Consumer Technology Association™

CTA Specification

Fast and Readable Geographical Hashing

CTA-5009-A

SVTA Streaming Video Technology Alliance

June 2024

NOTICE

Consumer Technology Association (CTA)™ Standards, Bulletins and other technical publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards, Bulletins and other technical publications shall not in any respect preclude any member or nonmember of the Consumer Technology Association from manufacturing or selling products not conforming to such Standards, Bulletins or other technical publications, nor shall the existence of such Standards, Bulletins and other technical publications preclude their voluntary use by those other than Consumer Technology Association members, whether the standard is to be used either domestically or internationally.

Standards, Bulletins and other technical publications are adopted by the Consumer Technology Association in accordance with the American National Standards Institute (ANSI) patent policy. By such action, the Consumer Technology Association does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Standard, Bulletin or other technical publication.

This document does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this document to establish appropriate safety and health practices and to determine the applicability of regulatory limitations before its use.

(Formulated under the cognizance of the CTA **WAVE Project**; for information please see cta.tech/WAVE.)

Published by
CONSUMER TECHNOLOGY ASSOCIATION
Technology & Standards Department
www.cta.tech

FOREWORD

This document was developed by the Web Application Video Ecosystem (WAVE) Project of the Consumer Technology Association. The WAVE Project is a broad industry initiative of content, technology, infrastructure and device companies, all working together towards commercial Internet video interoperability based on industry standards.

Revision History

CTA-5009-A changes the CBOR Tag ID from 105 to 301, which is in a less restricted allocation space. It also corrects a typo in the CDDL for the CBOR tag.

(This page intentionally left blank.)

TABLE OF CONTENTS

(This page intentionally left blank.)

Fast and Readable Geographical Hashing

1 INTRODUCTION

Geographical hashes ("Geohashes") are short strings of letters and digits that identify a specific region. Longer Geohashes represent smaller regions and shorter Geohashes represent larger regions. In particular, a Geohash that is the prefix of a second longer Geohash represents a region that encloses the region represented by the second.

Geohashes are often used to identify locations by describing a region that encloses the location, including any uncertainty associated with the location data. A location with very large uncertainty would be represented with a short Geohash. A very long Geohash can represent a location very precisely.

A Geohash always describes a region bounded by two lines of longitude and two lines of latitude. The physical distance between lines of latitude decreases near the poles, so the amount of physical space described by a Geohash of a given length will vary depending on its location. Geohashes that describe regions near the equator will describe larger areas than Geohashes of the same length that describe regions near the poles.

2 REFERENCES

The following documents are referenced in this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed here.

[GEOHASH-WIKI] "Geohash" https://en.wikipedia.org/w/index.php?title=Geohash&oldid=1126143295 (accessed 2023-12-11).

[STD94] STD 94/IETF RFC 8949, *Concise Binary Object Representation (CBOR),* December 2020, https://www.rfc-editor.org/info/std94.

[CBOR-IANA] IANA *Concise Binary Object Representation (CBOR) Tags*, https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml.

[RFC7519] IETF RFC 7519, *JSON Web Token (JWT)*, DOI 10.17487/RFC7519, May 2015, https://www.rfc-editor.org/info/rfc7519.

[RFC8392] IETF RFC 8392, *CBOR Web Token (CWT)*, DOI 10.17487/RFC8392, May 2018, https://www.rfc-editor.org/info/rfc8392.

[RFC8610] IETF RFC 8610, *Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures*, DOI 10.17487/RFC8610, June 2019, https://www.rfc-editor.org/info/rfc8610.

[ISO 3166] ISO 3166, *Codes for the representation of names of countries and their subdivisions*, International Organization for Standardization, https://www.iso.org/standards.html.

[WGS84] US National Imagery and Mapping Agency, "Department of Defense (DoD) World Geodetic System 1984 (WGS 84), Third Edition", NIMA TR8350.2, January 2000.

3 COMPLIANCE NOTATION

This document uses the following compliance terms:

MUST	This word indicates specific provisions that are to be followed strictly (no deviation is permitted).
MUST NOT	This phrase indicates specific provisions that are absolutely prohibited.
SHOULD	This word indicates that a certain course of action is preferred but not necessarily required.
SHOULD NOT	This phrase means a certain possibility or course of action is undesirable but not prohibited.
MAY	This phrase indicates that a certain course of action is optional.

4 HISTORY

Geohash has historically been described in implementations as well as in a relatively comprehensive article on Wikipedia [GEOHASH-WIKI]. These are excellent resources, but they can change over time and are not generally suitable for building on in a consensus-based standards-making process.

