Binary to Decimal Conversion for Location Configuration Information

draft-ietf-geopriv-binary-lci-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on June 10, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007)
Abstract

This document describes the nature of the data expressed in the geographic LCI defined in RFC 3825, and includes examples of conversion from its binary format to decimal character strings.

Table of Contents

1. Terminology ......................................................... 2
2. Definitions .......................................................... 2
3. Introduction .......................................................... 3
4. Overview ............................................................... 3
5. Programming Hints .................................................... 4
6. Calculation of LCI values ............................................. 5
7. IANA Considerations .................................................. 6
8. Security ................................................................. 6
9. References .............................................................. 7
   9.1 Normative References .............................................. 7
   9.2 Informative References .......................................... 7
10. Author’s Address ..................................................... 7

1. 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 [0].

2. Definitions

This document uses the following terms to describe geo LCI binary to decimal conversion:

Location Configuration Information: (LCI) An object that carries location information. LCI has no ability to express privacy rules as outlined in [3] and [4], therefore is considered part of the ‘sighting’ function. For purposes of this discussion, all references to LCI refer to its use in [1].

GNU Compiler Collection: (GCC) The GNU Compiler Collection is a set of programming language compilers produced by the GNU Project.
3. Introduction

The LCI encodes a point’s latitude, longitude and altitude, along with the resolution of that point. LCI does not encode boundaries of an arbitrary region. The resolution is nothing more than the representation of significant digits for the fixed-length, binary values in the LCI. This document corrects misinterpretations of the non-normative examples in [1].

Format conversion is required between the binary LCI that a host can receive through DHCP [1] or LLDP-MED [5] and the decimal representation used by applications, e.g. PIDF-LO [2]. This conversion could be used by a host that provides its location to another party with the privacy rules of the [2], including to a server authorized to redistribute the information. It is unclear why anyone would need to convert from the geographic-coordinate location format of [2] to the LCI.

4. Overview

This section provides an overview of the programming hints in the next section for the translation from the efficient binary representation of the LCI [1][5] to the decimal string representation of geographic location used in PIDF-LO [2], for example. GCC syntax is used because it is well known. The binary values are converted to decimal, with the invalid bits removed and with the number of significant digits determined by the resolution of the binary values.

After unpacking the network-order bytes of the LCI into C variables sufficiently large to accommodate the fields, the sign bit of the two's-complement integers are extended to the size of the variable. The sign bit at 34 bits to the left is tested with an octal constant containing 33 bits in 11 octal-digits of zero. If negative, the sign is extended: the upper bits are set to Â‘Â‘ by ORing the value with a value of minus-one with the lower 34 bits inverted to zero with XOR. This operation is safe to perform more than once.

Because [1] says "Contents beyond the claimed resolution MAY be randomized ...", these contents are erased, i.e. set to zero. The number of bits to erase is the field length minus the resolution of the value in that field. A mask is constructed by left-shifting a one into the right of the mask for as many bits as to be erased. ANDing the inverse of the mask with the value erases the invalid bits.

The fixed-point fraction values are scaled into a floating-point (double for enough precision) by dividing by the constant reflecting the number of fractional bits. Note that latitude and longitude have 25 bits of fraction, while altitude has only 22 bits. The number of significant digits to the right of the decimal point is the resolution minus its integer portion, scaled by 3 decimal digits for 10 binary digits because 10 to the 3rd = 1000 approximates 2 to the 10th = 1024.
5. Programming hints

The LCI format is as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----------------+-----------------+-----------------+-----------------+
|     Code 123    |      16        |   LaRes   |     Latitude      |
+-----------------+-----------------+-----------------+-----------------+
| Latitude (cont’d)|   LoRes   |     Longitude |
+-----------------+-----------------+-----------------+-----------------+
|                        Latitude (cont’d)              |
+-----------------+-----------------+-----------------+-----------------+
|                Longitude                         |
+-----------------+-----------------+-----------------+-----------------+
|                  AltRes                           |
+-----------------+-----------------+-----------------+-----------------+
|                   Altitude                          |
+-----------------+-----------------+-----------------+-----------------+
|     AT         |    AltRes     |    Datum       |
+-----------------+-----------------+-----------------+-----------------+
| Alt (cont’d)    |        Datum   |
+-----------------+-----------------+-----------------+-----------------+```

Assume the following element values have been unpacked from the 16 bytes of the wire protocol above.

```c
struct LCIoption {
    int8_t code;        /* DHCP LCI option code = 123 */
    int8_t length;      /* Length 16 bytes - not incl code + length */
    int8_t LaRes;       /* Latitude Resolution 6 bits */
    int64_t Latitude;   /* Latitude 34 bits, 25 fractional */
    int8_t LoRes;       /* Longitude Resolution 6 bits */
    int64_t Longitude;  /* Latitude 34 bits, 25 fractional */
    int8_t AltType;     /* Altitude Type 4 bits */
    int8_t AltRes;      /* Altitude Resolution 6 bits */
    int64_t Altitude;   /* Altitude 30 bits 22 bits Fraction */
    int8_t Datum;       /* Datum code 8 bits */
};
```

