for routing out of dialog requests that relate to another dialog. The intermediary would include a Feature-Caps header field containing the Feature-Capability indicator with the URI that the intermediary requires related out of dialog requests to be routed to in the

requests and responses at dialog establishment. The UA would then include a Route header field containing the URI received in the Feature-Capability indicator in any related out of dialog requests it sends.

3.5. Embed Route header fields in the contact URI

The Contact URI can contain embedded header fields (see RFC 3261 [5]). The intermediary could embed a Route header field containing its own URI in the Contact URI. One advantage of this approach is that there may be some backward compatibility with this mechanism because RFC 3261 [5] compliant UAs should use the embedded Route header fields when constructing a request addressed to this Contact URI. However it is questionable as to how many implemntations actually will do this in practice. A disadvantage of this approach is if the Contact URI is secured using SMIME or a similar means for detecting man in the middle attaks on the Contact address then tampering with the URI could lead to the UAS believing that the Contact URI has been maliciously tampered with.

3.6. Option Tag

A new SIP option tag will be needed for a UA to indicate that it supports the new extension so that the the intermediary can use the new mechanism instead of other approaches that modify the contact address and force dialog reuse. SIP OPTIONS method could be used by the intermediary to determine whether the UAS supports the extension before forwarding the dialog creating request. Alternatively the intermediary might modify the dialog after discovering in a response whether the UAS supports the new extension or not.

4. Security Considerations

The capability to include a URI in a request or response which will cause a UA to route other requests via the intermediary provides the possibility to create man-in-the-middle attacks. However this is also true of existing SIP header fields like Record-Route. The same considerations apply as those to the use of Record-Route header fields.

5. IANA Considerations

This document does not currently have anything requiring action by IANA.

6. Acknowledgements

The author would like to thank Hadrial Kaplan, Paul Kyzivat, Christer Holmberg and Dale Worley for the inital list discussion on the issues raised by this draft and additional suggestions on the solutions.

7. Informative References

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

[2] Holmberg, C., Sedlacek, I., and H. Kaplan, "Mechanism to Indicate Support of Features and Capabilities in the Session Initiation Protocol (SIP)", RFC 6809, November 2012.

[3] Roach, A., "SIP-Specific Event Notification", RFC 6665, July 2012.

[4] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

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

[6] Sparks, R., , "Explicit Subscriptions for the REFER Method", Internet Draft draft-ietf-sipcore-refer-explicit-subscription-00, November 2014.

[7] Kaplan, H., , "Problems with the SIP Globally Routable User Agent URIs (GRUUs)", Internet Draft draft-kaplan-dispatch-gruu-problematic-00, October 2010.

Author’s Address

Andrew Allen (editor)
Blackberry
1200 Sawgrass Corporate Parkway
Sunrise, Florida 33323
USA

Email: aallen@blackberry.com

Independent
Internet-Draft
Intended status: Informational
Expires: January 19, 2016

H. Butler
Hobu Inc.
M. Daly
Cadcorp
A. Doyle
MIT
S. Gillies
Mapbox Inc.
T. Schaub
Planet Labs
S. Hagen

July 18, 2015

The GeoJSON Format
draft-butler-geojson-06

Abstract

GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON). It defines several types of JSON objects and the manner in which they are combined to represent data about geographic features, their properties, and their spatial extents. This document recommends a single coordinate reference system based on WGS 84. Other coordinate reference systems are not recommended.

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 January 19, 2016.

Table of Contents

1. Introduction

GeoJSON is a format for encoding data about geographic features using JavaScript Object Notation (JSON) [RFC7159]. The format is concerned with features in the broadest sense; any thing with qualities that are bounded in geographical space may be a feature whether it is a physical structure or not. The concepts in GeoJSON are not new; they are derived from pre-existing open geographic information system standards (for COM, SQL, and XML) and have been streamlined to better suit web application development using JSON.