This document exists to remediate that. It describes the practice as it exists at the time of writing and provides general, straightforward algorithms for understanding and communicating with Geohashes. It also exists to serve as a stable reference document for a few IANA registrations.

To that end, the algorithm documented in the “Technical description” section of [GEOHASH-WIKI] is described here. This document does not describe all the usages, limitations, or geographic and mathematical details for Geohashes because those are well described by the references.

The algorithm provided in [GEOHASH-WIKI] only decodes a Geohash, so the encoding routine has been calculated as the reverse of decoding. However, since Geohashes of a given length may not be able to describe exactly the region to encode, the algorithm must make some rounding choices. In every case, it rounds so as to describe a region that contains the region to encode.

5 TERMINOLOGY

References for Geohashes often refer to a Geohash as a “location” with a given “precision”. This framing is not precisely wrong, but it is imprecise. This document is careful to describe a Geohash as describing a region, not a location. For some usages, a small enough region may effectively describe a location, but this is an application-dependent decision.

This document uses “precision” to refer to the size of the region a given Geohash describes, given in bits or degrees. More bits of precision results in smaller regions. These regions are cells aligned to the

Geohash grid, not positioned arbitrarily. As a result, when a location is encoded to a cell, the “error” is not evenly distributed in all directions.

6 NOTATION

There is some mathematical notation below in the encoding and decoding. These are to be read in accordance with usual mathematical practice.

⌊*x*⌋ is understood to mean the "floor of *x*". That is, the largest integer smaller than or equal to *x*.

⌈*x*⌉ is understood to mean the "ceiling of *x*". That is, the smallest integer larger than or equal to *x*.

For clarity, words are used instead of single letter variables. So latitude is one value named "latitude", NOT a product of eight single letter values that simplifies to $adeilt^2u$.

Likewise, a word in front of parentheses is understood to be a function, not multiplication. *min(a,b)* is a function named "min" with arguments "a" and "b".

Interval notation is used herein. Square brackets [] represent inclusive endpoints and parentheses () represent exclusive endpoints. So the interval [1, 5) includes all the values between one and five, and includes one, but does not include five.

When describing numbers, they are prefixed to indicate the base. Decimal (base 10) numbers have no prefix. Hexadecimal numbers are prefixed with "0x" and binary numbers are prefixed with "0b". It is often useful to add separators inside the number to make it easier to read. Commas (",") and underscores ("_") in numbers have no meaning on their value but can be used to illustrate grouping or improve readability.

The formulae herein define numbers and their relations exactly, but the examples round as is suitable for display. When example numbers are not exact, this is indicated with the ≈ symbol. Implementations that use floating point numbers to represent values may have rounding errors introduced, as usual. Management of these rounding errors is out of the scope of this document.

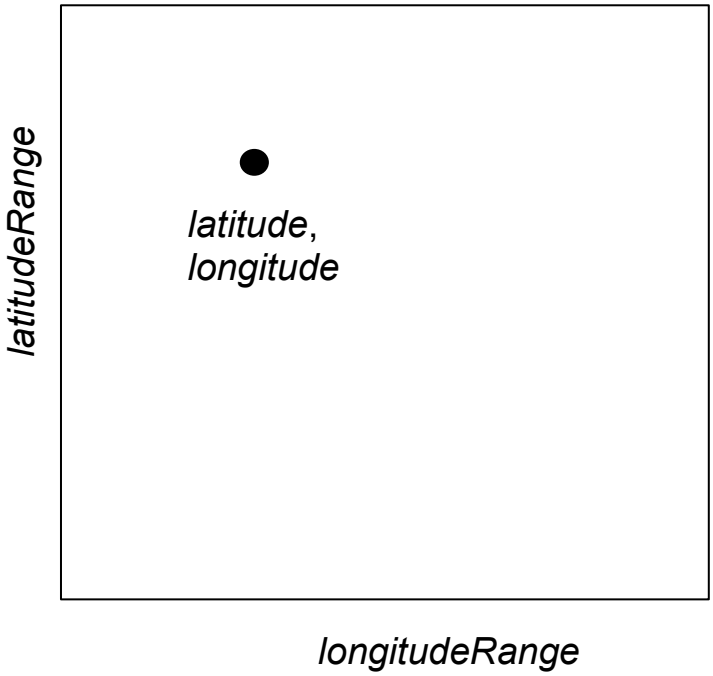
7 ENCODING

The inputs to the Geohash encoding algorithm are:

- A Latitude and Longitude inside the region to be encoded; and
- Latitude and Longitude Ranges to be encoded.

Input Ranges: Latitudes MUST be in the range [-90°,90°). Longitudes MUST be in the range [-180°, 180°).

