Put a UUID On It!

Ben Ramsey

php[tek] • 18 May 2023


UUIDs
We put bids on things.

pad0002 I S

```
$ wd /etc
$ ld hosts

hosts

1 entry listed.
$ cpscr //grault/dm
```

/etc/hosts R 1

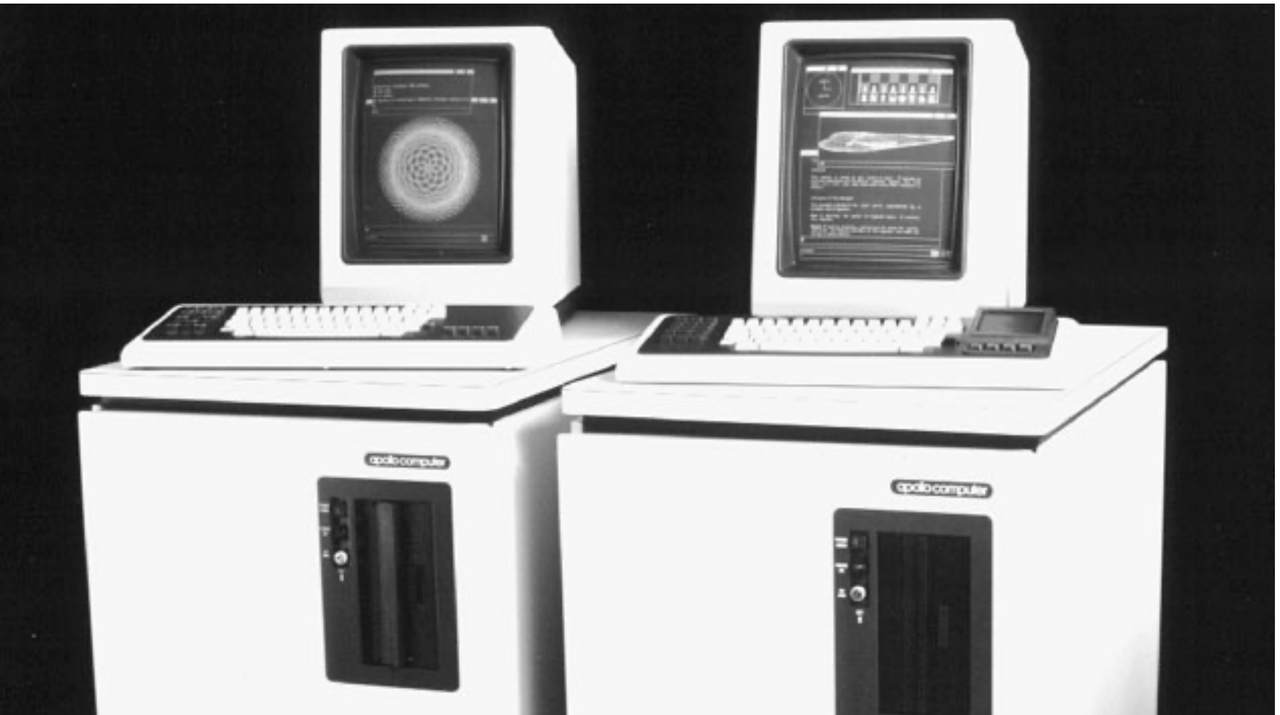
```
127.0.0.1          localhost
255.255.255.0      defaultmask
128.95.142.250     grault.zed  grault
```

pad0001

```
Apollo Domain/OS  Version SR10.4

Copyright (c)  Hewlett-Packard Co., 1986-1992
Copyright (c)  University of CA., 1980, 1985, 1986, 1987, 1988
Copyright (c)  UNIX System Laboratories, Inc., 1980, 1984-1988

                RESTRICTED RIGHTS LEGEND
Use, duplication, or disclosure by the U.S. Government is subject to
restrictions as set forth in subparagraph (c) (1) (ii) of the Rights
in Technical Data and Computer Software clause in DFARS 252.227-7013.
Rights for non-DOD U.S. Government Departments and Agencies are as
set forth in FAR 52.227-19(c) (1,2).
```



pad0002

pad0000

```
dir   nil          1      1024  P   prwx-   pine3.8
dir   nil          1      1024  P   prw--   pio
dir   nil          1      1024  P   prw--   pio-
file  unstruct     1       754  P   prw--   piodrv.
dir   nil          6      6144  P   prwx-   plot
file  unstruct    10     9728  P   prw--   pms.c
dir   nil          1      1024  P   prwx-   proxy-x
dir   nil          6      6144  P   prwx-   ras
file  coff         4      2048  P   prwx-   rcvbits
file  unstruct     5      3873  P   prwx-   rees-us
file  coff         7      4319  P   prwx-   rexecd
file  unstruct     1       597  P   prw--   rmail.c
file  unstruct     1      3873  P   prw--   rx.c-3.
file  unstruct     1       597  P   prw--   s
dir   nil          1      1024  P   prwx-   s48
file  unstruct     1       597  P   prw--   scheme
file  unstruct     1      3873  P   prw--   see.el
file  coff         1      4319  P   prwx-   sendbit
file  unstruct     1       597  P   prw--   sketch.
dir   nil          1      1024  P   prwx-   src
dir   nil          1      1024  P   prwx-   star
dir   nil          1      1024  P   prwx-   stbd
dir   nil          1      1024  P   prwx-   sys
file  unstruct     1       176  P   prwx-   test
dir   nil          1      1024  P   prwx-   tmp
link                                        u
dir   nil          1      1024  P   prwx-   ubud
file  coff       140    142070  P   prwx-   ud-3
link                                        umich
file  unstruct    27     27648  P   prw--   unix2dc
dir   nil          1      1024  P   prwx-   user_da
dir   nil          1      1024  P   prwx-   uw
dir   nil          6      6144  P   prwx-   uw-ras
dir   nil          1      1024  P   prwx-   v2_sr9.
file  unstruct     1       660  P   prw--   x.bak
file  unstruct     1       515  P   prw--   x0
file  unstruct     4      3873  P   prw--   xx

121 entries, 2842 blocks used.
% //dabo/com/cpscr x
% ls -lT x
rec        -rwxr-xr-x   1 rees       101544 Jul 19 23:10 x
%
```

I

universally unique identifier

U U ID

universally unique identifier

universally unique

There are 16^{32} possible UUIDs

“only after generating 1 billion UUIDs every second for the next 100 years, the probability of creating just one duplicate would be about 50%”

— Wikipedia, UUID

unsigned 128-bit integer

`f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

unsigned 128-bit integer

329 800 735 698 586 629 295 641 978 511 506 172 918

unsigned 128-bit integer

```
11111000 00011101 01001111 10101110
01111101 11101100 00010001 11010000
10100111 01100101 00000000 10100000
11001001 00011110 01101011 11110110
```

unsigned 128-bit integer

```
b"ø\x1DO®}ì\x11Ð§e\0 É\x1Eköö"
```

unsigned 128-bit integer

```
ffffffff-ffff-ffff-ffff-ffffffffffff
```

unsigned 128-bit integer

340 282 366 920 938 463 463 374 607 431 768 211 455

unsigned 128-bit integer

340 282 366 920 938 463 463 374 607 431 768 211 455

9 223 372 036 854 775 807

Signed 64-bit max integer

ramsey/uuid

ramsey/uuid

```
composer require ramsey/uuid
```

UUID Format

329 800 735 698 586 629 295 641 978 511 506 172 918

11111000 00011101 01001111 10101110
01111101 11101100 00010001 11010000
10100111 01100101 00000000 10100000
11001001 00011110 01101011 11110110

