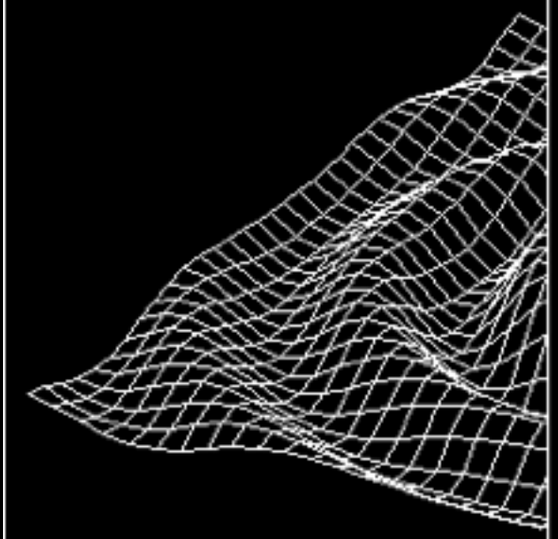
Identify All the Things With UUIDs

Ben Ramsey
True North PHP
4 Nov 2016

pad0002

pad0001



pad0000 I S

```
dir   nil          1      1024  P    prwx-       pine3.89
dir   nil          1      1024  P    prwx-       pio
dir   nil          1      1024  P    prwx-       pio-
file  unstruct     1       754  P    prwx-       piodrv.make_ddf
dir   nil          6      6144  P    prwx-       plot
file  unstruct    10      9779  P    prw--       pms.c
dir   nil          1      1024  P    prwx-       proxy-xrn
dir   nil          6      6144  P    prwx-       ras
file  coff         4      4075  P    prwx-       rcvbits
file  unstruct     5      4719  P    prwx-       rees-usenix-48.pbm
file  coff         7      6669  P    -r-x-       rexecd
file  unstruct    19     19046  P    prw--       rmail.ml
file  unstruct   143    144836  P    pr---       rx.c-3.2
file  unstruct     1        95  P    prwx-       s
dir   nil          6      6144  P    prwx-       s48
file  unstruct  1024   1042726  P    prw--       scheme48-0.36.tar.gz
file  unstruct     1       373  P    prw--       see.el
file  coff         5      4319  P    prwx-       sendbits
file  unstruct     1       597  P    prw--       sketch.pl
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--       unix2dos.tar
dir   nil          1      1024  P    prwx-       user_data
dir   nil          1      1024  P    prwx-       uw
dir   nil          6      6144  P    prwx-       uw-ras
dir   nil          1      1024  P    prwx-       v2_sr9.2
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
```

%

Command: I

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

379dae82-5a2b-4c4b-8193-b8e7749a3495

Unsigned 128-bit integer

73 926 269 841 798 498 179 266 328 839 241 479 317

Unsigned 128-bit integer

b"7Ø«éZ+LKüô©þtÜ4ò"