The resultant Geohash will represent a region on Earth that contains the point described by the $latitude$ and $longitude$. The region will be no smaller (as measured by latitude and longitude) than $latitudeRange$ and $longitudeRange$ but may be much larger. This is because Geohash regions are aligned to a series of grids, not arbitrary regions.



Each of these inputs is in the units of decimal degrees. The Ranges represent the size of the range to be encoded. Because not all ranges are representable by Geohashes, the encoded Range may be larger (but not smaller) than the input Ranges. An input shape with a very large difference between the Latitude Range and the Longitude Range will experience the most variation when encoded. These shapes are sometimes better broken up into multiple smaller regions that will encode with less error.

Alternately, Ranges can be efficiently described by specifying the desired final Geohash Length.

Geohashes are encoded in four steps:

1. Convert Latitude and Longitude Ranges to Latitude and Longitude Lengths.
2. Convert Latitude and Longitude to their respective Latitude and Longitude Codes.
3. Combine Latitude and Longitude codes into a single Binary Geohash.
4. Encode the Binary Geohash as a string.

7.1 Lengths

The easiest way to get latitude and longitude lengths is by determining them from the Geohash Lengths:

$$latitudeLength = \lfloor 2.5 \times geohashLength \rfloor$$
$$longitudeLength = \lceil 2.5 \times geohashLength \rceil$$

Geohash Length can be determined by selecting the desired precision:

Geohash Length	Lat Precision	Long Precision	km Precision	Miles Precision
1	23°	23°	2500 km	1600 mi
2	2.8°	5.6°	630 km	390 mi
3	0.7°	0.7°	78 km	48 mi
4	0.087°	0.18°	20 km	12 mi
5	0.022°	0.022°	2.4 km	1.5 mi
6	0.0027°	0.0055°	0.61 km	0.38 mi
7	0.00068°	0.00068°	0.076 km	0.047 mi
8	0.000085°	0.00017°	0.019 km	0.012 mi
9	0.000021°	0.000021°	0.0023 km	0.0014 mi

These precision values have maximum size when one of the boundaries is at the equator. The longitude error varies with the cosine of the longitude, so at 40°, the size will be about 70% of what is listed. At 60°, it will be half. Given that at even Geohash Lengths, the decimal degrees size for longitude is twice that of the latitude, this means that these Geohashes can often describe regions with smaller sizes than the maximums listed above. Very near the poles, these errors can be quite small in the longitude direction.

7.1.1 Alternate Length Calculation

If the desired Geohash Length is unknown, it can be calculated from Latitude and Longitude Ranges.

To convert Latitude and Longitude Ranges to Latitude and Longitude Lengths, start by calculating the Initial Lengths:

$$initialLatitudeLength = \left\lfloor \boldsymbol{log}_2 \left(\frac{180°}{latitudeRange}\right) \right\rfloor$$
$$initialLongitudeLength = \left\lfloor \boldsymbol{log}_2 \left(\frac{360°}{longitudeRange}\right) \right\rfloor$$

The Lengths provide the length, in bits, of the Latitude and Longitude Codes. Not all lengths are representable, however, in geocodes.

Reduce the Initial Latitude Length to the largest multiple of 2.5 rounded down smaller than it and reduce the Initial Longitude Length to the largest multiple of 2.5 rounded up smaller than it:

$$initialLatitudeLength_2 = \left\lfloor 2.5 \times \left\lfloor \frac{initialLatitudeLength + 0.5}{2.5} \right\rfloor \right\rfloor$$

$$initialLongitudeLength_2 = \left\lceil 2.5 \times \left\lfloor \frac{initialLongitudeLength}{2.5} \right\rfloor \right\rceil$$

And then calculate the final ranges by reducing a larger length to match the other length if necessary:

$$\text{latitudeLength} = \left\lfloor 2.5 \times min\left(\left\lfloor \frac{initialLatitudeLength_2 + 0.5}{2.5} \right\rfloor, \left\lfloor \frac{initialLongitudeLength_2}{2.5} \right\rfloor \right) \right\rfloor$$

$$\text{longitudeLength} = \left\lceil 2.5 \times min\left(\left\lfloor \frac{initialLatitudeLength_2 + 0.5}{2.5} \right\rfloor, \left\lfloor \frac{initialLongitudeLength_2}{2.5} \right\rfloor \right) \right\rceil$$

7.2 Codes

To convert Latitude and Longitudes into their respective Codes, each is scaled into its range:

$$\text{latitudeQstep} = \frac{180°}{2^{latitudeLength}}$$

$$longitudeQstep = \frac{360°}{2^{longitudeLength}}$$

$$latitudeCode = \left\lfloor \frac{latitude + 90°}{latitudeQstep} \right\rfloor$$

$$longitudeCode = \left\lfloor \frac{longitude + 180°}{longitudeQstep} \right\rfloor$$

7.3 Binary Geohash

In order to combine the Codes into a single Binary Geohash, their bits must be interleaved. Given a function $interleave(a, b)$ that takes two numbers a and b and interleaves their bits such that the least significant bit of the result will be the least significant bit of b, the interleaving will depend on whether the lengths are equal or not equal.