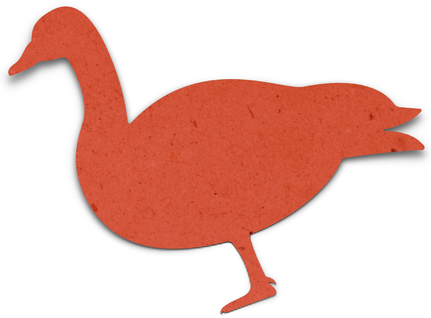
f81d4fae-7dec-11d0-a765-00a0c91e6bf6

variant

f81d4fae-7dec-11d0-a765-00a0c91e6bf6

version

f81d4fae-7dec-11d0-a765-00a0c91e6bf6



```
00000000-0000-1000-8000-000000000000
00000000-0000-2000-9000-000000000000
00000000-0000-3000-a000-000000000000
00000000-0000-4000-b000-000000000000
00000000-0000-5000-8000-000000000000
00000000-0000-6000-9000-000000000000
00000000-0000-7000-a000-000000000000
00000000-0000-8000-b000-000000000000
00000000-0000-9000-8000-000000000000
00000000-0000-a000-9000-000000000000
00000000-0000-b000-a000-000000000000
00000000-0000-c000-b000-000000000000
00000000-0000-d000-8000-000000000000
00000000-0000-e000-9000-000000000000
00000000-0000-f000-a000-000000000000
```