GeoJSON comprises the seven concrete geometry types defined in the OpenGIS Simple Features Implementation Specification for SQL [SFSQL]: 0-dimensional Point and MultiPoint; 1-dimensional curve LineString and MultiLineString; 2-dimensional surface Polygon and MultiPolygon; and the heterogeneous GeometryCollection. GeoJSON representations of instances of these geometry types are analogous to the well-known binary (WKB) and text (WKT) representations described in that same specification.

GeoJSON also comprises the types Feature and FeatureCollection. Feature objects in GeoJSON contain a geometry object with one of the above geometry types and additional properties. A FeatureCollection object contains an array of feature objects. This structure is analogous to that of the Web Feature Service (WFS) response to GetFeatures requests specified in [WFSv1] or to a KML Folder of Placemarks [KMLv2.2]. Some implementations of the WFS specification also provide GeoJSON formatted responses to GetFeature requests, but there is no particular service model or feature type ontology implied in the GeoJSON format specification.

Since its initial publication in 2008 [GJ2008], the GeoJSON format specification has steadily grown in popularity. It is widely used in JavaScript web mapping libraries, JSON-based document databases, and web APIs.

1.1. Requirements Language

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

1.2. Conventions Used in This Document

The ordering of the members of any JSON object defined in this document MUST be considered irrelevant, as specified by [RFC7159].

Some examples use the combination of a JavaScript single line comment (//) followed by an ellipsis (...) as placeholder notation for content deemed irrelevant by the authors. These placeholders must of course be deleted or otherwise replaced, before attempting to validate the corresponding JSON code example.

Whitespace is used in the examples inside this document to help illustrate the data structures, but is not required. Unquoted whitespace is not significant in JSON.

1.3. Specification of GeoJSON

This document updates the original GeoJSON format specification [GJ2008].

1.4. Definitions

- JavaScript Object Notation (JSON), and the terms object, name, value, array, number, true, false, and null are to be interpreted as defined in [RFC7159].

- Inside this document the term "geometry type" refers to the seven case-sensitive strings: "Point", "MultiPoint", "LineString", "MultiLineString", "Polygon", "MultiPolygon", and "GeometryCollection".

- As another shorthand notation, the term "GeoJSON types" refers to the nine case-sensitive strings "Feature", "FeatureCollection" and the geometry types listed above.

1.5. Example

A GeoJSON feature collection:

```
{
    "type": "FeatureCollection",
    "features": [{
        "type": "Feature",
        "geometry": {
            "type": "Point",
            "coordinates": [102.0, 0.5]
        },
        "properties": {
            "prop0": "value0"
        }
    }, {
        "type": "Feature",
        "geometry": {
            "type": "LineString",
            "coordinates": [
                [102.0, 0.0],
                [103.0, 1.0],
                [104.0, 0.0],
                [105.0, 1.0]
            ]
        },
        "properties": {
            "prop0": "value0",
            "prop1": 0.0
        }
    }, {
        "type": "Feature",
        "geometry": {
            "type": "Polygon",
            "coordinates": [
                [
                    [100.0, 0.0],
                    [101.0, 0.0],
                    [101.0, 1.0],
                    [100.0, 1.0],
                    [100.0, 0.0]
                ]
            ]
        },
        "properties": {
            "prop0": "value0",
            "prop1": {
                "this": "that"
            }
        }
    }]
}
```

2. GeoJSON Object

GeoJSON always consists of a single object. This object (referred to as the GeoJSON object below) represents a geometry, feature, or collection of features.

- The top level of GeoJSON text MUST be a JSON object.
- The GeoJSON object MUST have a member with the name "type". The value of the member MUST be one of the GeoJSON types.
- A GeoJSON object MAY have a "bbox" member, the value of which MUST be a bounding box array (see 4. Bounding Boxes).
- The GeoJSON object MAY have any number of other members. Implementations MUST ignore unrecognized members.