If the Latitude Length is equal to the Longitude Length:

$$binaryGeohash = interleave(longitudeCode, latitudeCode)$$

Otherwise:

$$binaryGeohash = interleave(latitudeCode, longitudeCode)$$

7.4 String

To encode the Binary Geohash as a string, start by calculating the length, unless it was part of the initial input as an alternative way to define the ranges:

$$geohashLength = \left\lfloor \frac{longitudeLength}{2.5} \right\rfloor$$

This Geohash Length will be the number of characters in the Geohash. The Geohash is almost identical to a numeral using positional notation in base 32 and an alternate set of digits. The meanings of the Geohash digits are given by the following table:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**	**n**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Note that the Geohash digits omit several symbols that could be easily confused with one another when represented visually. Geohash encoders MUST use lowercase letters when encoding.

As a positional numeral in base 32, each symbol in position n, where position 0 is the least significant digit, is given by the following:

$$geohash(n) = (binaryGeohash >> \ 5 \times n) \ \& \ 0x1F$$

Where $>>$ is the right-shift operator and $\&$ is the bitwise "and" operator. And 0x1F is the hexadecimal value representing 31.

Geohashes are almost positional numerals, but they carry one additional piece of information that true positional numerals do not: length. While positional numerals that differ only by leading zeroes are not different, Geohashes that differ by leading zeroes are different. The algorithms to produce positional numerals and Geohashes are the same as long as they process exactly Geohash Length number of digits, even if the leading digits are zero.

7.5 Encoding Common Regions

In order to encode a Geohash that describes a region guaranteed to enclose a nonzero area described by two bounding latitudes y_1 and y_2 and two bounding longitudes x_1 and x_2, where the first in each pair is smaller than the second, do this:

1. Encode a region enclosing latitude y_1 and longitude x_1 with the desired degree of precision.
2. Do likewise for a region enclosing latitude y_2 and x_2.
3. Take the common prefix of the two Geohashes. That is, take the longest prefix that is the same for both Geohashes.

That common prefix will enclose both regions. It may, however, be drastically larger than that described by the initial bounds. In a worst-case scenario, the common prefix may be zero length, which describes the entire globe.

If, in step 3, the common prefix is the whole of both Geohashes, it means that both points lie within the same Geohash region to the degree of precision specified in step 1. If the most precise Geohash that describes the region is required, repeat the steps with a larger degree of precision until the Geohashes differ.

When attempting to encode a point with a known error, this can pose a challenge. Given latitude and longitude with known error for each value, the geohash that encloses that region may be untenably large. How this is handled is application-specific, but encoders should be aware that the error values for Geohashes are not evenly distributed and multiple more-precise Geohashes may be preferred in order to usefully describe the region.

7.6 Example

Consider a location to encode: 32.449247755342455° latitude, -99.73357454336144° longitude; and a precision that mustn't be worse than 0.0001°.

The simplest method is to look up on the chart above and observe that a length of 9 is the first value that does not exceed this in either dimension.

Plug in the numbers:

$$latitudeLength = \lfloor 2.5 \times 9 \rfloor = 22$$
$$longitudeLength = \lceil 2.5 \times 9 \rceil = 23$$

$latitudeQstep = \frac{180°}{2^{22}} \approx 0.000042915344238281 25°$

$\text{longitudeQstep} = \frac{360°}{2^{23}} \approx 0.000042915344238281 25°$

$latitudeCode = \left\lfloor \frac{32.449247755342455° + 90°}{0.000042915344238281 25°} \right\rfloor = 2{,}853{,}274 = 0x2B899A$; binary: 0b0010_1011_1000_1001_1001_1010

$longitudeCode = \left\lfloor \frac{-99.73357454336144° + 180°}{0.000042915344238281 25°} \right\rfloor = 1{,}870{,}343 = 0x1C8A07$; binary: 0b0001_1100_1000_1010_0000_0111

Interleave those bits to get:

```
(lat)  0b0 0 1 0 _1 0 1 1 _1 0 0 0 _1 0 0 1 _1 0 0 1 _1 0 1 0
(long) 0b 0 0 0 1_ 1 1 0 0_ 1 0 0 0_ 1 0 1 0_ 0 0 0 0_ 0 1 1 1
(both) 0b00001001_11011010_11000000_11000110_10000010_10011101
```