Because the latitude, longitude, and altitude values are twos complement of non-standard length, they require sign-extension that is not built into typical variable types. For the Latitude example:

```c
struct LCIoption OptIn;

/* if negative 34-bit field, set all one-bits above the 34-bit field */
if (OptIn.Latitude & 0100000000000LL)
    OptIn.Latitude = OptIn.Latitude | (-1 ^ 0177777777777LL)
/* XOR ‘^’ to flip one bits to zero before ORing in the field */
```
Translation from the binary resolution of the LCI to the correct number of significant decimal digits in the character string representation used for numbers in PIDF-LO is as in the following example for Latitude:

```c
int8_t eraseBits = 34 - OptIn.LaRes;
int64_t mask = 0LL;
if (eraseBits > 0) while (eraseBits--) mask = (mask << 1) | 1;
    /* invert mask and AND to zero invalid bits */
OptIn.Latitude &= ~mask;

double latitude = OptIn.Latitude / exp2 (25);
    /* scale integer for 25 bits of fraction */
int8_t LatFractDigits = (OptIn.LaRes - 9) * 3 / 10;
    /* deduct integer part, 2 to 10 ~= 10 to 3 */

if (LatFractDigits < 0) LatFractDigits = 0;
    /* report integer part if resolution is lower */
printf ("%11.*F\n", LatFractDigits, latitude);
```

6. Calculation of LCI values

Since the Global Positioning System (GPS) or survey methods do not provide location in the LCI format, this section illustrates how a network administrator might calculate the values in preparation for delivering them to hosts connected to her network.

Where geographic location is expressed with the correct number of significant digits, it is easy to compute resolution because 3 decimal digits approximate 10 bits. The number of digits to the right of the decimal point, times 10, divided by 3 is the number of fractional bits. Adding 9 for the integer part yields the resolution.

Where a geographic location comes with an explicit error specification, this error can be translated into the resolution of the LCI. If the error measure is in distance (e.g. meters) rather than degrees, the conversion of longitude to degrees depends on the distance from the equator. Dividing the error distance by the distance for one degree (computed with the method described at [6]) yields the error in (presumably fractional) degrees.
double DegreeError;
int64_t FixedPntErrDeg = DegreeError * exp2 (25);
/* convert error to fixed point 25-bit */

int64_t TopBit = 0100000000000LL;
if (FixedPntErrDeg & TopBit) FixedPntErrDeg = - FixedPntErrDeg;
/* if negative make positive */

/* shift test bit to find first non-zero error */
int8_t resolution = 1;
while ((FixedPntErrDeg & (TopBit >>= 1)) == 0LL) {
    if (TopBit == 0LL) break;
    resolution++;
}
/* the shift count is the number of valid bits */

If all that is available is the bounding points of a region, the difference between the extremes and the center in both latitude and longitude estimates the error in degrees, which can be converted to resolution as above. Find the maximum and minimum of both, calculate the value of the latitude/longitude as the average, and half the difference as the error.

For the example bounds ranging about 0.5 meters in distances across about 32 degrees, the binary and decimal values are as follows:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>000011111111111111111111011110</td>
<td>31.99999850</td>
</tr>
<tr>
<td>0001000000.00000000000000000000101110</td>
<td>32.00000274</td>
</tr>
<tr>
<td>001000000.00000000000000000000010101</td>
<td>64.00000124 sum</td>
</tr>
<tr>
<td>000100000.00000000000000000000010110</td>
<td>32.00000062 average</td>
</tr>
<tr>
<td>000000000.000000000000000000000010001110</td>
<td>00.00000423 difference</td>
</tr>
</tbody>
</table>

With 26 bits above the difference, which is twice the error, this example yields 27 bits of resolution (remembering to add 9 bits for left of the binary point).

7. IANA Considerations

No IANA Considerations

8. Security

This document discusses binary to decimal conversion within an end host, which raises no particular security considerations.
9. References

9.1 Normative References

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


9.2 Informative References


10. Author’s Address

John Schnizlein
Cisco Systems, Inc.
Fort Washington, MD, USA
Email: john.schnizlein@cisco.com

Marc Linsner
Cisco Systems, Inc.
Marco Island, Florida, USA
Email: marc.linsner@cisco.com

Comments are solicited and should be addressed to the working group’s mailing list at geopriv@ietf.org and/or the authors.