2.1. Geometry Object

A geometry object is a GeoJSON object where the "type" value is one of the geometry types. A GeoJSON geometry object of any type other than "GeometryCollection" MUST have a member with the name "coordinates". The value of the coordinates member is always an array. The structure of the elements in this array is determined by the type of geometry. GeoJSON processors MAY interpret geometry objects with empty coordinates arrays as null objects.

2.1.1. Position

A position is the fundamental geometry construct. The "coordinates" member of a geometry object is composed of either:

- one position (in the case of a Point geometry),
- an array of positions (LineString or MultiPoint geometries),
- an array of arrays of positions (Polygons, MultiLineStrings),
- or a multidimensional array of positions (MultiPolygon).

A position is represented by an array of numbers. There MUST be two or more elements. The first two elements will be longitude and latitude, or easting and northing, precisely in that order and using decimal numbers. Altitude or elevation MAY be included as an optional third element.

Additional position elements MAY be included but MUST follow the three specified above and MAY be ignored by software. Interpretation

and meaning of additional elements is beyond the scope of this specification.

Examples of positions and geometries are provided in "Appendix A. Geometry Examples".

2.1.2. Point

For type "Point", the "coordinates" member MUST be a single position.

2.1.3. MultiPoint

For type "MultiPoint", the "coordinates" member MUST be an array of positions.

2.1.4. LineString

For type "LineString", the "coordinates" member MUST be an array of two or more positions.

2.1.5. MultiLineString

For type "MultiLineString", the "coordinates" member MUST be an array of LineString coordinate arrays.

2.1.6. Polygon

To specify a constraint specific to polygons, it is useful to introduce the concept of a linear ring:

- A linear ring is a closed LineString with 4 or more positions.
- The first and last positions are equivalent (they represent equivalent points).
- A linear ring is the boundary of a surface or the boundary of a hole in a surface.
- A linear ring SHOULD follow right-hand rule with respect to the area it bounds (ie. exterior rings are counter-clockwise, holes are clockwise)

Though a linear ring is not explicitly represented as a GeoJSON geometry type, it leads to a canonical formulation of the Polygon geometry type definition as follows:

- For type "Polygon", the "coordinates" member MUST be an array of linear ring coordinate arrays.

- For Polygons with more than one of these rings, the first MUST be the exterior ring and any others MUST be interior rings. The exterior ring bounds the surface and the interiors rings (if present) bound holes within the surface.

2.1.7. MultiPolygon

For type "MultiPolygon", the "coordinates" member MUST be an array of Polygon coordinate arrays.

2.1.8. Geometry Collection

A GeoJSON object with type "GeometryCollection" is a geometry object which represents a collection of geometry objects. A geometry collection MUST have a member with the name "geometries". The value corresponding to "geometries" is an array. Each element in this array is a GeoJSON geometry object.

2.2. Feature Object

A GeoJSON object with the type "Feature" is a feature object.

- A feature object MUST have a member with the name "geometry". The value of the geometry member SHALL be either a geometry object as defined above or, in the the case that the feature is unlocated, a JSON null value.

- A feature object MUST have a member with the name "properties". The value of the properties member is an object (any JSON object or a JSON null value).

- If a feature has a commonly used identifier, that identifier SHOULD be included as a member of the feature object with the name "id" and the value of this member is either a JSON string or number.

2.3. Feature Collection Object

A GeoJSON object with the type "FeatureCollection" is a feature collection object. An object of type "FeatureCollection" MUST have a member with the name "features". The value corresponding to "features" is an array. Each element in the array is a feature object as defined above.

3. Coordinate Reference System

The default reference system for all GeoJSON coordinates SHALL be a geographic coordinate reference system, using the [WGS84] datum, and with longitude and latitude units of decimal degrees. This coordinate reference system is equivalent to the OGC's "http://www.opengis.net/def/crs/OGC/1.3/CRS84" [OGCURL]. An OPTIONAL third position element SHALL be the height in meters above the WGS 84 reference ellipsoid. For widest interoperability, GeoJSON data SHOULD use this default coordinate reference system.

