Network Working Group A. Mayrhofer
Internet-Draft enum.at
Intended status: Standards Track C. Spanring
Expires: August 18, 2008 OIR-ID
Feb 15, 2008

A Uniform Resource Identifier for Geographic Areas ('geo' URI)
draft-mayrhofer-geo-uri-02

Status of this Memo

This Internet-Draft will expire on August 18, 2008.

Copyright Notice

Abstract

This document specifies an Uniform Resource Identifier (URI) for geographic areas using the 'geo' scheme name. A 'geo' URI provides a compact, human-readable, and protocol independent way to identify an area by means of a hierarchical tiling scheme.

Table of Contents

1. Change Log

[Note to editors: This section is to be removed before publication - XML source available on request]

draft-mayrhofer-geo-uri-02
completely new way to create URI - tiling scheme, base32 encoding, etc.

draft-mayrhofer-geo-uri-01
removed parameters

draft-mayrhofer-geo-uri-00
initial draft

2. Introduction

The use of spatial location in internet technology has gained significant importance over the last few years. More and more protocols and data formats are being extended to carry spatial information, most prominently geographic location.

Most of those specifications are optimized for machine readability, capable of carrying complex location information (and are therefore complex by themselves), or are tied to a specific protocol or data format. On the contrary, the success of domain names, URIs and telephone numbers indicates that there is a demand for conveying simple, concise identity information by a variety of "manual" means between humans.

This document aims at specifying such a simple, concise identifier for geographic location using a hierarchical tiling scheme. It is intended to coexist with and support existing "machine-readable" location information schemes.

According to [RFC3986], a Uniform Resource Identifier (URI) "is a compact sequence of characters that identifies an abstract or physical resource". The URI scheme specified in this document (using the 'geo' scheme name) identifies such a physical resource (a geographic area) in a compact, human-readable, and protocol idependent way.

The URI scheme specified in this document uses only geographic coordinates - the provision of civic addresses to identify locations is out of scope of this document.

3. 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].

In addition this document uses the following terminology:

- area bitstring: A string of bits, describing a geographic tile according to Section 6.
- tile identifier: A base32 encoded string containing an area bitstring, padding and parity information, as described in Section 7.

4. Requirements

(Note: Earlier versions of this document proposed URIs in the form of "geo:<latitude>,<longitude>". Feedback from the GEOPRIV working group indicated that this format does not seem to fulfill some of the requirements mentioned during the presentation. Requirements are therefore clarified in this section).

The IETF has already produced several specifications in the area of encoding and conveying geographical location (RFC 4119, RFC 3825, RFC 4776). However, none of those schemes specifically focuses on conveying geographic location identification among humans or between humans and machines (for example, for user input into a device).

Identifiers handled by humans are considered to have the following requirements:

- Interopability: Good identifiers are not limited to an administrative domain, but are universally usable and recognized. They have a clear public specification that allows anybody to create, encode, decode and dereference identifier instances, and they are easily distinguished from other identifier spaces. URI schemes are an example of such an identifier scheme.
- Length: Good identifiers are as short as possible. Such identifiers require less time to read, write and spell them. The probability of "transmission errors" such as mistyped, missing, or flipped characters is proportional to the length of an identifier.
- Character set: Good identifiers use a limited character set, which reduces the risk of confusing two similar characters. A good character set is also globally recognized.
- Ambigiuity: Good identifiers have little ambigiuity.
- Recognition: Good identifiers allow recognizing rough "similarity" among a set of identifiers (for example, domain names below the same top level domain, phone numbers within the same area or network, email addresses within the same domain name). Such

identifiers allow a quick "categorization" even by humans.

- Error detection: Good identifiers provide basic error detection, for example a mistyped identifier can at least be recognized as invalid.

5. IANA Registration of 'geo' URI Scheme

This section contains the fields required for the URI scheme registration, following the guidelines in section 5.4 of [RFC4395].

5.1. URI Scheme Name

geo

5.2. Status

permanent

5.3. URI Scheme Syntax

The syntax of the 'geo' URI scheme is specified below in Augmented Backus-Naur Form (ABNF) [RFC4234]:

```
          geo-URI        = geo-scheme ":" geo-tile-id \
                           [ *("." geo-extension) ]

          geo-scheme     = "geo"
          geo-tile-id    = 2*32( ALPHA / base32-digits )
          base32-digits  = "2" / "3" / "4" / "5" / "6" / "7"

          geo-extension  = *(ALPHA / DIGITS / "-" / "," )
```

"geo-tile-id" and "geo-extension" are specified in section Section 5.4.1.