Split into 9 sets of 5 instead:

0b01001_11011_01011_00000_01100_01101_00000_10100_11101

Convert to positional values:

9, 27, 11, 0, 12, 13, 0, 20, 29

Convert to final string values:

9vc0de0nx

This encoding specifies a Geohash cell of size $latitudeRange \times longitudeRange$ to represent a region enclosing the desired point 32.449247755342455° latitude, -99.73357454336144° longitude. The region's $latitudeRange$ and $longitudeRange$ were selected to be smaller than the required precision of 0.0001° (or roughly 10 m). We were able to encode with bitstrings of length $latitudeLength$ and $longitudeLength$ as an intermediate step, which resulted in a Geohash string of length 9 characters after encoding was completed.

8 DECODING

The input to the Geohash decoding algorithm is just a Geohash string to be decoded.

There are three steps for decoding Geohashes:

1. Decode the string into a Binary Geohash.
2. Split the Binary Geohash into the Latitude and Longitude Codes and Lengths.
3. Determine the Latitude and Longitude Ranges and values.

Every Geohash describes a specific region.

8.1 Binary Geohash

Convert the Geohash to a number using the table above and interpreting it as a positional numeral:

$$\text{binaryGeohash} = \sum_{n=0}^{geohashLength} \text{ungeohash(n)} \times 32^n$$

Geohash decoders SHOULD interpret upper case letters as though they were lower case. A Geohash with characters not on the table is invalid and MUST NOT be processed.

8.2 Codes

The Latitude and Longitude Codes are extracted from the Geohash Value by taking alternating bits. When the number of characters in the Geohash String is even, the number of bits in the Latitude and Longitude Codes will both be the number of characters in the Geohash String multiplied by 2.5. In this case, the latitude is formed from the even bits in the Geohash Value, the longitude the odd, starting from position zero in the last bit of the value.

When the number of characters in the Geohash String is odd, the number of bits in the latitude is the number of characters in the Geohash Value, multiplied by 2.5, then rounded down by subtracting 0.5. The number of bits in the longitude is the number of characters in the Geohash Value, multiplied by 2.5, then rounded up by adding 0.5. In this case, the latitude is formed from the odd bits and the longitude the even bits, starting from position zero in the last bit of the value.

$$latitudeLength = \lfloor geohashLength \times 2.5 \rfloor$$

$$longitudeLength = \lceil geohashLength \times 2.5 \rceil$$

If the Latitude Length is equal to the Longitude Length (which is to say, if the Geohash Length is even):

$$longitudeCode, latitudeCode = deinterleave(binaryGeohash)$$

Otherwise:

$$latitudeCode, longitudeCode = deinterleave(binaryGeohash)$$

8.3 Values

The Latitude and Longitude Codes define values and ranges for latitudes and longitudes.

$$latitudeRange = \frac{180°}{2^{latitudeLength}}$$

$$latitude = (latitudeRange \times latitudeCode) - 90°$$

$$longitudeRange = \frac{360°}{2^{longitudeLength}}$$

$$longitude = (longitudeRange \times longitudeCode) - 180°$$

The Geohash represents the region on earth bounded the lines of latitude $latitude$ and $latitude + latitudeRange$ and the lines of longitude $longitude$ and $longitude + longitudeRange$.

8.4 Determining If a Point Lies in a Geohash Region

To determine if a point lies in a given Geohash Region:

1. Encode the point with the same Geohash Length as the Region. If the point is already encoded as a Geohash, this is accomplished by truncating it to the length of the region. If the point is encoded with lower Geohash Length than the Region, it will be impossible to determine if it lies within the Region because its error is higher than the size of the Region.
2. Check to see if they are identical, using a case-insensitive comparison.

If they are the same string, the point lies within the Geohash Region.

8.5 Example

Consider the string "9vc0de0nx" as a Geohash to be decoded. (This is the example given above for encoding.)

First, the length is noted: 9.

Then, it is converted using the table to positional values:

9, 27, 11, 0, 12, 13, 0, 20, 29

These are encoded positionally:

$$9 \times 32^8 + 27 \times 32^7 + 11 \times 32^6 + 0 \times 32^5 + 12 \times 32^4 + 13 \times 32^3 + 0 \times 32^2 + 20 \times 32^1 + 29 \times 32^0$$

$$= 10{,}835{,}141{,}755{,}549 = 0x009D\_AC0C\_829D$$

That value, in binary, is:

```
(both) 0b00001001_11011010_11000000_11000110_10000010_10011101
(lat)  0b0 0 1 0 _1 0 1 1 _1 0 0 0 _1 0 0 1 _1 0 0 1 _1 0 1 0
(long) 0b 0 0 0 1_ 1 1 0 0_ 1 0 0 0_ 1 0 1 0_ 0 0 0 0_ 0 1 1 1
```

Deinterleave that into two parts:

0b0010_1011_1000_1001_1001_1010 = 2853274

0b0001_1100_1000_1010_0000_0111 = 1870343

Since the length is odd, the first number is the Latitude Code and the second is the Longitude Code.