Other coordinate reference systems, including ones described by CRS objects of the kind defined in [GJ2008] are NOT RECOMMENDED. GeoJSON processing software SHALL NOT be expected to have access to coordinate reference systems databases. Applications requiring CRS other than the default MUST assume all responsibility for reference system and coordinate accuracy. Furthermore, GeoJSON coordinates MUST NOT under any circumstances use latitude, longitude order. See Section 6, Interoperability Considerations, for guidance in processing GeoJSON documents that do contain such a CRS object.

4. Bounding Box

A GeoJSON object MAY have a member named "bbox" to include information on the coordinate range for its geometries, features, or feature collections. The value of the bbox member MUST be an array of length 2*n where n is the number of dimensions represented in the contained geometries, with all axes of the most south-westerly point followed by all axes of the more north-easterly point. The axes order of a bbox follows the axes order of geometries.

Example of a bbox member on a feature:

```
{
    "type": "Feature",
    "bbox": [-180.0, -90.0, 180.0, 90.0],
    "geometry": {
        "type": "Polygon",
        "coordinates": [
            [
                [-180.0, 10.0],
                [20.0, 90.0],
                [180.0, -5.0],
                [-30.0, -90.0]
            ]
        ]
    }
    //...
}
```

Example of a bbox member on a feature collection:

```
{
    "type": "FeatureCollection",
    "bbox": [100.0, 0.0, 105.0, 1.0],
    "features": [
    //...
    ]
}
```

Example of a bbox for line crossing the date-line:

```
{
    "type": "Feature",
    "bbox": [170, 10, -170, 11],
    "geometry": {
        "type": "LineString",
        "coordinates": [
            [-170, 10],
            [170, 11]
        ]
    }
    //...
}
```

5. Security Considerations

GeoJSON shares security issues common to all JSON content types. See [RFC7159] Section 12 for additional information. GeoJSON does not provide executable content.

As with other geographic data formats, e.g., [KMLv2.2], providing details about the locations of sensitive persons, animals, habitats, and facilities can expose them to unauthorized tracking or injury. GeoJSON does not provide privacy or integrity services; if sensitive data requires privacy or integrity protection the service must be provided externally.

6. Interoperability Considerations

6.1. I-JSON

GeoJSON texts SHOULD follow the constraints of I-JSON [RFC7493] for maximum interoperability.

6.2. Coordinate Precision

The size of a GeoJSON text in bytes is a major interoperability consideration and precision of coordinate values has a large impact on the size of texts. A GeoJSON text containing many detailed polygons can be inflated almost by a factor of two by increasing coordinate precision from 6 to 15 decimal places. For geographic coordinates with units of degrees, 6 decimal places (a default common in, e.g., sprintf) amounts to about 10 centimeters, a precision well within that of current GPS systems. Implementations should consider the cost to using a greater precision than necessary.

6.3. Coordinate Order

There are conflicting precedents among geographic data formats over whether latitude or longitude come first in a pair of numbers. Longitude comes first in GeoJSON coordinates as it does in [KMLv2.2].

Some commonly-used CRS definitions specify coordinate ordering that is not longitude then latitude (for a geographic CRS) or easting then northing (for a projected CRS). The CRS historically known as "EPSG:4326" and more accurately named "http://www.opengis.net/def/crs/EPSG/0/4326" is a prime example. Using such a CRS is NOT RECOMMENDED due to the potential disruption of interoperability. When such a CRS is encountered in GeoJSON, the document should be processed with caution. Heuristics may be necessary to interpret the coordinates properly; they may not be in the required longitude, latitude order.