UUID Layouts

UUID Version 1

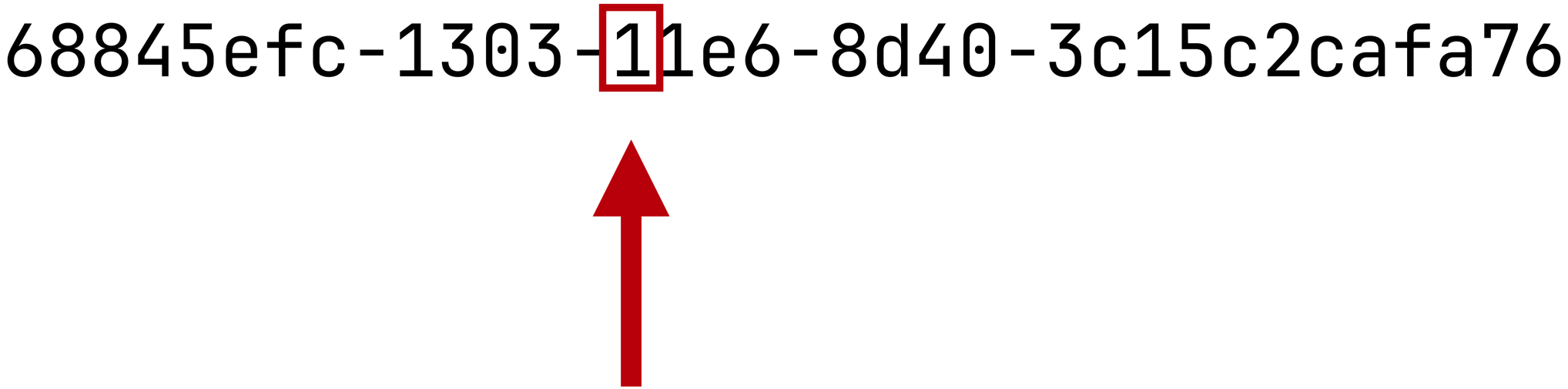
Gregorian Time

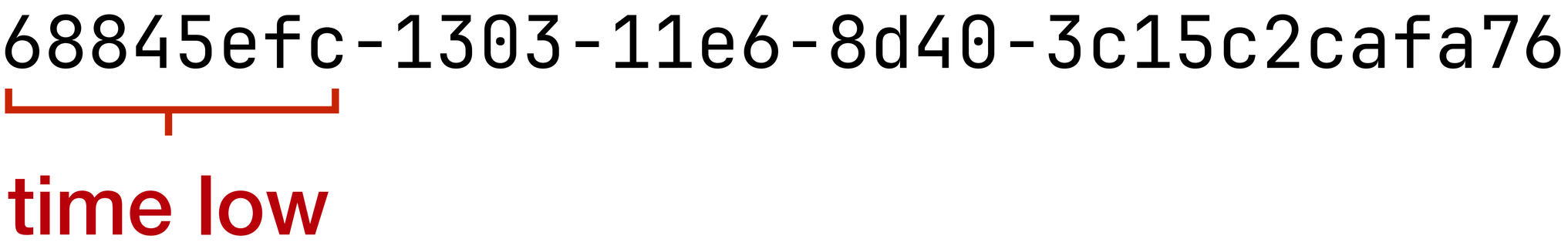
68845efc-1303-11e6-8d40-3c15c2cafa76

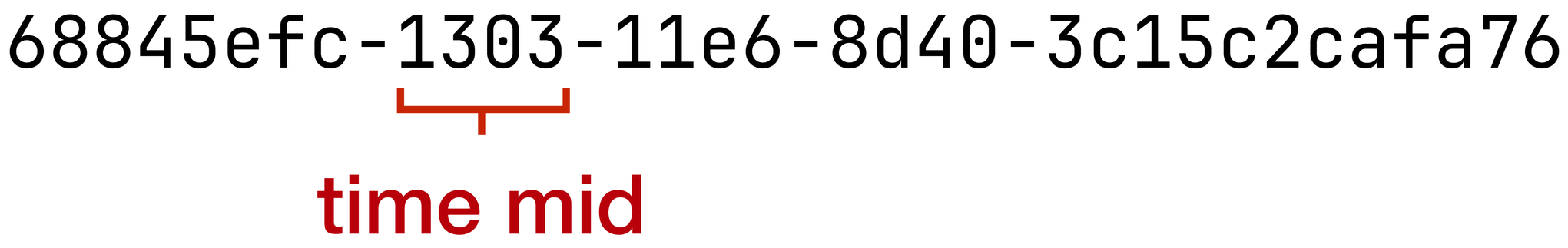
variant

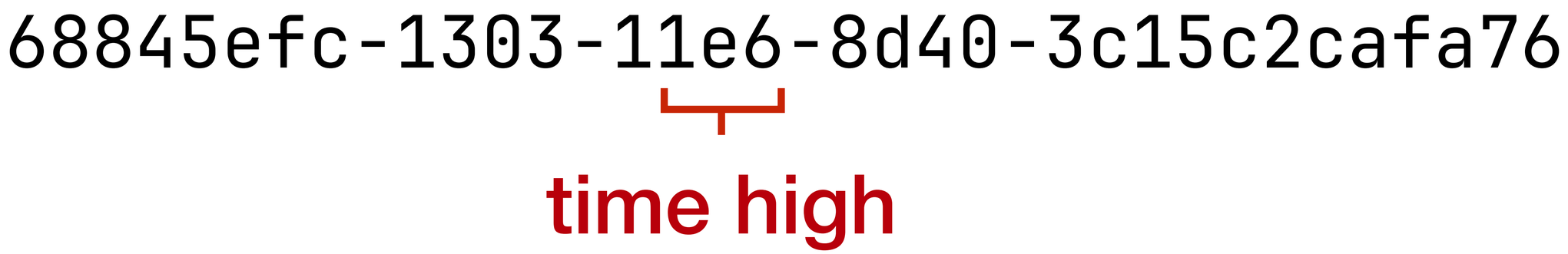
68845efc-1303-11e6-8d40-3c15c2cafa76

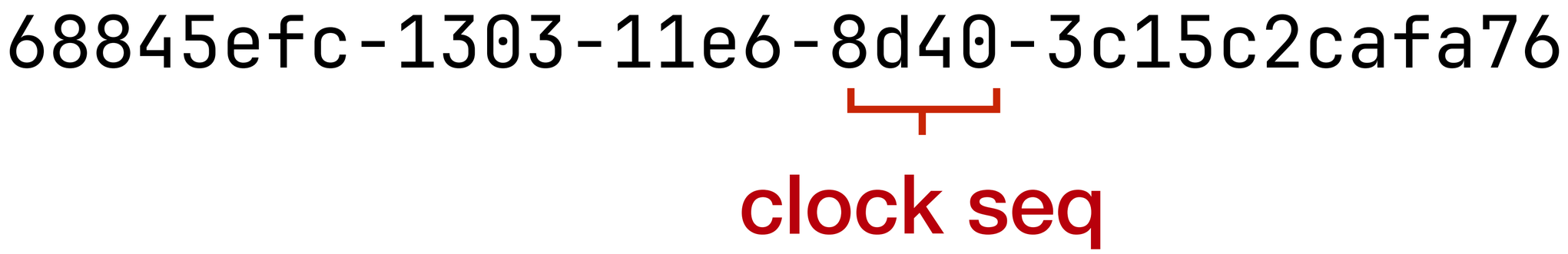
version

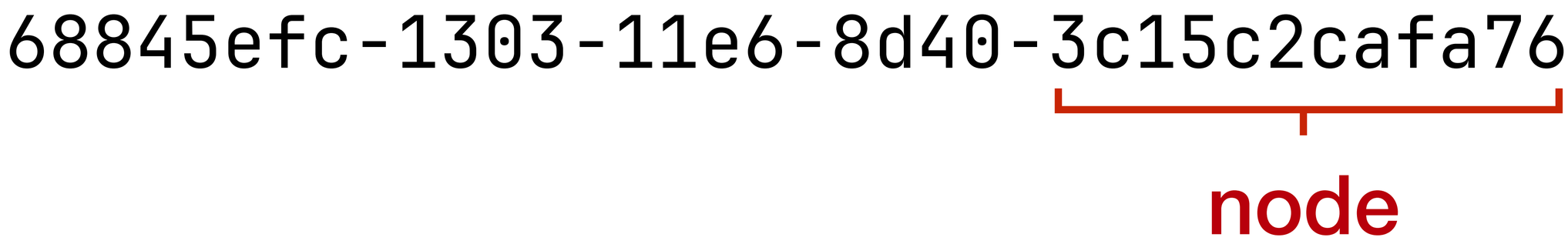


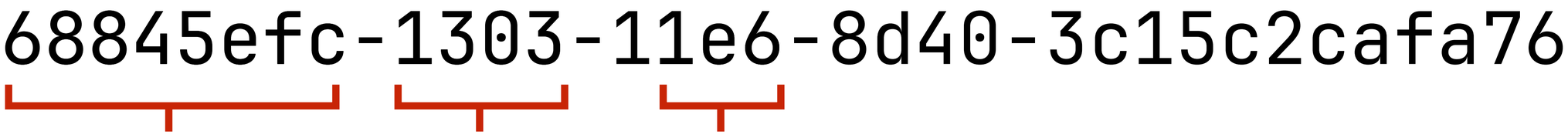

68845efc-1303-11e6-8d40-3c15c2cafa76
time low


68845efc-1303-11e6-8d40-3c15c2cafa76
time mid


68845efc-1303-11e6-8d40-3c15c2cafa76
time high


68845efc-1303-11e6-8d40-3c15c2cafa76
clock seq


68845efc-1303-11e6-8d40-3c15c2cafa76
node


68845efc-1303-11e6-8d40-3c15c2cafa76


68845efc 1303 1e6

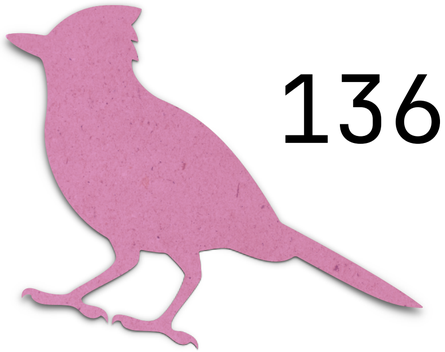
1e6130368845efc

1e6130368845efc

100-nanosecond intervals
since October 15, 1582

136 817 744 040 713 980

100-nanosecond intervals
since October 15, 1582



136 817 744 040 713 980

122 192 928 000 000 000

100-nanosecond intervals
between October 15, 1582
and January 1, 1970

```
$ts = 136 817 744 040 713 980
```

```
U_DIFF = 122 192 928 000 000 000
```

100-nanosecond intervals
between October 15, 1582
and January 1, 1970

```php
ini_set('precision', 16);

const I_SEC = 10_000_000;
const U_DIFF = 122_192_928_000_000_000;
$ts = 136_817_744_040_713_980;

$time = ($ts - U_DIFF) / I_SEC;
[$sec, $usec] = explode('.', $time, 2);
$usec = str_pad($usec, 6, '0');
```

```php
$date = new DateTimeImmutable(
    "@$sec.$usec"
);

echo $date->format(
    DateTimeImmutable::RFC3339_EXTENDED,
);
```

2016-05-05T20:53:24.071+00:00

```php
use Ramsey\Uuid\Uuid;

$uuid = Uuid::uuid1();

echo $uuid->getDateTime()->format(
    DateTimeImmutable::RFC3339_EXTENDED,
);
```