(Note: The "geo-extension" component is under discussion, perspective extensions are currently worked on. The authors seek feedback whether an extension mechanism is considered useful)

5.4. URI Scheme Semantics

Data contained in a 'geo' URI identifies a physical resource, namely the geographic coordinates of a spatial area on Earth in the World Geodetic System 1984 [WGS84] datum / reference system.

5.4.1. Component Description

The "geo-tile-id" component of the URI contains a tile-identifier (Base32 encoded), as specified in section Section 6. The "geo-tile-id" component is case-insensitive.

The "geo-extension" component is currently not specified, but applications should be prepared to receive URI instances with such components. Applications should ignore the "geo-extension" for now. The "geo-extension" components may be used in future revisions of the specification to further narrow down the area identified.

Note: An application MAY attempt to recover a mistyped geo-tile-path component by replacing the digit "0" (zero) with the character "o", the digit "1" (one) with "i" and the digit "8" with "B" (even though those character may not occur in that component). No replacement SHOULD be attempted for the digit "9" or any character not listed. An application SHOULD also strip any whitespace from the input string before decoding a geo URI.

5.4.2. URI Comparison

Two 'geo' URIs are equal when their lowercased "geo-tile-id" components are identical, and they contain identical lowercased "geo-extension" components in identical order.

5.5. Properties of the URI scheme

The described URI scheme provides the following properties:

- URI instances are very short compared to other textual methods to convey location - a 'geo' URI with a just 8 character long tile-identifier (including padding and parity) already identifies an area of approximately 150 x 150 meters (in the worst case, close to the equator). An instance with 12 characters of tile-identifier identifies an area of about 0.3 x 0.3 meters.
- By decoding 'geo' URIs while typing, applications can provide rough results to the user even before the full URI has been entered. Whenever the user types another character of the tile-identifier, the application can "zoom in" further.
- It's easy to guess the approximate "scale" of a 'geo' URI by looking at the length of the tile-identifier, much like numbers.
- The choice of Base32 encoding provides a character set that is well known by almost all internet users, since it's a subset of the characters used in Domain Names.
- The parity information provides a mechanism to recognize most mistyped URIs without external information.

- 'geo' URIs can be visually compared. URIs with similar prefixes reflect areas that are spatially nearby.
- Instances of 'geo' URIs are easy to read, spell and write, because the use of special characters has been deliberately limited.

(Note: the current specification does not consider altitude - the authors seek feedback whether that specification should be extended to 3 dimensions)

5.6. Applications/protocols That Use This URI Scheme

The 'geo' URI provides resource identification independent of a specific application or protocol. Examples of potential protocol mappings and use cases can be found in Section 8.

5.7. Interopability Considerations

Applications MAY generate tile-identifiers using either lowercase or uppercase characters during Base32 encoding, but SHOULD NOT mix lowercase and uppercase characters in a single URI instance. Applications MUST be able to decode URI instances with upper-, lower as well as mixed case characters.

FIXME: accept geo URIs with "wrong" padding bits?

5.8. Security Considerations

See Section 10 of [insert reference to this document]

5.9. Contact

Christian Spanring (mailto:spanring@oir.at, http://spanring.eu/),
Alexander Mayrhofer (mailto:alexander.mayrhofer@enum.at,
http://nona.net/)

More information and sample applications can be found at http://www.geouri.org/

5.10. Author/Change controller

The 'geo' URI scheme is registered under the IETF part of the URI tree. As such, change control is up to the IETF.

5.11. References

tbd.

6. The Tile Hierarchy

The "tile-identifier" of the 'geo' URI uses a hierarchical scheme to tile an equirectangular projection of Earth's surface into rectangular tiles.

Tiling starts with a global plate carree (longitude ranging from 180 degrees west to 180 degrees east, and latitude ranging from 90 degrees north to 90 degrees south). The WGS 84 reference system and datum is used. Such a plate carree reflects an empty area bitstring, and can be illustrated as follows:

```
   -180                      0                     +180
 +90 +-----------------------------------------------+
     |                                               |
     |                                               |
     |                                               |
     |                                               |
     |                                               |
   0 |                                               |
     |                                               |
     |                                               |
     |                                               |
     |                                               |
     |                                               |
 -90 +-----------------------------------------------+
```

The first tiling step splits that area vertically, into two aras of (-180 to 0 longitude / +90 to -90 latitude) and (0 to 180 longitude / +90 to -90 latitude), respectively. The "left" area is assigned a binary 0, and the "right" area is assigned a binary 1, as illustrated below:

```
   -180                      0                     +180
 +90 +-----------------------+-----------------------+
     |                       |                       |
     |                       |                       |
     |                       |                       |
     |                       |                       |
     |                       |                       |
   0 |           0           |           1           |
     |                       |                       |
     |                       |                       |
     |                       |                       |
     |                       |                       |
     |                       |                       |
 -90 +-----------------------+-----------------------+
```