6.4. Coordinate Reference System Identifiers

Earlier versions of the GeoJSON specification recommended use of OGC URNs such as "urn:ogc:def:crs:OGC:1.3:CRS84" to name a CRS. This version deprecates the URNs and recommends a change to HTTP URLs

[Section 3.1]. Widely deployed systems using, e.g. the GDAL and OGR libraries, currently write the deprecated OGC URNs into GeoJSON documents and will do so until replaced by newer versions. GeoJSON processors should be prepared for either form.

6.5. Bounding boxes

In representing features that cross the dateline or the poles, following the ring-orientation best practice (counter-clockwise external rings, clockwise internal rings) and ensuring your bounding boxes use the south-west corner as the first coordinate will improve interoperability. Remain aware that software that represents edges as straight cartesian lines and software that represents edges as great circles will have different interpretations of edges, which vary more the longer the edges are. Try to avoid edges of more than 180 degrees in length as far as possible.

7. IANA Considerations

The MIME media type for GeoJSON text is application/vnd.geo+json.

Type name: application

Subtype name: vnd.geo+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: binary

Security considerations: See section 5 above

Interoperability considerations: See section 6 above

Published specification: draft-butler-geojson

Applications that use this media type: various

Additional information:

Magic number(s) : n/a

File extension(s) : .json, .geojson

Macintosh file type code : TEXT

Object Identifiers: n/a

Person to contact for further information:

Sean Gillies

sean.gillies@gmail.com

Intended usage: COMMON

Restrictions on usage: none

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <http://www.rfc-editor.org/info/rfc7493>.

8.2. Informative References

[GJ2008] Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., and C. Schmidt, "The GeoJSON Format Specification", June 2008.

[KMLv2.2] Wilson, T., "OGC KML", OGC 07-147r2, April 2008.

[OGCURL] Cox, S., "OGC-NA Name type specification - definitions: Part 1 - basic name", OGC 09-048r3, March 2010.

[SFSQL] OpenGIS Consortium, Inc., "OpenGIS Simple Features Specification For SQL Revision 1.1", OGC 99-049, May 1999.

[WFSv1] Vretanos, P., "Web Feature Service Implementation Specification", OGC 02-058, May 2002.

[WGS84] National Imagery and Mapping Agency, "Department of Defense World Geodetic System 1984, Third Edition", 1984.

Appendix A. Geometry Examples

Each of the examples below represents a valid and complete GeoJSON object.

A.1. Points

Point coordinates are in x, y order (easting, northing for projected coordinates, longitude, latitude for geographic coordinates):

```
{
    "type": "Point",
    "coordinates": [100.0, 0.0]
}
```

A.2. LineStrings

Coordinates of LineString are an array of positions (see "2.1.1. Position"):

```
{
    "type": "LineString",
    "coordinates": [
        [100.0, 0.0],
        [101.0, 1.0]
    ]
}
```

A.3. Polygons

Coordinates of a Polygon are an array of LinearRing (cf. "2.1.6 Polygon") coordinate arrays. The first element in the array represents the exterior ring. Any subsequent elements represent interior rings (or holes).

No holes:

```
{
    "type": "Polygon",
    "coordinates": [
        [
            [100.0, 0.0],
            [101.0, 0.0],
            [101.0, 1.0],
            [100.0, 1.0],
            [100.0, 0.0]
        ]
    ]
}
```

With holes:

```
{
    "type": "Polygon",
    "coordinates": [
        [
            [100.0, 0.0],
            [101.0, 0.0],
            [101.0, 1.0],
            [100.0, 1.0],
            [100.0, 0.0]
        ],
        [
            [100.8, 0.8],
            [100.8, 0.2],
            [100.2, 0.2],
            [100.2, 0.8],
            [100.8, 0.8]
        ]
    ]
}
```

With hole crossing dateline:

```
{
    "type": "Polygon",
    "coordinates": [
        [
            [-170.0, 10.0],
            [170.0, 10.0],
            [170.0, -10.0],
            [-170.0, -10.0],
            [-170.0, 10.0]
        ],
        [
            [175.0, 5.0],
            [-175.0, 5.0],
            [-175.0, -5.0],
            [175.0, -5.0],
            [175.0, 5.0]
        ]
    ]
}
```