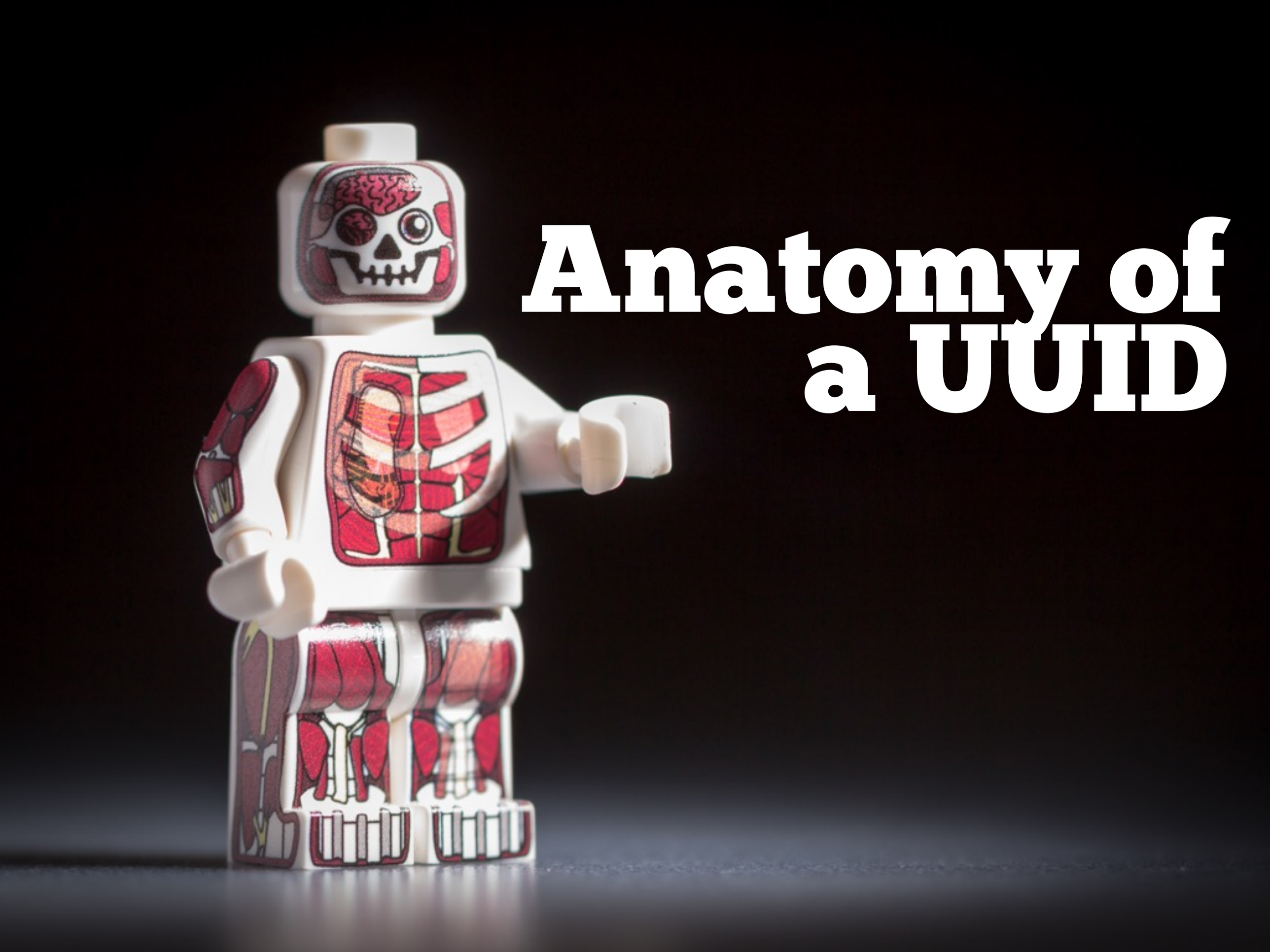
ramsey/uuid

composer require ramsey/uuid

```php
use Ramsey\Uuid\Uuid;
```

RFC 4122

Defines a specific variant of UUID with five versions.


Anatomy of
a UUID

379dae82-5a2b-4c4b-8193-b8e7749a3495

379dae82-5a2b-4c4b-8193-b8e7749a3495

time low

379dae82-5a2b-4c4b-8193-b8e7749a3495

time mid

379dae82-5a2b-4c4b-8193-b8e7749a3495

time high & version

UUID version

379dae82-5a2b-4c4b-8193-b8e7749a3495

time high & version

379dae82-5a2b-4c4b-8193-b8e7749a3495

clock sequence high & reserved

UUID variant

379dae82-5a2b-4c4b-8193-b8e7749a3495

clock sequence high & reserved

379dae82-5a2b-4c4b-8193-b8e7749a3495

clock sequence low

379dae82-5a2b-4c4b-8193-b8e7749a3495

node

Version 1: Time Based

```
$uuid = Uuid::uuid1();
echo $uuid->toString();
```

```php
for ($i = 0; $i < 5; $i++) {
    echo Uuid::uuid1()->toString();
    echo "\n";
}
```