Caveats

Gregorian Time, UUID version 1

Leaks MAC address for the machine's network controller*

14-bit clock sequence allows 16,383 unique UUIDs every microsecond

Weird layout means they're not sortable

There are better options

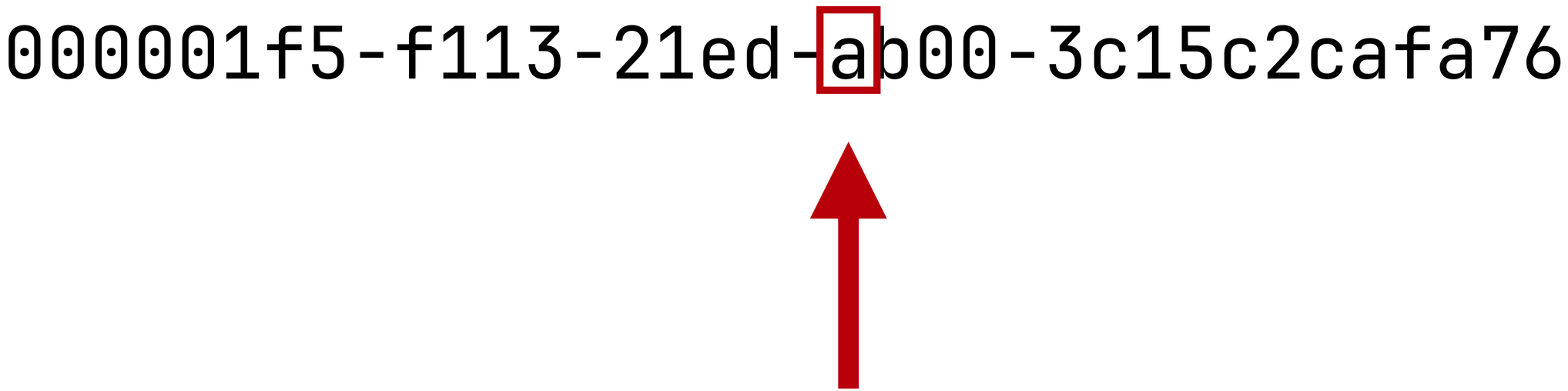
UUID Version 2

DCE Security

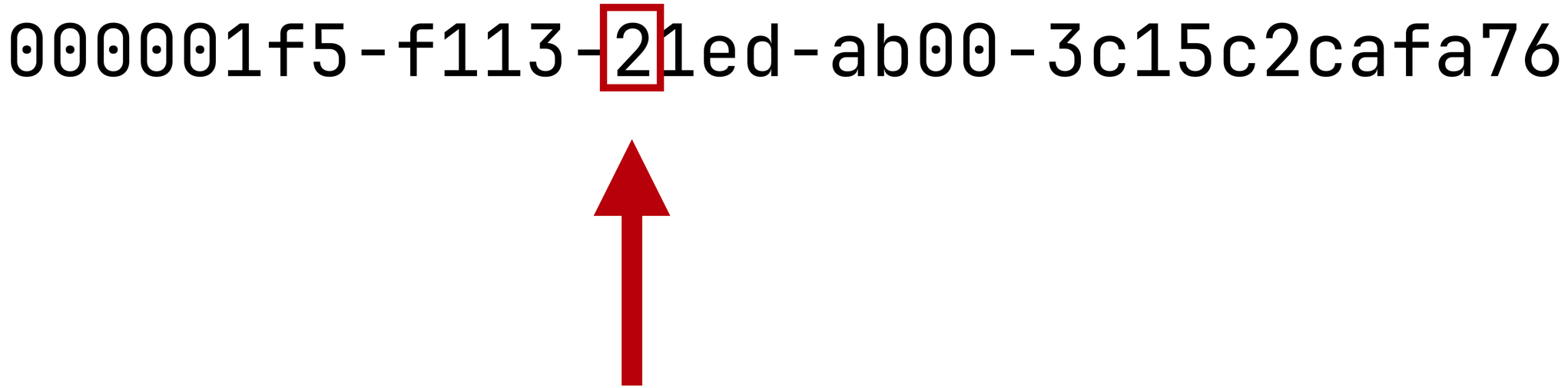
```php
$uuid = Uuid::uuid2(
    Uuid::DCE_DOMAIN_PERSON,
);
```

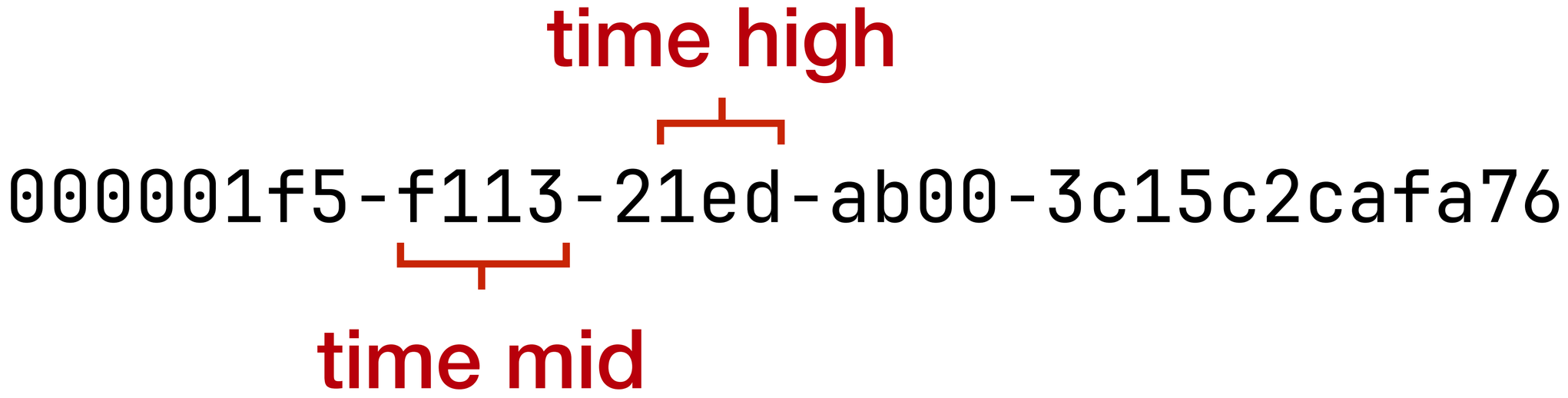
000001f5-f113-21ed-ab00-3c15c2cafa76

variant



version




time high
000001f5-f113-21ed-ab00-3c15c2cafa76
time mid


000001f5-f113-21ed-ab00-3c15c2cafa76
local ID
domain

Warning

DCE Security, UUID version 2

Time can be off by up to 7 minutes, 9 seconds, and 496,730 microseconds (i.e., $2^{32} - 1$)

Leaks local identifiers (UID, GID, etc.)

Leaks MAC address for the machine's network controller*

Chance for collisions since there's only a 6-bit clock sequence

Don't use unless working with DCE

UUID
Versions 3 & 5

Name-based (MD5 & SHA-1)

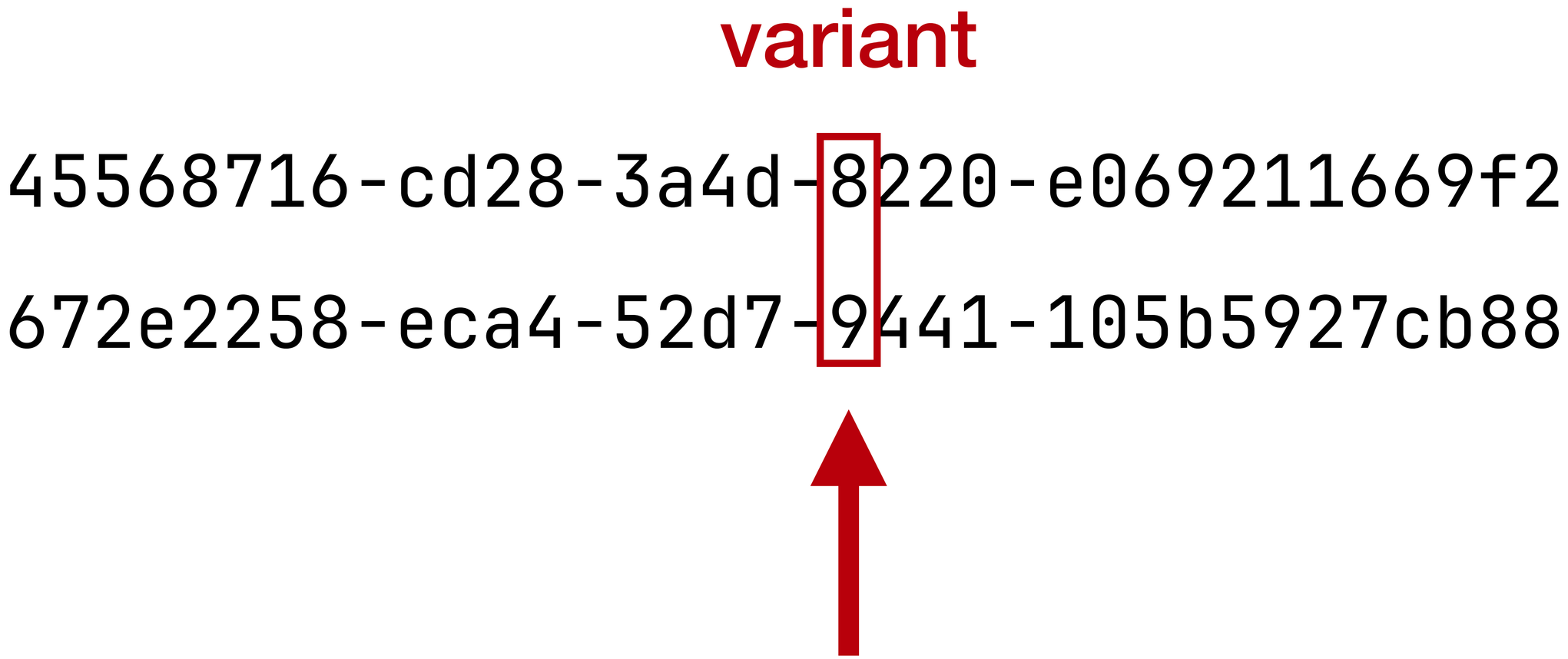
```php
$uuid = Uuid::uuid3(
    Uuid::NAMESPACE_URL,
    'https://tek.phparch.com/',
);
```