A.4. MultiPoints

Coordinates of a MultiPoint are an array of positions::

```
{
    "type": "MultiPoint",
    "coordinates": [
        [100.0, 0.0],
        [101.0, 1.0]
    ]
}
```

A.5. MultiLineStrings

Coordinates of a MultiLineString are an array of LineString coordinate arrays:

```
{
    "type": "MultiLineString",
    "coordinates": [
        [
            [100.0, 0.0],
            [101.0, 1.0]
        ],
        [
            [102.0, 2.0],
            [103.0, 3.0]
        ]
    ]
}
```

A.6. MultiPolygons

Coordinates of a MultiPolygon are an array of Polygon coordinate arrays:

```
{
    "type": "MultiPolygon",
    "coordinates": [
        [
            [
                [102.0, 2.0],
                [103.0, 2.0],
                [103.0, 3.0],
                [102.0, 3.0],
                [102.0, 2.0]
            ]
        ],
        [
            [
                [100.0, 0.0],
                [101.0, 0.0],
                [101.0, 1.0],
                [100.0, 1.0],
                [100.0, 0.0]
            ],
            [
                [100.2, 0.2],
                [100.8, 0.2],
                [100.8, 0.8],
                [100.2, 0.8],
                [100.2, 0.2]
            ]
        ]
    ]
}
```

A.7. GeometryCollections

Each element in the geometries array of a GeometryCollection is one of the geometry objects described above:

```
{
    "type": "GeometryCollection",
    "geometries": [{
        "type": "Point",
        "coordinates": [100.0, 0.0]
    }, {
        "type": "LineString",
        "coordinates": [
            [101.0, 0.0],
            [102.0, 1.0]
        ]
    }]
}
```

Appendix B. Contributors

The GeoJSON format is the product of discussion on the GeoJSON mailing list: http://lists.geojson.org/listinfo.cgi/geojson-geojson.org.

Comments are solicited and should be addressed to the GeoJSON mailing list at geojson@lists.geojson.org or to the GeoJSON issue tracker at https://github.com/geojson/draft-geojson/issues.

Authors' Addresses

H. Butler
Hobu Inc.

M. Daly
Cadcorp

A. Doyle
MIT

S. Gillies
Mapbox Inc.

Email: sean.gillies@gmail.com
URI: http://sgillies.net

T. Schaub
Planet Labs

S. Hagen
Rheinaustr. 62
Bonn 53225
DE

Email: stefan@hagen.link
URI: http://stefan-hagen.website/

IETF
Internet-Draft
Intended status: Informational
Expires: July 24, 2015

P. Kyzivat
US VRS Providers
January 20, 2015

Telecommunications Relay Service Purpose for the Call-Info Header Field in the Session Initiation Protocol (SIP)
draft-kyzivat-dispatch-trs-call-info-purpose-01

Abstract

This document defines and registers a value of "original-identity" for the "purpose" header field parameter of the Call-Info header field in the Session Initiation Protocol (SIP).

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 July 24, 2015.

Copyright Notice

Table of Contents

1. Introduction

The United States Federal Communications Commission (FCC) sponsors specialized Telecommunications Relay Services (TRS) for US residents who are deaf, deaf-blind, and hard-of-hearing. These "relay" services translate audio telephone calls to/from any user of the public telephone system into other modalities (such as American Sign Language) that can be used by these TRS users. These services are being updated to employ SIP signaling.

TRS services are supplied by a number of private TRS service providers. Individual TRS users enroll with one provider, and the providers peer with one another and with public telecommunications providers to provide connectivity equivalent with the basic telephone network. Users enrolled with one provider are legally entitled to request and receive relay services from any of the other providers on a call-by-call basis. In that case the default provider acts as an intermediary between the TRS user's device and the provider selected by the user to provide the service for the call.