68845efc-1303-11e6-8d40-3c15c2cafa76
68846d0c-1303-11e6-8088-3c15c2cafa76
68847216-1303-11e6-830b-3c15c2cafa76
68847590-1303-11e6-892a-3c15c2cafa76
68847806-1303-11e6-913b-3c15c2cafa76

Version 1

```
68845efc-1303-11e6-8d40-3c15c2cafa76
68846d0c-1303-11e6-8088-3c15c2cafa76
68847216-1303-11e6-830b-3c15c2cafa76
68847590-1303-11e6-892a-3c15c2cafa76
68847806-1303-11e6-913b-3c15c2cafa76
```

Version 1

```
68845efc-1303-11e6-8d40-3c15c2cafa76
68846d0c-1303-11e6-8088-3c15c2cafa76
68847216-1303-11e6-830b-3c15c2cafa76
68847590-1303-11e6-892a-3c15c2cafa76
68847806-1303-11e6-913b-3c15c2cafa76
```

RFC 4122

Version 4: Random

```php
for ($i = 0; $i < 5; $i++) {
    echo Uuid::uuid4()->toString();
    echo "\n";
}
```

28b7883f-3df0-4e8a-9994-57b28f01fab2
dd78278c-03d7-4ca4-8411-06380c4cfd57
ae8de9f1-6ffc-454e-8522-b2c260e4c66b
a5e9ac6b-0680-4fb7-a019-f305c7d2d5da
09fd4b15-2b2d-49be-af51-30899169ff99

Version 4

```
28b7883f-3df0-4e8a-9994-57b28f01fab2
dd78278c-03d7-4ca4-8411-06380c4cfd57
ae8de9f1-6ffc-454e-8522-b2c260e4c66b
a5e9ac6b-0680-4fb7-a019-f305c7d2d5da
09fd4b15-2b2d-49be-af51-30899169ff99
```

Version 4

```
28b7883f-3df0-4e8a-9994-57b28f01fab2
dd78278c-03d7-4ca4-8411-06380c4cfd57
ae8de9f1-6ffc-454e-8522-b2c260e4c66b
a5e9ac6b-0680-4fb7-a019-f305c7d2d5da
09fd4b15-2b2d-49be-af51-30899169ff99
```

RFC 4122

Name Based: Version 3 & 5

```php
$uuid = Uuid::uuid3(
    Uuid::NAMESPACE_DNS,
    'truenorthphp.ca'
);
```

ec187192-1064-3e19-9a5f-704c6bb9d947

Version 3

ec187192-1064-3e19-9a5f-704c6bb9d947

Version 3

ec187192-1064-3e19-9a5f-704c6bb9d947

RFC 4122

```php
$uuid = Uuid::uuid5(
    Uuid::NAMESPACE_DNS,
    'truenorthphp.ca'
);
```

acfd6b8c-7bea-5767-b488-dc8424eae21a

Version 5

acfd6b8c-7bea-5767-b488-dc8424eae21a

Version 5

acfd6b8c-7bea-5767-b488-dc8424eae21a

RFC 4122

```php
$dpcNS = Uuid::uuid5(
    Uuid::NAMESPACE_DNS,
    'truenorthphp.ca'
);

define(
    'NAMESPACE_TN',
    $dpcNS->toString()
);

$pageId = Uuid::uuid5(
    NAMESPACE_TN,
    'sponsors.php'
);
```

Version 2?