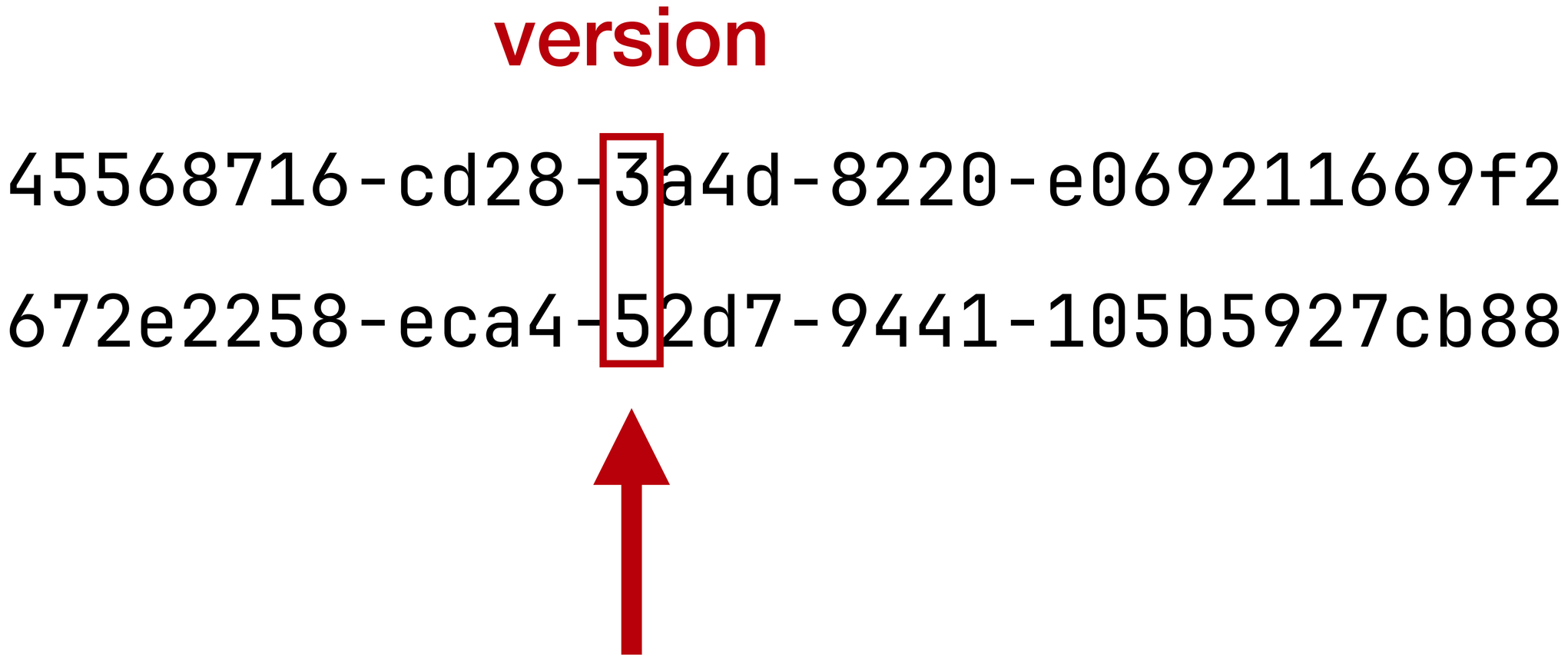
```php
$uuid = Uuid::uuid3(
    Uuid::NAMESPACE_URL,
    'https://tek.phparch.com/',
);

$uuid = Uuid::uuid5(
    Uuid::NAMESPACE_URL,
    'https://tek.phparch.com/',
);
```

45568716-cd28-3a4d-8220-e069211669f2

672e2258-eca4-52d7-9441-105b5927cb88


variant
45568716-cd28-3a4d-8220-e069211669f2
672e2258-eca4-52d7-9441-105b5927cb88


version
45568716-cd28-3a4d-8220-e069211669f2
672e2258-eca4-52d7-9441-105b5927cb88

45568716-cd28-3a4d-8220-e069211669f2

672e2258-eca4-52d7-9441-105b5927cb88

md5/sha-1
high

45568716-cd28-3a4d-8220-e069211669f2

672e2258-eca4-52d7-9441-105b5927cb88

md5/sha-1

mid

45568716-cd28-3a4d-8220-e069211669f2

672e2258-eca4-52d7-9441-105b5927cb88

md5/sha-1
low

Take Note

Name-based (MD5, SHA-1), UUID versions 3 & 5

Used for generating UUIDs from *names* that are unique within a *name space*

The same name and name space will always produce the same UUID

Prefer the use of SHA-1 (version 5) over MD5 (version 3)