Telephone users who call a TRS user are by default routed to the default provider for that user, and provided TRS relay service by the default provider. However, callers are also entitled to call a TRS provider of their choice and request connection to a TRS user enrolled with a different provider. In that case the TRS provider called by the user provides the relay service and the default provider acts as an intermediary.

For a provider to receive reimbursement for providing relay service on a call the FCC requires that the provider supply call detail including the IP address of the device the TRS user is using for the call. When the service provider that is rendering the service is not the one enrolling the user this information may not be available naturally in the SIP signaling.

In some simple cases the IP address of the deaf user could be present in the signaling as the Contact URI of the caller or callee. However this is often not case due to the presence of intermediate devices

acting as B2BUAs. B2BUAs often modify the Contact URI, because they need the downstream entity to contact the B2BUA rather than the actual contact. There are relay services that actually need the "real" contact URI, and TRS is one of them.

This document identifies requirements for a new mechanism, not currently available in SIP, for the described environment to function, and defines a SIP mechanism to meet those requirements.

2. Requirements

[TRS-1] The mechanism must support carrying the IP address of the source of a request in an INVITE request.

[TRS-2] The mechanism must support carrying the IP address of the target of an INVITE request in the responses to that request.

[TRS-3] The mechanism must work in the presence of B2BUA intermediaries in the signaling path.

[TRS-4] It must be possible for intermediaries to insert the IP address on behalf of the source or recipient.

[TRS-5] It must be possible for intermediaries to remove or obfuscate the IP address to enforce trust and privacy policies.

[TRS-6] The mechanism must support topologies where the source and/or target use a signaling protocol other than SIP (e.g. H.323) and an intermediary converts the signaling to/from SIP.

3. Mechanism

To provide a mechanism that supplies the needed information in common deployments, this document calls for using the SIP Call-Info header field to carry a URI containing the IP address of the deaf user. To distinguish this use of Call-Info from other uses, a new ’purpose’ parameter value ("original-identity") is used. For example:

```
Call-Info: <sip:10.1.1.1>; purpose=original-identity
```

A Call-Info with this purpose can be inserted by a server in requests and responses in the following cases:

- in requests sent toward a peer TRS server when this server knows the the device originating the request is an enrolled TRS device;

- o in responses sent toward a peer TRS server when this server knows that the target of the corresponding request is an enrolled TRS device.

4. Security Considerations

The security considerations for the extension defined herein are comparable to those for the Contact-URI. The security considerations of [RFC3261] apply to this extension and deal with disclosure of this information to entities that are not in the signaling path for the call.

In the case where the caller or callee wishes to withhold its identity from the other UA in the call, the considerations of [RFC3323] can be employed. Because 'original-identity' is used to enable an intermediary to provide service, user-provided-privacy is not an option, and network-provided-privacy is the only option. TRS service providers MUST act as a privacy service and remove or anonymize the URI in a Call-Info with purpose 'original-identity' when providing 'header' privacy.

[NOTE: I would prefer to avoid having to define specific extensions to RFC3323 for this, and complex normative requirements. But I'm not sure it is possible to avoid doing so.]

5. IANA Considerations

This document defines and registers a new predefined value "original-identity" for the "purpose" header field parameter of the Call-Info header field (defined in [RFC3261]), following the procedures specified in [RFC3968].

IANA is requested to revise the existing entry for the Call-Info header Field in the "Header Field Parameters and Parameter Values" table of the "Session Initiation Protool (SIP) Parameters" registry, by adding a reference to this document.

6. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3323] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", RFC 3323, November 2002.

[RFC3968] Camarillo, G., "The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", BCP 98, RFC 3968, December 2004.

Author's Address

Paul Kyzivat
US VRS Providers
Hudson, MA
US

Email: pkyzivat@alum.mit.edu