Then, we determine the code lengths:

$$latitudeLength = \lfloor 9 \times 2.5 \rfloor = 22$$

$$longitudeLength = \lceil 9 \times 2.5 \rceil = 23$$

And the ranges:

$$latitudeRange = \frac{180°}{2^{22}} \approx 0.00004291534423828125°$$

$$longitudeRange = \frac{360°}{2^{23}} \approx 0.00004291534423828125°$$

And finally, the values:

$$latitude = (0.00004291534423828125 \times 2853274) - 90° \approx 32.4492359161376953125$$

$$longitude = (0.00004291534423828125 \times 1870343) - 180° \approx 99.73358631134033203125$$

Adding the ranges to these values, we can determine that the range covers the area between latitudes 32.4492359161376953125° and 32.44927883148193359375° and between longitudes -99.73358631134033203125° and -99.73354339599609375°.

8.6 Checking against the original values

The original point encoded was 32.449247755342455° latitude, -99.73357454336144° longitude. One can validate that:

32.449**235**9161376953125 < 32.449**247**755342455 < 32.449**278**83148193359375

and:

-99.733**586**31134033203125 < -99.733**574**54336144 < -99.733**543**39599609375

(The highlighted portion shows where the values begin to differ.)

This decoding confirms that the string "9vc0de0nx" represents the encoding of a Geohash cell of size $latitudeRange \times longitudeRange$ that identifies a region enclosing the original point, and that the region's $latitudeRange$ and $longitudeRange$ are smaller than the required precision of 0.0001° (or roughly 10 m), and that we were able to do it with bitstrings of length $latitudeLength$ and $longitudeLength$, which after encoding resulted in a Geohash string of length 9 characters.

9 CAUTIONARY EXAMPLE

Consider the situation if a single Geohash is used to describe the region within 44.999° latitude, -90.001° longitude and 45.001° latitude, -89.999° longitude. This could, for example, be the result of a subject very near the 45° latitude, -90° longitude with a thousandth of a degree error on the sensor in both directions.

To find the common region, we encode both and take the prefix. We start with a prefix length of 5, since the size of the region is about 0.02°. (The starting precision does not ultimately matter. If it is insufficient, the regions will match and you can try again with a larger value, per the algorithm.) We start by encoding both points:

44.999° latitude, -90.001° longitude:

$latitudeCode = \left\lfloor \frac{44.999° + 90°}{0.043945°} \right\rfloor = 3071 = 0xBFF$; binary: 0b1011_1111_1111

$longitudeCode = \left\lfloor \frac{-90.001° + 180°}{0.043945°} \right\rfloor = 2047 = 0x7FF$; binary: 0b0_0111_1111_1111

Interleave and group by fives to get:

0b01001_11111_11111_11111_11111

Encode that to get:

9zzzz

45.001° latitude, -89.999° longitude:

$latitudeCode = \left\lfloor \frac{45.001° + 90°}{0.043945°} \right\rfloor = 3072 = 0xC00$; binary: 0b1100_0000_0000

$longitudeCode = \left\lfloor \frac{-89.999° + 180°}{0.043945°} \right\rfloor = 2048 = 0x800$; binary: 0b0_1000_0000_0000

Interleave and group by fives to get:

0b01110_00000_00000_00000_00000

Encode that to get:

f0000

Then, we find the common prefix of "9zzzz" and "f0000", which is an empty string. A zero-length Geohash describes the entire planet, so the most precise Geohash that describes the given region encloses the whole planet. This is not particularly useful, so care must be taken when using direct error ranges and specific values.

10 PRACTICAL EXAMPLE

Consider a practical situation in Paris. If the goal is to describe Paris with a single Geohash, the smallest Geohash that covers the whole of Paris is "u09". However, this zone is very large, encompassing all of the Greater Paris Metropolis and much of the countryside, all the way down to Orléans.

Let's suppose that Paris is a rectangle bounded by the points 48.835707° latitude, 2.284042° longitude and 48.898580° latitude, 2.391896° longitude. It is not, but it is a useful simplification. With nine characters, the former encodes to "u09tgfr0w" and the latter to "u09wnmtwz". The shortest common prefix is "u09", as described, which overstates the area quite drastically.