A subsequent tiling step involves splitting each of the areas vertically, resulting in a total number of 4 tiles. To identify a sub-tile in this step, a binary 0 is appended to the area bitstring of each individual "upper" tile, and a binary 1 to the "lower" tile:

```
      -180                      0                     +180
    +90 +-----------------------+-----------------------+
        |                       |                       |
        |                       |                       |
        |          00           |          10           |
        |                       |                       |
        |                       |                       |
      0 +-----------------------+-----------------------+
        |                       |                       |
        |                       |                       |
        |          01           |          11           |
        |                       |                       |
        |                       |                       |
    -90 +-----------------------+-----------------------+
```

From now on, vertical and horizontal splitting is continued until the desired resolution is achieved, recursively tiling the whole range of coordinates into smaller areas. Whenever a "sub-area" is split, the new "left" (or "upper") area is identified by appending a binary 0 to the area string , and the new "right" (or "lower") area is identified by appending a binary 1 to the area bitstring. After another repetition of a vertical split, the areas look like this:

```
      -180                      0                     +180
    +90 +-----------+-----------+-----------+-----------+
        |           |           |           |           |
        |           |           |           |           |
        |    000    |    001    |    100    |    101    |
        |           |           |           |           |
        |           |           |           |           |
      0 +-----------+-----------+-----------+-----------+
        |           |           |           |           |
        |           |           |           |           |
        |    010    |    011    |   110     |    111    |
        |           |           |           |           |
        |           |           |           |           |
    -90 +-----------+-----------+-----------+-----------+
```

After another horizontal split, the following areas are identified:

```
      -180                      0                     +180
    +90 +-----------+-----------+-----------+-----------+
        |    0000   |    0010   |   1000    |    1010   |
        |           |           |           |           |
        +-----------+-----------+-----------+-----------+
        |    0001   |    0011   |   1001    |    1011   |
        |           |           |           |           |
      0 +-----------+-----------+-----------+-----------+
        |    0100   |    0110   |   1100    |    1110   |
        |           |           |           |           |
        +-----------+-----------+-----------+-----------+
        |   0101    |    0111   |   1101    |    1111   |
        |           |           |           |           |
    -90 +-----------+-----------+-----------+-----------+
```

After another ("vertical" split), the following area bitstrings exist:

```
      -180                      0                     +180
    +90 +-----+-----+-----+-----+-----+-----+-----+-----+
        |00000|00001|00100|00101|10000|10001|10100|10101|
        |     |     |     |     |     |     |     |     |
        +-----+-----+-----+-----+-----+-----+-----+-----+
        |00010|00011|00110|00111|10010|10011|10110|10111|
        |     |     |     |     |     |     |     |     |
      0 +-----+-----+-----+-----+-----+-----+-----+-----+
        |01001|01101|01100|01101|11000|11001|11100|11101|
        |     |     |     |     |     |     |     |     |
        +-----+-----+-----+-----+-----+-----+-----+-----+
        |01010|01011|01110|01111|11010|11011|11110|11111|
        |     |     |     |     |     |     |     |     |
    -90 +-----+-----+-----+-----+-----+-----+-----+-----+
```

Those splitting steps can be repeated as long until the desired size of the area to be identified is reached. Tiling MUST always be initiated with a vertical split. After a vertical split, the next split MUST be a horizontal split. After a horizontal split, the next split MUST be a vertical split, etc..

The resulting area bitstring is used in encoding tile identifiers (see Section 7).

Note that area bitstrings can be obviously calculated directly as well - the above "splitting" procedure is considered to have illustrative value.

7. Encoding of Tile Identifiers

A "geo" URI contains base32 encoded tile identifiers. This section describes how to transform area bitstrings into their Base32 representation.

7.1. Area Bitstring Padding

Since tiling as outlined above may be stopped at any "resolution", area bitstrings may have arbitrary numbers of binary digits, and therefore not fit into the 5-bit boundaries of Base32. The following process MUST be followed to pad area bitstrings to multiples of 5 bit (the example uses an area bitstring of 12 bits):

1. Identify number of significant bits in area bitstrings (void bits are illustrated as dots below):

```
         01011 10100 11...  (12 significant bits)
area id: --------------
```

2. Pad area bitstring to the next 5-bit border with "0" bits, and record the number of padding bits used in "padding-counter":

```
         01011 10100 11000  (total lenght: 15 bit)
area id: --------------
padding:               ---  (padding-counter: 3)
```

(Note that after padding, the number of significant bit cannot be recovered from the padded area bitstring without the padding-counter).