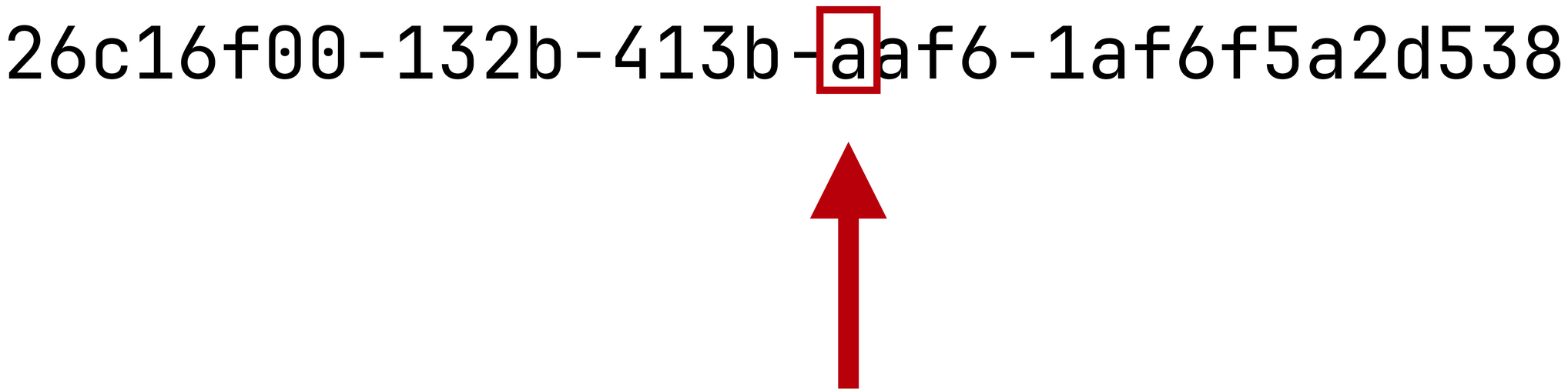
UUID Version 4

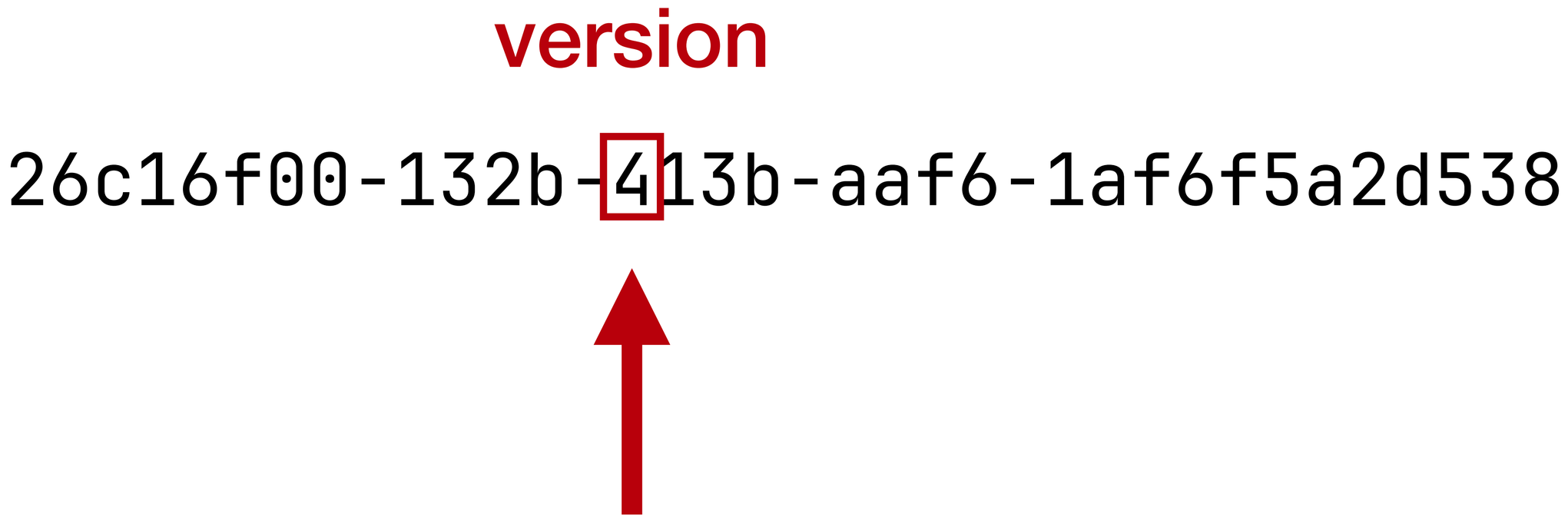
Random

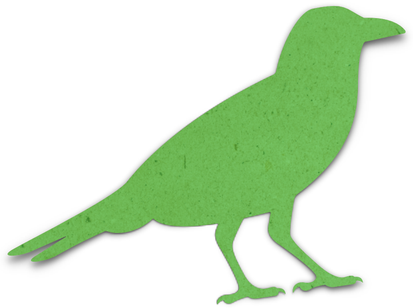
```php
$uuid = Uuid::uuid4();
```

26c16f00-132b-413b-aaf6-1af6f5a2d538

variant




version
26c16f00-132b-413b-aaf6-1af6f5a2d538



26c16f00-132b-413b-aafb-1af6f5a2d538

random a random b random c

Thoughts

Random, UUID version 4

Most commonly used

May be the most unique, with 122 bits of randomness

This is probably the one you want to use*

On the Horizon

draft-ietf-uuidrev-rfc4122bis-03

UUID Version 6

Reordered Time

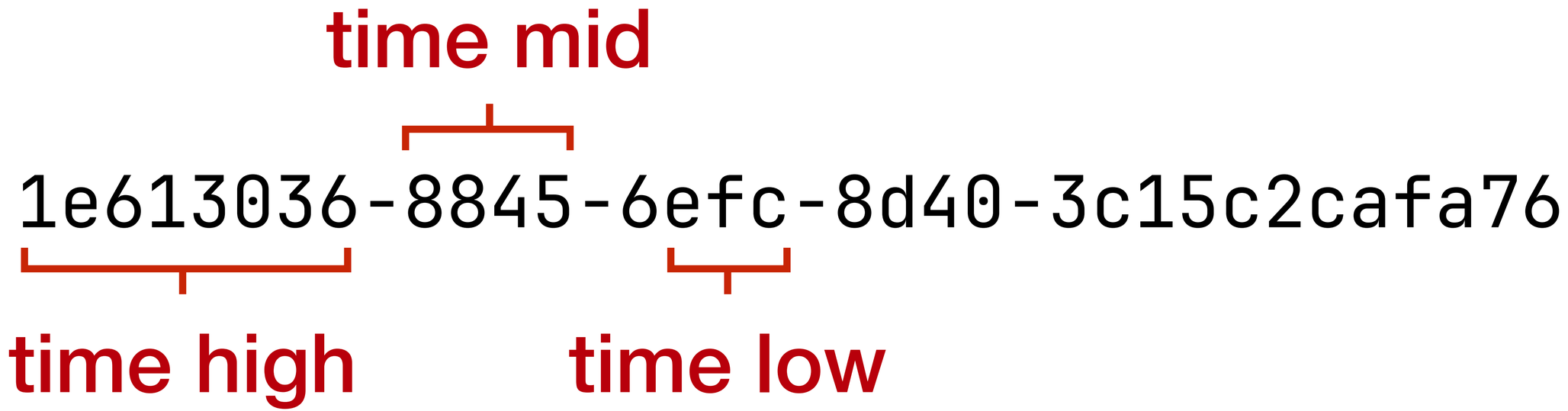
1e613036-8845-6efc-8d40-3c15c2cafa76

variant

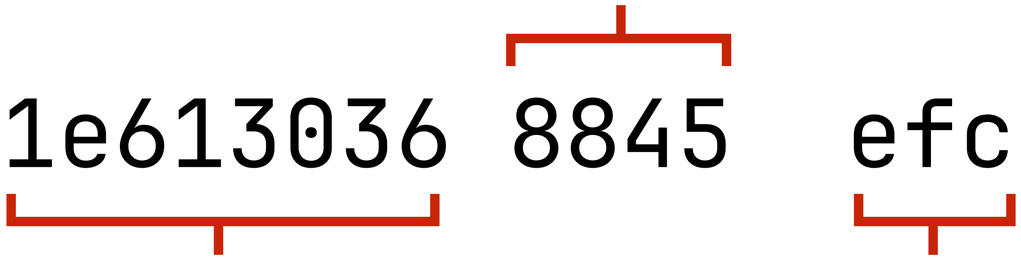
1e613036-8845-6efc-8d40-3c15c2cafa76

version

1e613036-8845-6efc-8d40-3c15c2cafa76


time mid
1e613036-8845-6efc-8d40-3c15c2cafa76
time high
time low


1e613036-8845-6efc-8d40-3c15c2cafa76


1e613036 8845 efc

1e6130368845efc

```php
$uuid = Uuid::uuid6();

echo $uuid->getDateTime()->format(
    DateTimeImmutable::RFC3339_EXTENDED,
);
```

Details

Reordered Time, UUID version 6

Gregorian time, just like UUID version 1

Time fields are reordered for proper sorting

Able to convert between version 1 and 6

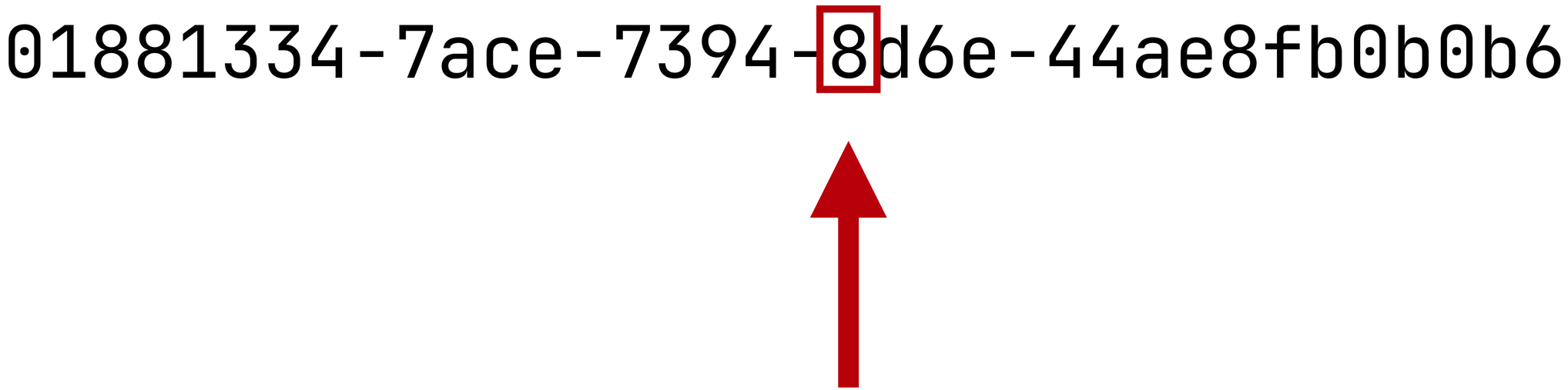
Equivalent to the (now deprecated) OrderedTimeCodec in ramsey/uuid