If, however, you use an array of more precise Geohashes, you can more closely cover the area:

["u09tg","u09tu","u09tv","u09ty","u09w5",u09wh","u09wj","u09wn"]

Calculating these fields is outside the scope of this document, but it is a useful demonstration of how a few more precise Geohashes are sometimes a more useful way to accurately encode a large area. This is especially true when the area is not a rectangle.

11 REGISTRATIONS

This document requests a number of registrations of the Internet Assigned Numbers Authority, as described in the sections below.

11.1 Concise Data Definition Language

Some registrations have Concise Data Definition Language (CDDL) [RFC8610] definitions associated. As a convention, specifications using CDDL define a single definition first, in order to assist machine processors. It is recommended that any such processors process each of the following sections as though they were separate CDDL documents. However, for convenience, the following CDDL is provided first:

```
geohash-registrations = geohash / crsw / cwt-claims
```

12 CBOR TAG

The Geohash CBOR tag [STD94] identifies a Geohash String.

```
Tag: 301
Data Item: text string or array
Semantics: Geohash String
Point of Contact: Consumer Technology Association
```

When the data item is a text string, it is a Geohash that represents a region with the semantics above. A sufficiently small region can represent a location for some applications. When the data item is an array, it MUST be an array of text strings that are Geohashes. An array of Geohashes represents a region that is the union of the regions represented by each Geohash. The regions may be non-adjacent and/or overlapping.

A Geohash CBOR tag is described by the CDDL [RFC8610]:

```
geohash = #6.301(tstr) / #6.301([ * tstr ])
```

Some application-specific purposes may allow Geohash strings and arrays to be wrapped with Coordinate Reference System Wrappers.

12.1 Coordinate Reference System (CRS) Wrapper

The CRS CBOR tag identifies a Coordinate Reference System that wraps another data object.

```
Tag: 279
Data Item: array
Semantics: Coordinate Reference System Wrapper
Point of Contact: Consumer Technology Association
```

The data item is an array that MUST have exactly two elements. The first element MUST contain a data item that represents a Coordinate Reference System (CRS). The second element is a data item that is the element being wrapped. The meaning of this tag is application-specific but SHOULD be the same as the meaning of the second data item, interpreted in the CRS of the first. If a tag is implied by the context of the wrapper, the same implication SHOULD apply to the second data item.

When the first data item is not tagged, it MUST be interpreted as though it were tagged with tag 104 [CBOR-IANA] (Geographic Coordinate Reference System WKT or EPSG number).

A CRS Wrapper tag is described by the CDDL:

```
crsw = #6.279([crs: any, data: any])
```

In most cases, the context of the wrapper will constrain the data object.

13 JWT CLAIM

```
Claim Name: geohash
Claim Description: Geohash String or Array
Change Controller: Consumer Technology Association
```

The "geohash" claim identifies a geographical region associated with the subject of the JSON Web Token [RFC7519]. The processing of this claim is generally application-specific. Sufficiently small regions can be used to represent locations for some applications. The "geohash" claim value is either an array of case-sensitive strings or a single case-sensitive string. A single string is a Geohash String and represents the region described by the Geohash.

An array of Geohashes represents the union of the regions described by the Geohashes. These regions may be non-adjacent and/or overlapping. A list of sufficiently small regions may be used to describe a list of locations.

Unless the application-specific context defines otherwise, the regions described by a Geohash claim MUST be interpreted in the WGS84 coordinate system as defined in [WGS84].

14 CWT CLAIM

```
Claim Name: geohash
Claim Description: Geohash String
JWT Claim Name: geohash
Claim Key: 282
Claim Value Type(s): text string or array
Change Controller: Consumer Technology Association
```

The "geohash" CBOR Web Token [RFC8392] claim has the same meaning and processing rules as the "geohash" JWT claim, except that its type is a CBOR text string or array. The Claim Key 282 is used to identify this claim. The value of this claim MUST NOT be prefixed by a Geohash String tag (301). The semantics of this claim already require that the value be interpreted as a Geohash String and so a tag prefix would consume space for no benefit.

This claim value MAY be prefixed with a CRS Wrapper tag (279) in accordance with the semantics defined by that tag. If a CRS Wrapper is used, it MUST be tagged so that parsers can differentiate an array that is a CRS Wrapper from an array of Geohash strings. Likewise, each Geohash string in an array MAY be prefixed with a CRS Wrapper tag.

An application that uses a "geohash" claim MUST specify a default Coordinate Reference System that applies when no CRS Wrapper is present, or it MUST require a CRS Wrapper. It MUST specify whether a CRS Wrapper that describes a CRS that is already described by the context (either explicitly or as a default) is permitted, prohibited, or required. It SHOULD prohibit it unless application-specific requirements make that impossible or undesirable.

