Package ‘ulid’

July 9, 2019

Type Package
Title Generate Universally Unique Lexicographically Sortable Identifiers
Version 0.3.0
Date 2019-07-04
Maintainer Bob Rudis <bob@rud.is>
Description Universally unique identifiers ('UUIDs') can be suboptimal for many uses-cases because they aren't the most character efficient way of encoding 128 bits of randomness; v1/v2 versions are impractical in many environments, as they require access to a unique, stable MAC address; v3/v5 versions require a unique seed and produce randomly distributed IDs, which can cause fragmentation in many data structures; v4 provides no other information than randomness which can cause fragmentation in many data structures. 'ULIDs' (<https://github.com/ulid/spec>) have 128-bit compatibility with 'UUID', 1.21e+24 unique 'ULIDs' per millisecond, are lexicographically sortable, canonically encoded as a 26 character string, as opposed to the 36 character 'UUID', use Crockford's 'base32' for better efficiency and readability (5 bits per character), are case insensitive, have no special characters (i.e. are 'URL' safe) and have a onotonic sort order (correctly detects and handles the same millisecond).

URL https://gitlab.com/hrbrmstr/ulid
BugReports https://gitlab.com/hrbrmstr/ulid/issues
SystemRequirements C++11
NeedsCompilation yes
Encoding UTF-8
License MIT + file LICENSE
Suggests covr, tinytest, knitr, rmarkdown
Depends R (>= 3.2.0)
Imports Rcpp
RoxygenNote 6.1.1
LinkingTo Rcpp
VignetteBuilder knitr

Author Bob Rudis [aut, cre] (<https://orcid.org/0000-0001-5670-2640>), suyash [aut] (ULID C++ Port <https://github.com/suyash/ulid/>)

Repository CRAN
Date/Publication 2019-07-09 16:00:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts_generate</td>
<td>2</td>
</tr>
<tr>
<td>ulid</td>
<td>3</td>
</tr>
<tr>
<td>ULIDgenerate</td>
<td>4</td>
</tr>
<tr>
<td>unmarshal</td>
<td>5</td>
</tr>
</tbody>
</table>

Index 6

---

**ts_generate**

Generate ULIDs from timestamps

**Description**

This function generates a new Universally Unique Lexicographically Sortable Identifier from a vector of POSIXct timestamps.

**Usage**

```
   ts_generate(tsv)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsv</td>
<td>vector of POSIXct values</td>
</tr>
</tbody>
</table>

**Examples**

```
   ts_generate(as.POSIXct("2017-11-01 15:00:00", origin="1970-01-01"))
```
ulid

Generate Universally Unique Lexicographically Sortable Identifiers

Description

(grifted from https://github.com/ulid/spec)

Details

UUID can be suboptimal for many uses-cases because:

- It isn’t the most character efficient way of encoding 128 bits of randomness
- UUID v1/v2 is impractical in many environments, as it requires access to a unique, stable MAC address
- UUID v3/v5 requires a unique seed and produces randomly distributed IDs, which can cause fragmentation in many data structures
- UUID v4 provides no other information than randomness which can cause fragmentation in many data structures

Instead, herein is proposed ULID:

ulid() // 01AZ3NDEKTSV4RFFQ69G5FAV

- 128-bit compatibility with UUID
- 1.21e+24 unique ULIDs per millisecond
- Lexicographically sortable!
- Canonically encoded as a 26 character string, as opposed to the 36 character UUID
- Uses Crockford’s base32 for better efficiency and readability (5 bits per character)
- Case insensitive
- No special characters (URL safe)
- Monotonic sort order (correctly detects and handles the same millisecond)

01AN4Z07BY 79KA1307SR9X4MV3

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Randomness</td>
</tr>
<tr>
<td>48bits</td>
<td>80bits</td>
</tr>
</tbody>
</table>

Components

Timestamp

- 48 bit integer
- UNIX-time in milliseconds
- Won’t run out of space till the year 10889 AD.
Randomness

- 80 bits
- Cryptographically secure source of randomness, if possible

Sorting

The left-most character must be sorted first, and the right-most character sorted last (lexical order). The default ASCII character set must be used. Within the same millisecond, sort order is not guaranteed.

- URL: https://gitlab.com/hrbrmstr/ulid
- BugReports: https://gitlab.com/hrbrmstr/ulid/issues

Author(s)

Bob Rudis (bob@rud.is)

ULIDgenerate

Generate ULID

Description

ULIDgenerate() generates a new Universally Unique Lexicographically Sortable Identifier.

Usage

ULIDgenerate(n = 1L)

generate(n = 1L)

ulid_generate(n = 1L)

Arguments

n number of id’s to generate (default = 1)

Examples

ULIDgenerate()
unmarshal

Unmarshal a ULID into a data frame with timestamp and random bit-string columns

Description

Unmarshal a ULID into a data frame with timestamp and random bitstring columns

Usage

unmarshal(ulids)

Arguments

ulids character ULIDs (e.g. created with ULIDgenerate() )

Value

data frame (tibble)

Examples

unmarshal(ULIDgenerate())
Index

generate (ULIDgenerate), 4

ts_generate, 2

ulid, 3
ulid-package (ulid), 3
ulid_generate (ULIDgenerate), 4
ULIDgenerate, 4
ULIDgenerate(), 4, 5
unmarshal, 5