Version 2: DCE Security

Advanced Topics

COMB

- Combined random UUID/timestamp
- Replaces the node field with the current timestamp
- Tries to compensate for the reduced clustering in database indexes
- Not sequential
- Non-standard

```php
$factory = new UuidFactory();

$generator = new CombGenerator(
    $factory->getRandomGenerator(),
    $factory->getNumberConverter()
);

$factory->setRandomGenerator($generator);

$combUuid = $factory->uuid4();
```

fe4c9653-b201-431e-86ef-866e9fd03eea

Version 4

fe4c9653-b201-431e-86ef-866e9fd03eea

RFC 4122

fe4c9653-b201-431e-86ef-866e9fd03eea

timestamp

Timestamp First COMB

- Combined random UUID/timestamp
- Replaces the node field with time low and time mid fields
- Adds the current timestamp to the time low and time mid fields
- Provides sequential “random” UUIDs
- Non-standard

```php
$factory = new UuidFactory();

$generator = new CombGenerator(
    $factory->getRandomGenerator(),
    $factory->getNumberConverter()
);

$codec = new TimestampFirstCombCodec(
    $factory->getUuidBuilder()
);

$factory->setCodec($codec);
$factory->setRandomGenerator($generator);

$tsFirstCombUuid = $factory->uuid4();
```

```php
$factory = new UuidFactory();

$generator = new CombGenerator(
    $factory->getRandomGenerator(),
    $factory->getNumberConverter()
);

$codec = new TimestampFirstCombCodec(
    $factory->getUuidBuilder()
);

$factory->setCodec($codec);
$factory->setRandomGenerator($generator);

$tsFirstCombUuid = $factory->uuid4();
```

866e9fd0-3eea-431e-86ef-fe4c9653b201

Version 4

866e9fd0-3eea-431e-86ef-fe4c9653b201

RFC 4122

866e9fd0-3eea-431e-86ef-fe4c9653b201

timestamp

Database Considerations

- Some databases have UUID as native type
    - MySQL does not
    - InnoDB uses PK in all secondary keys; if PK is big (`CHAR(36)`), then all keys will be huge
    - InnoDB stores data in PK order; if UUID is PK and not sequential, then inserts are scattered

Database Solutions

- Don’t use UUID as a primary key
- Use binary instead of string UUID
  - `CHAR(16)`
  - `$uuid->getBytes()`
- If used as PK, use `OrderedTimeCodec`
- Read Percona blog post “Store UUID in an optimized way”

Ordered Time UUID

- Version 1 UUID with timestamp re-ordered
- Swaps the time low and time high (and version) fields
- Provides sequential time-based UUIDs
- Non-standard
- Loses access to version

OrderedTimeCodec

- Only re-arranges fields when encoding to a byte string (i.e. `Uuid::getBytes()`)
- When decoding bytes, it puts the fields back in order
  - **Caveat**: Can result in unintended effects if decoding bytes not encoded with this codec

```php
$factory = new UuidFactory();

$codec = new OrderedTimeCodec(
    $factory->getUuidBuilder()
);

$factory->setCodec($codec);

$orderedTimeUuid = $factory->uuid1();
```

bad96ae6-a118-11e6-aa2b-3c15c2cafa76

Version 1

bad96ae6-a118-11e6-aa2b-3c15c2cafa76

RFC 4122

Note how the codec doesn't rearrange the fields when converting to a string.

Version 1

bad96ae6-a118-11e6-aa2b-3c15c2cafa76

RFC 4122

Let's see what happens when we decode the bytes without the codec.

```
$uuid = Uuid::fromBytes(
    $orderedTimeUuid->getBytes()
);

echo $uuid->toString();
```