The value of a "geohash" claim in a CWT is described by the CDDL:

```
cwt-claims = {
      ? geohash-label => geohash-value
      * label => any

}
label = int / tstr

geohash-label = 282
geohash-value = untagged-geohash-value
```

```
              / #6.279([crsw-system, untagged-geohash-value])
untagged-geohash-value = geohash-string
                       / [ * geohash-string ]
geohash-string = tstr
               / #6.279([crsw-system, tstr])
crsw-system = any ; interpreted as tag 104 when untagged
```

15 SECURITY CONSIDERATIONS

A Geohash is nothing more or less than a description of a specific region. As such, any security considerations exist entirely within the context in which it is used. When a Geohash is used as a basis for access control, the security of the access control mechanism will only be as secure as the Geohash itself.

Geohashes used for sensitive purposes MUST be conveyed securely. What that means will depend on the context; implementers should refer to the security considerations of the related protocols for this.

Likewise, Geohashes MUST be sufficiently accurate for the purposes to which they are set. When they are based on sensors with accuracy limits, these limits and errors will contribute to the error in the ultimate usage of the Geohash. Providing sets of Geohashes that encompass the entire error of the sensor will often be a useful way to accommodate this.

Special care should be taken when using Geohashes to determine or describe political or civic boundaries. Adequately describing these regions is difficult without using large sets of Geohashes, because they tend to have highly irregular borders. A protocol using a Geohash for this purpose may also convey the intended political or civic region using other protocols for that purpose, such as [ISO 3166] country and region codes.

16 PRIVACY CONSIDERATIONS

Geohashes describe locations. To the extent that they describe people, places, and other entities with privacy interests, they MUST be kept confidential by use of encryption. Likewise, a protocol MUST use the lowest suitable precision.

Exactly how these values are protected and what constitutes suitable precision will depend on context.

Geohashes can gain considerable precision when an adversary can observe them over time as well. Sensor jitter or movement of the subject can result in Geohash changes that allow adversaries to observe boundary changes and deduce much more accurately where in the Geohash the subject is. Geohashes based on sensors or moving subjects MUST be conveyed using suitable encryption when privacy interests are at risk.

Annex A
Informative

A ENCODING TEST VECTORS

Given the following locations and lengths, these are the encoded Geohash Strings:

Latitude	Longitude	Length	Geohash
0	0	9	s00000000
-89.99	-179.99	4	0000
48.856667	2.352222	9	u09tvw0fd
32.449247755342455	-99.73357454336144	9	9vc0de0nx
89	179	4	zzz6
32.449247755342455	-99.73357454336144	5	9vc0d
24.668889	102.977222	4	wk3h
8.529722	77.249722	3	t9w
-18.286111	147.7	6	rk9pbz
48.9	22.183333	4	u2xy
-9.33333	-77.4	5	6q2fy
4	-56.5	7	d8xyf21
9.119355	-79.731240	8	d1x7csjk
17.073	-119.114	5	97531
21.972	69.2571	4	tech
-72.0778	123.2274	7	neptune

Annex B
Informative

B DECODING TEST VECTORS

Given the following Geohash Strings, these are the regions they encode, to six decimal places:

Geohash	Latitude	Longitude	Latitude Range	Longitude Range
s00000000	0	0	0.000043	0.000043
0000	-90	-180	0.175781	0.351563
u09tvw0fd	48.856630	2.352190	0.000043	0.000043
9vc0de0nx	32.449236	-99.733586	0.000043	0.000043
zzz6	88.945313	178.945313	0.175781	0.351563
9vc0d	32.431641	-99.755859	0.043945	0.043945
wk3h	24.609375	102.65625	0.175781	0.351563
t9w	8.4375	75.9375	1.40625	1.40625
rk9pbz	-18.286743	147.689209	0.005493	0.010986
u2xy	48.867188	22.148438	0.175781	0.351563
6q2fy	-9.360352	-77.431641	0.043945	0.043945
d8xyf21	3.999023	-56.501312	0.001373	0.001373
d1x7csjk	9.119339	-79.731560	0.000172	0.000343
97531	17.050781	-119.135742	0.043945	0.043945
tech	21.796875	68.90625	0.175781	0.351563
neptune	-72.078552	123.226776	0.001373	0.001373

Consumer Technology Association Document Improvement Proposal

If in the review or use of this document a potential change is made evident for safety, health or technical reasons, please email your reason/rationale for the recommended change to standards@CTA.tech.

Consumer Technology Association
Technology & Standards Department
1919 S Eads Street, Arlington, VA 22202
FAX: (703) 907-7693 standards@CTA.tech

Consumer Technology Association™