UUID Version 7

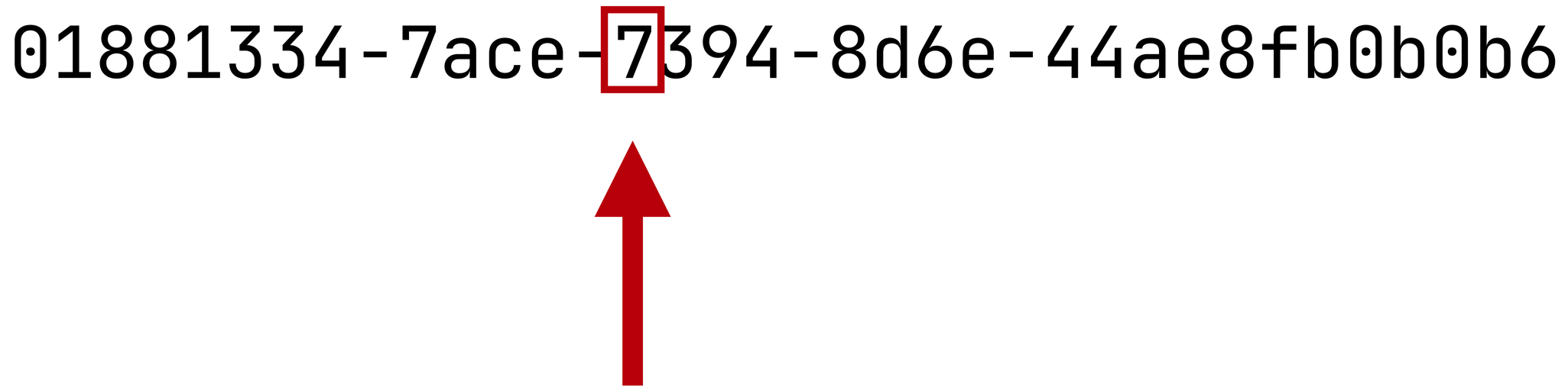
Unix Epoch Time

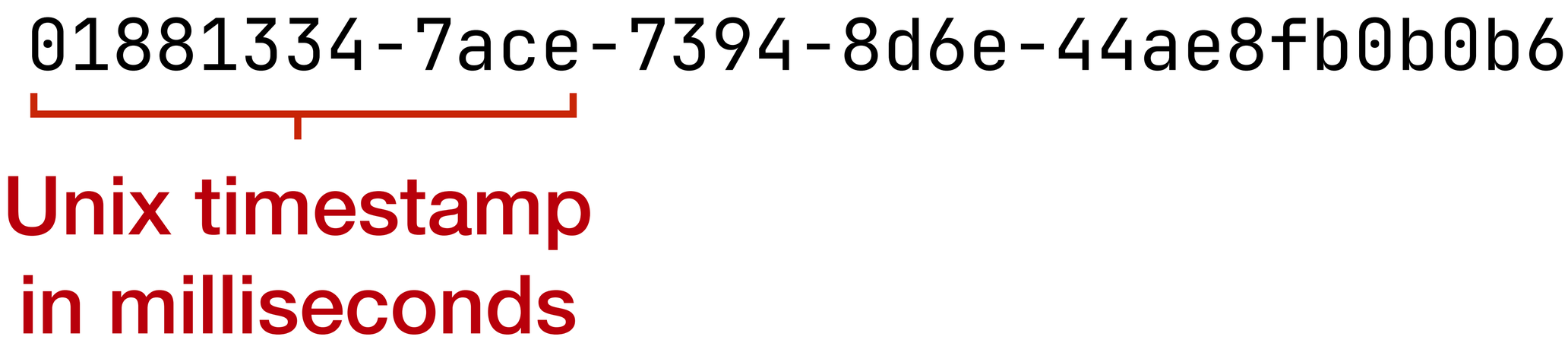
01881334-7ace-7394-8d6e-44ae8fb0b0b6

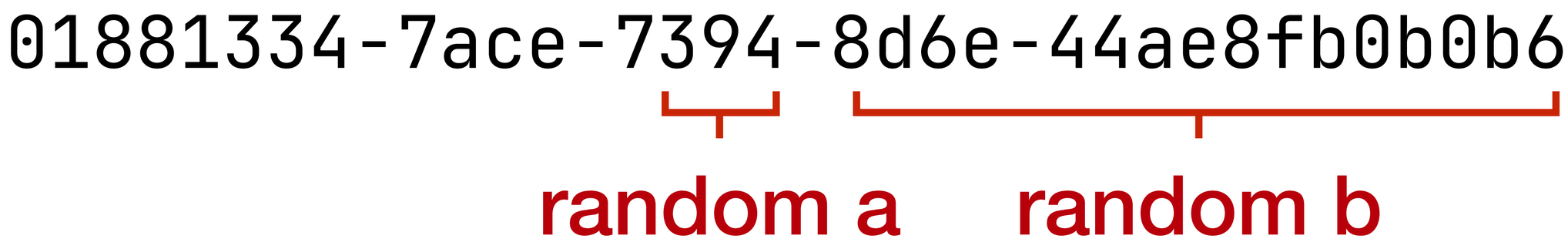
variant

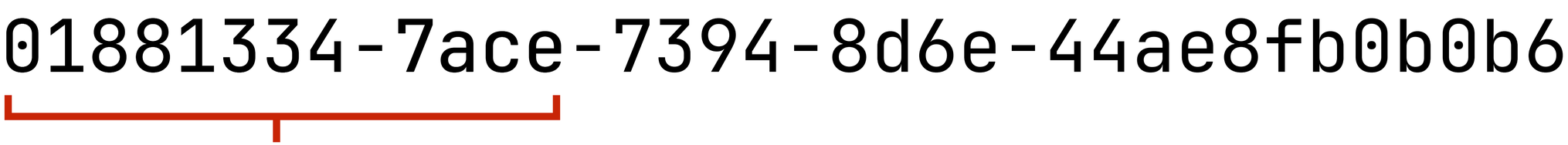


version




01881334-7ace-7394-8d6e-44ae8fb0b0b6
Unix timestamp
in milliseconds


01881334-7ace-7394-8d6e-44ae8fb0b0b6
random a
random b


01881334-7ace-7394-8d6e-44ae8fb0b0b6


01881334 7ace

018813347ace

018813347ace

milliseconds
since January 1, 1970

1 683 949 386 446

milliseconds
since January 1, 1970

```
$ts = 1 683 949 386 446
```

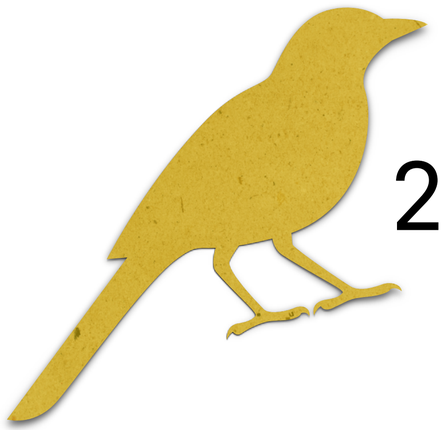
milliseconds
since January 1, 1970

```php
const MS = 1000;
$ts = 1_683_949_386_446;

$time = $ts / MS;
[$sec, $usec] = explode('.', $time, 2);
$usec = str_pad($usec, 6, '0');
```

```php
$date = new DateTimeImmutable(
    "@$sec.$usec"
);

echo $date->format(
    DateTimeImmutable::RFC3339_EXTENDED,
);
```



2023-05-13T03:43:06.446+00:00

```php
$uuid = Uuid::uuid7();

echo $uuid->getDateTime()->format(
    DateTimeImmutable::RFC3339_EXTENDED,
);
```