7.2. Appending Padding Counter and Parity Bits

To recover the number of significant bits in the resulting URI string, the binary representation of padding-counter is appended to the padded area bitstring. Since the padding size is between 0 and 4 bits, 3 bits are required to encode padding-counter.

```
         01011 10100 11000 011  (padding-counter: 3)
area id: --------------
padding:               ---
padding-counter:           ---
```

The encoded padding-counter of 3 bits always leaves 2 bits of "free space" in the last 5 bit group. Those 2 bits are used for parity bits, which are calculated as follows:

- The first parity bit (A) is used to convey even parity over the "longitude" bits

- The second parity bit (B) is used to convey even parity for the "latitude" bits

```
                  01011 10100 11000 01100
         area id: ----- ----- --
     parity bits:                      --
   longitude bit: 0 0 1  0 0  1        0  (parity bit A - longitude)
    latitude bit:  1 1  1 1 0  0        0 (parity bit B - latitude)
```

Note that padding bits MUST NOT be included in parity calculation (the use of '0' bits for padding does not affect parity anyways, however, mangled URI instances could contain '1's in padding).

7.3. Final Base32 Encoding

The bitstream resulting from the operations above (including the significance / parity bits) is Base32 encoded according to section 5 of [RFC3548].

```
             Bitstream: 01011 10100 11000 01100
             Decimal:      11    20    24    12
             Base32:        L     U     Y     M
```

Therefore, the final base32 encoded tile identifier is "LUYM". That string is to be used in the "geo-tile-id" component of the URI (see Section 5.3).

8. Example

The following 'geo' URI identifies a geographic area that resembles the square in front of one of the author's office:

- Center coordinates (approx.): 48.200179 latitude, 16.367957 longitude.
- Number of tiling steps: 34
- Area bitstring: 1000010111001111100111010000111011
- Geographic coordinates of respective area: 48.1997680664 to 48.2011413574 degrees north, 16.3668823242 to 16.3696289062 degrees east (approximately 150 x 300 meters).
- Area bitstring padded:

```
             10000 10111 00111 11001 11010 00011 10110
                                                     =
```

- Area bitstring with padding-counter (1 bit padding):

```
             10000 10111 00111 11001 11010 00011 10110 001
                                                       ===
```

- Adding parity bits:

```
     10000 10111 00111 11001 11010 00011 10110 00110
                                                  ==
lon: 1 0 0  0 1  0 1 1  1 0  1 0 0  0 1  1 1 0 -> 1
lat:  0 0  1 1 1  0 1  1 0 1  1 1  0 0 1  0 1  --> 0
```

- Base32 encoding:

```
10000 10111 00111 11001 11010 00011 10110 00110
  Q     X     H     Z     2     D     W     G
```

- Resulting 'geo' URI:

```
geo:QXHZ2DWG
```

9. IANA Considerations

This document requests assignment of the 'geo' URI scheme in the IETF part of the URI scheme tree, according to the guidelines in BCP 115 (RFC 4395) [RFC4395]. The definitions required for the assignment are contained in Section 5.

10. Security Considerations

Because the 'geo' URI is not tied to any specific protocol, and identifies a physical area rather than an abstract resource, most of the general security considerations on URIs (Section 7 of RFC 3986) do not apply.

Instances of 'geo' URIs convey location information. Such location information may be publicly known, and therefore not be very sensitive (for example, 'geo' URIs conveying the location of public sights, hotels, etc.). However, 'geo' URIs might be used in situations that have considerable privacy implications (for example, the current location of a person combined with Personally Identifieable Information).

Therefore, security and privacy must be considered for individual use cases.

11. References

11.1. Normative References

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

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

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC3548] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.

11.2. Informative References

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 115, RFC 4395, February 2006.

[WGS84] National Imagery and Mapping Agency, "Department of Defense World Geodetic System 1984, Third Edition", NIMA TR8350.2, January 2000.

Authors' Addresses

Alexander Mayrhofer
enum.at GmbH
Karlsplatz 1/9
Wien A-1010
Austria

Phone: +43 1 5056416 34
Email: alexander.mayrhofer@enum.at
URI: http://www.enum.at/

Christian Spanring
OIR-ID GmbH
Franz-Josefs-Kai 27
Wien A-1010

Phone: +43 1 5338747 36
Email: spanring@oir.at
URI: http://www.oir.at/

Full Copyright Statement

Intellectual Property

Acknowledgment

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