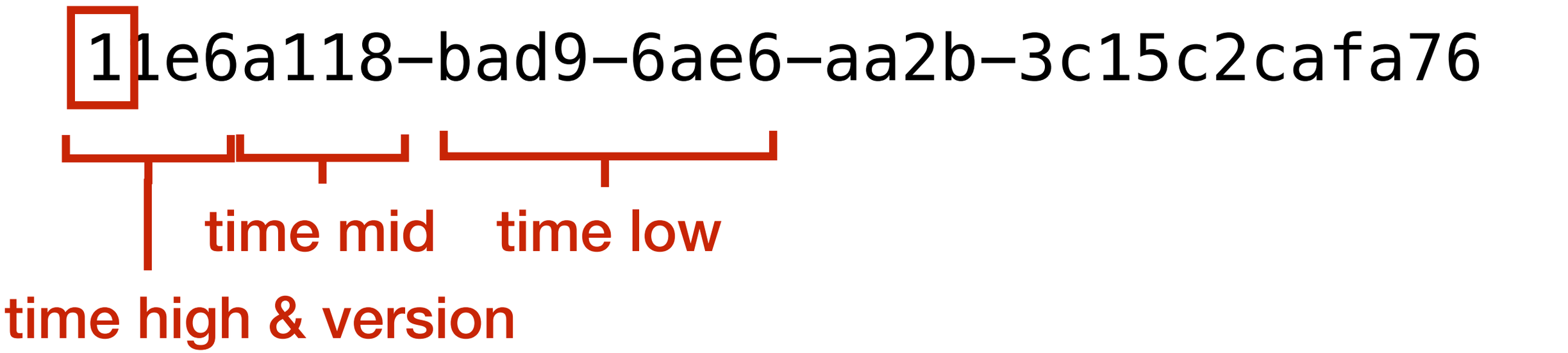
11e6a118-bad9-6ae6-aa2b-3c15c2cafa76

Version ???

11e6a118-bad9-6ae6-aa2b-3c15c2cafa76

RFC 4122

Version 1


11e6a118-bad9-6ae6-aa2b-3c15c2cafa76
time mid
time low
time high & version

Override Time Generator

Want to use **pecl-uuid** as your time generator?

We can do that.

```
$factory = new UuidFactory();

$factory->setTimeGenerator(
    new PeclUuidTimeGenerator()
);

$uuid = $factory->uuid1();
```

Override Random Generator

Want to use **libsodium** as your random generator?

We can do that.

```php
$factory = new UuidFactory();

$factory->setRandomGenerator(
    new SodiumRandomGenerator()
);

$uuid = $factory->uuid4();
```

More Customization

In fact, you're able to customize ramsey/uuid in any way you need by using interfaces and injecting dependencies in the factory:

- Generators
- Codecs
- Converters
- Providers

UUID all the things?

Cons

- Takes up much more DB space than integer
- By default (and spec) not sequential
- Decreases database performance
- Can’t use ORDER BY*
- Ugly in URLs

Pros

- Unique everywhere
- Easily able to merge recordsets from multiple sources
- Easy distribution of databases across multiple servers
- May be generated anywhere, independently

THANK YOU.
ANY QUESTIONS?

If you want to talk more, feel free to contact me.

joind.in/talk/56aa0

benramsey.com

@ramsey

github.com/ramsey

ben@benramsey.com

This presentation was created using Keynote. The text is set in Chunk Five and Helvetica Neue. The source code is set in Menlo. The iconography is provided by Font Awesome.

Unless otherwise noted, all photographs are used by permission under a Creative Commons license. Please refer to the Photo Credits slide for more information.

Identify All the Things With UUIDs
Copyright © 2016 Ben Ramsey

This work is licensed under Creative Commons Attribution-ShareAlike 4.0 International. For uses not covered under this license, please contact the author.

Ramsey, Ben. “Identify All the Things With UUIDs.” True North PHP. Microsoft Canada, Mississauga. 4 Nov. 2016. Conference presentation.

Photo Credits

1. “38-365 Fingerprint” by Bram Cymet
2. “Baby Feet” by Katelyn Kenderdine
3. “Random” by Vladimer Shioshvili
4. “Hello My Name Is.... 221/365” by Robert Occhialini
5. “Curvy Road Ahead” by Kit Ng
6. “Security door” by reynermedia
7. “Hello! My Name Is JEDi.” by Tyrone J Moore
8. “Mr Anatomini” by clement127
9. “Everything” by Dan Dvorscak
10. “thinker” by Fredrik Rubensson