Info

Unix Epoch Time, UUID version 7

Uses Unix Epoch timestamp in milliseconds

Monotonically increasing; sortable

Use this instead of UUID version 1 or version 6

Equivalent to the (now deprecated) TimestampFirstCombCodec in ramsey/uuid

Binary compatible with ULIDs

UUID Version 8

Custom

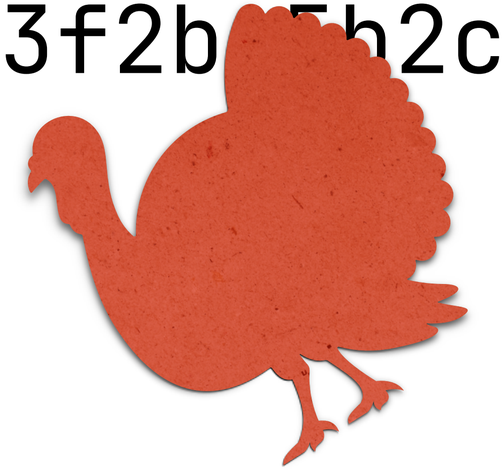
401835fd-a627-870a-873f-ed73f2bc5b2c

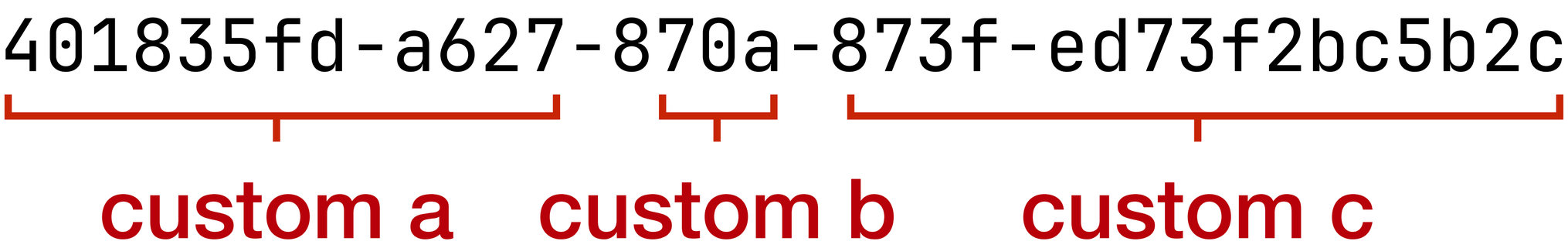
variant

401835fd-a627-870a-873f-ed73f2bc5b2c

version

401835fd-a627-870a-873f-ed73f2b[illegible]2c




401835fd-a627-870a-873f-ed73f2bc5b2c
custom a
custom b
custom c

```php
$uuid = Uuid::uuid8(
    "\x40\x18\x35\xfd\xa6\x27\x07\x0a"
    ."\x07\x3f\xed\x73\xf2\xbc\x5b\x2c",
);
```

Considerations

Custom, UUID version 8

Format is wide open; very flexible

The format is custom to your use; no one else will know what your format means

Uniqueness is implementation-specific and must not be assumed

ULIDs?

01881334-7ace-7394-8d6e-44ae8fb0b0b6

01881334-7ace-7394-8d6e-44ae8fb0b0b6

01H09K8YPEEEA8TVJ4NT7V1C5P

ULIDs

Universally Unique Lexicographically Sortable Identifier

Binary-compatible with UUID version 7 (i.e., the bytes are the same)

Uses Crockford's Base 32 encoding for string representation

Not supported in ramsey/uuid (yet?)


Databases

Database Issues

Some databases have UUID as a native type

InnoDB uses PK in all secondary keys

- If the PK is big (i.e., `CHAR(36)`), then all keys will be huge

InnoDB stores data in PK order

- If PK is a UUID and not sequential, inserts are scattered on disk

Database Solutions

Don’t use UUID as a primary key

Use binary instead of string UUID, `CHAR(16)`

Use UUID version 6 or version 7, if using as PK

If needing to sort on UUIDs, use version 6 or 7

Prefer UUID version 7

Read Percona blog post, “Storing UUID Values in MySQL”


Should You
Put a UUID
On It?

Cons

Takes up more DB space than integer

Some UUIDs are not sequential

Could decrease database performance

Might not be able to `ORDER BY`

Look ugly in URLs; not memorable

Pros

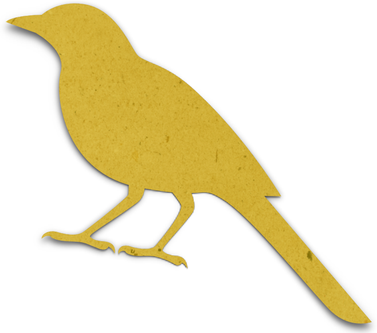
Unique everywhere

Easy to merge records from multiple sources

Easy to distribute databases across multiple servers (i.e., sharding)

Able to generate anywhere, independent of central authority

Thanks!



joind.in/talk/11af7

phpc.social/@ramsey

github.com/ramsey

ben@benramsey.com

1. Boris Smokrovic, *Blue Kingfisher*, photograph, Unsplash, February 6, 2017, https://unsplash.com/photos/DPXytK8Z59Y.
2. *Portlandia*, season 1, episode 2, “A Song for Portland,” directed by Jonathan Krisel, written by Fred Armisen, Carrie Brownstein, and Jonathan Krisel, featuring Fred Armisen and Carrie Brownstein, aired January 28, 2011, IFC, Broadway Video Television, 2011.
3. milatchi, “Re: Apollo Domain/OS,” *BetaArchive* (forum), April 28, 2012, https://www.betaarchive.com/forum/viewtopic.php?p=293580&sid=961354fb440da3fe47cbd2b52f4aff8e#p293580.
4. Stein-Mason Studio, Inc., *Apollo Domain Workstation*, photograph, Computer History Museum, 1982, https://www.computerhistory.org/collections/catalog/102626960.
5. Fredrik Rubensson, *thinker*, photograph, Flickr, December 18, 2012, https://www.flickr.com/photos/froderik/8283727226/.
6. Cathal Mac an Bheatha, *Yellow bird painting*, photograph, Unsplash, March 10, 2017, https://unsplash.com/photos/YaE8m2Oj36I.
7. Boris Smokrovic, Untitled, photograph, Unsplash, September 1, 2016, https://unsplash.com/photos/FGthCBTIFSk.
8. Daniela Popescu, Untitled, photograph, Unsplash, May 26, 2019, https://unsplash.com/photos/-zF8t2kjjig.
9. Arno Senoner, *A mural bird or owl in Athens by the artist Fotizontas below the Acropolis*, photograph, Unsplash, July 6, 2021, https://unsplash.com/photos/6A2wQmIwpsw.
10. *Encanto*, directed by Jared Bush and Byron Howard, (Walt Disney Pictures, 2021).
11. Rohit Tandon, Untitled, photograph, Unsplash, April 8, 2022, https://unsplash.com/photos/J3oIr1B1VZU.
12. Deepak Nautiyal, *Plumbeous Water Redstart in Dehradun*, photograph, Unsplash, November 18, 2019, https://unsplash.com/photos/KN-tHM5ulF8.
13. Etienne Girardet, *roman statue in blue sky, visited by an italian seagull*, photograph, Unsplash, December 12, 2020, https://unsplash.com/photos/ALIEN0anXKg.
14. Annie Spratt, *Plants on a wood table*, photograph, Unsplash, September, 23, 2016, https://unsplash.com/photos/67L7k6dlZVA.
15. Daria Volkova, Untitled, photograph, Unsplash, August 11, 2019, https://unsplash.com/photos/NiAwzcqF0cM.
16. Justin Wilkens, *Metal bird greets nesting birds*, photograph, Unsplash, June 1, 2022, https://unsplash.com/photos/np5EeoRRit0.
17. Maksym Kaharlytskyi, Untitled, photograph, Unsplash, August 12, 2022, https://unsplash.com/photos/mSFFiAz2zY4.
18. Kevin Lanceplaine, *Origami*, photograph, Unsplash, July 5, 2020, https://unsplash.com/photos/oRmeOaeVfXo.
19. Jojo Yuen, Untitled, photograph, Unsplash, April 4, 2021, https://unsplash.com/photos/TQiGm5fQw0Y.