Package: uuid (via r-universe)

October 7, 2024

Version 1.2-1

Title Tools for Generating and Handling of UUIDs

Author Simon Urbanek [aut, cre, cph] (https://urbanek.org, <https://orcid.org/0000-0003-2297-1732>), Theodore Ts'o [aut, cph] (libuuid)

Maintainer Simon Urbanek <Simon.Urbanek@r-project.org>

Depends R (>= 2.9.0)

Description Tools for generating and handling of UUIDs (Universally Unique Identifiers).

License MIT + file LICENSE

URL https://www.rforge.net/uuid

BugReports https://github.com/s-u/uuid/issues

Repository https://s-u.r-universe.dev

RemoteUrl https://github.com/s-u/uuid

RemoteRef HEAD

RemoteSha c68c92ab93c7fc9f0a89d8d4e76cc695854b6d40

Contents

UUID *UUID Data Type*

Description

S3 class "UUID" represents vector of UUIDs in native form (128-bit). They are typically obtained by calling UUIDgenerate, UUIDparse or as.UUID.

Methods exist for common operations such as as.character, print, c, subsetting and comparison operators. Note that arithmetic and other operations are not allowed.

UUIDs have three possible representations: as character vectors (in the hyphenated 8-4-4-4-12 hexadecimal form), the UUID class described here and raw vectors. In the latter case the raw vector must be of length 16 or it must be a matrix with 16 rows. Since matrices in R are stored in colum-major format, UUID must be contiguous and thus form the *columns* of the raw matrix, which may be slightly counter-intuitive, but is far morecefficient.

as.character method exists for UUID objects and converts it to a character vector of lower-case UUID string representation.

as.raw method converts UUIDs to raw vectors or matrices as describe above. Similarly, a as.UUID method for raw vectors performs the inverse transformation.

Usage

```
as.UUID(x, ...)
is.UUID(x)
```

Arguments

x	object to coerce / check
...	unused

Details

Internally, the underlying object uses complex numbers to store 128-bit values with each UUID represented as one complex number. There may be cases where some operations strip the class attribute which will lead to complex values being visible, but their behavior should be regarded as undefined.

NA values in the UUID class are internally stored as a special value a2070000-0000-f07f-a207-00000000f07f which is not a valid UUID (since the version of that UUID is 15 which does not exist). This is an R extension and will be automatically converted to NA where possible, but the raw format does not support NAs so it will be visible there. Coercions to/from string and UUIDs handle NAs correctly and thus this internal representation should not be relied upon by any code and may change in the future.

Value

as.UUID returns an object of the class "UUID" representing a vector of UUIDs. Any elements that are not valid UUIDs will yield NA values.

is.UUID returns TRUE if the object is of the class "UUID" and FALSE otherwise.

Note

Comparisons are much faster between UUID vectors than between UUID vectors and other types, because in the latter case all values are coerced to strings before comparison which is very expensive.

However, x == y does not necessarily yield the same result as as.UUID(x) == as.UUID(y). For example, for x a valid UUID object of length one and y = "foo" the former will be FALSE while the latter will be NA due to coercion not yielding a valid UUID value represented by NA.

Author(s)

Simon Urbanek

Examples

```
(u <- as.UUID("837bc850-07d9-42f9-9afb-716409bf87b7"))
(uv <- c(u, NA, UUIDgenerate(n=3, output="uuid")))
as.character(u)
uv == u
is.na(uv)
identical(as.UUID(as.character(uv)), uv)
as.raw(u)

## all forms are can be coerced losslessly
identical(as.UUID(as.raw(uv)), uv)
identical(as.UUID(as.character(as.UUID(as.raw(uv)))), uv)
```

UUIDgenerate *UUID Functions*

Description

UUIDgenerate generates new Universally Unique Identifiers. It can be either time-based or random.

UUIDfromName generates deterministic UUIDs based on namespace UUID and a name (UUID version 3 and 5).

UUIDparse parses one of more UUIDs in string form and converts them to other internal formats.

UUIDvalidate checks the valitiy of UUIDs in string form.

Usage

```
UUIDgenerate(use.time = NA, n = 1L, output = c("string", "raw", "uuid"))
UUIDfromName(namespace, name, type = c("sha1", "md5"),
             output = c("string", "raw", "uuid"))
UUIDparse(what, output = c("uuid", "string", "raw", "logical"))
UUIDvalidate(what)
```

Arguments

use.time	logical, if TRUE then time-based UUID is generated, if FALSE then a random UUID is generated, if NA then random one is generated if a sufficiently reliable source of random numbers can be found, otherwise a time-based UUID is generated.
n	integer, number of UUIDs to generate.
output	type of the output. Valid types are: "string" for a character vector with UUIDs in textual representation (always lowercase), "raw" for a vector or matrix of raw bytes, "uuid" for an object of the class UUID and "logical" which only reports failure/success of the parsing, but not the actual values.
namespace	UUID defining the namespace
name	character vector of names to use for generating UUIDs. The result will yield as many UUIDs as there are elements in this vector.
type	string, type of the hash function to use when generating the UUIDs. "sha1" is recommended (version 5 UUID), "md5" is available for compatibility (version 3 UUID).
what	character vector which will be parsed into UUIDs.

Value

UUIDgenerate, UUIDfromName and UUIDparse values depend on the output argument as follows:

"string"	character vector with each element UUID in lowercase form, for UUIDparse strings that cannot be parsed will result in NA values
"raw"	raw vector with the UUIDs stores each as 16 bytes seqeuntially. If the output is more than one UUID then the result is a raw matrix with 16 rows and as many columns as there are input elements.
"uuid"	object of the class UUID which is a vector of UUIDs in 128-bit internal representation.
"logical"	only supported in UUIDparse and return code TRUE for valid UUID, FALSE for invalid input and NA for NA input.

UUIDvalidate is just a shorthand for UUIDparse(what, output="logical").

Note

The first argument is not n for historical reasons, beause the first version did only generate a single UUID.

Author(s)

Simon Urbanek, based on libuuid by Theodore Ts'o.

Examples

```
UUIDgenerate()
UUIDgenerate(TRUE)
UUIDgenerate(FALSE)

## see if the randomness is any good
length(unique(UUIDgenerate(n=1000)))

## generate a native UUID vector
(u <- UUIDgenerate(n=3, output="uuid"))

as.character(u)
as.raw(u[1])

UUIDgenerate(output="raw")

## UUID for DNS namespace
DNS.namespace <- "6ba7b810-9dad-11d1-80b4-00c04fd430c8"
## SHA1 (v5) - default
UUIDfromName(DNS.namespace, "r-project.org")
## MD5 (v3)
UUIDfromName(DNS.namespace, "r-project.org", type="md5")

## see ?UUID for more examples on UUID objects
```

Index