

Package ‘uuidx’

May 8, 2026

Title Modern UUIDs for R with a Rust Backend

Version 0.0.1

Description Generate, parse, and validate RFC 9562 UUIDs from R using the Rust 'uuid' crate via 'extendr'. Developed by Thomas Bryce Kelly at Icy Seas Co-Laboratory LLC. Version 7 UUIDs are the default for new identifiers, while versions 4, 5, 6, and legacy version 1 are also supported. Functions return character vectors by default and can also expose 16-byte raw representations for low-level workflows.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Config/rextendr/version 0.4.1

Config/testthat/edition 3

SystemRequirements Cargo (Rust's package manager), rustc >= 1.81

NeedsCompilation yes

Depends R (>= 4.3)

Suggests testthat (>= 3.0.0)

Author Thomas Bryce Kelly [aut, cre],
Icy Seas Co-Laboratory LLC [cph]

Maintainer Thomas Bryce Kelly <tkelly@icyseascolab.io>

Repository CRAN

Date/Publication 2026-04-10 11:30:02 UTC

Contents

uuid_generate	2
uuid_max	3
uuid_nil	3
uuid_parse	4
uuid_v1	5
uuid_v4	5

uuid_v5	6
uuid_v6	6
uuid_v7	7
uuid_validate	7
uuid_version	8

Index	9
--------------	----------

uuid_generate	<i>Generate UUIDs</i>
---------------	-----------------------

Description

uuid_generate() is the main entry point for new UUID creation. It defaults to version 7, which is typically a practical default for new identifiers because it keeps the randomness users expect while sorting naturally by time.

Usage

```
uuid_generate(
  n = 1L,
  version = c("v7", "v4", "v5", "v6", "v1"),
  namespace = NULL,
  name = NULL,
  output = c("string", "raw")
)
```

Arguments

n	Number of UUIDs to generate. Must be a single non-negative integer.
version	UUID version to generate. Defaults to "v7".
namespace	Namespace UUID used for version 5 generation. Standard aliases "dns", "url", "oid", and "x500" are accepted.
name	Name value or values used for version 5 generation.
output	Output format. "string" returns canonical UUID strings and "raw" returns a list of raw vectors, each of length 16.

Details

Version 5 is name-based and deterministic, so it uses namespace and name instead of n.

Value

A character vector for output = "string" or a list of raw vectors for output = "raw".

Examples

```
uuid_generate()
uuid_generate(3, version = "v4")
uuid_generate(version = "v5", namespace = "dns", name = c("a", "b"))
```

uuid_max	<i>Return the Max UUID</i>
----------	----------------------------

Description

Return the Max UUID

Usage

```
uuid_max(output = c("string", "raw"))
```

Arguments

output	Output format. "string" returns the canonical UUID string and "raw" returns a raw vector of length 16.
--------	--

Value

A length-one character vector or a raw vector.

Examples

```
uuid_max()
uuid_max(output = "raw")
```

uuid_nil	<i>Return the Nil UUID</i>
----------	----------------------------

Description

Return the Nil UUID

Usage

```
uuid_nil(output = c("string", "raw"))
```

Arguments

output	Output format. "string" returns the canonical UUID string and "raw" returns a raw vector of length 16.
--------	--

Value

A length-one character vector or a raw vector.

Examples

```
uuid_nil()
uuid_nil(output = "raw")
```

uuid_parse

Parse UUIDs

Description

uuid_parse() canonicalizes UUID strings and can optionally expose their raw bytes or field-level structure.

Usage

```
uuid_parse(x, output = c("string", "raw", "fields"))
```

Arguments

x	Character vector of UUID strings.
output	Output format. "string" returns canonical strings, "raw" returns a list of 16-byte raw vectors, and "fields" returns a data frame with class "uuid_fields".

Details

For output = "fields", the returned data frame exposes structural UUID fields. These field names follow the conventional UUID layout, although the semantics of those fields vary by UUID version.

Value

A character vector, a list of raw vectors, or a "uuid_fields" data frame depending on output.

Examples

```
x = uuid_v7(2)
uuid_parse(x)
uuid_parse(x, output = "raw")
uuid_parse(x, output = "fields")
```

uuid_v1	<i>Generate Legacy Version 1 UUIDs</i>
---------	--

Description

Version 1 remains available for compatibility work, but it is treated as a legacy option in this package.

Usage

```
uuid_v1(n = 1L, output = c("string", "raw"))
```

Arguments

n	Number of UUIDs to generate. Must be a single non-negative integer.
output	Output format. "string" returns canonical UUID strings and "raw" returns a list of raw vectors, each of length 16.

Value

A character vector or a list of 16-byte raw vectors.

uuid_v4	<i>Generate Version 4 UUIDs</i>
---------	---------------------------------

Description

Version 4 UUIDs are fully random identifiers.

Usage

```
uuid_v4(n = 1L, output = c("string", "raw"))
```

Arguments

n	Number of UUIDs to generate. Must be a single non-negative integer.
output	Output format. "string" returns canonical UUID strings and "raw" returns a list of raw vectors, each of length 16.

Value

A character vector or a list of 16-byte raw vectors.

 uuid_v5

Generate Deterministic Version 5 UUIDs

Description

Version 5 UUIDs are derived from a namespace UUID and a name using SHA-1.

Usage

```
uuid_v5(namespace, name, output = c("string", "raw"))
```

Arguments

namespace	Namespace UUID or one of "dns", "url", "oid", or "x500".
name	One or more name strings.
output	Output format. "string" returns canonical UUID strings and "raw" returns a list of raw vectors, each of length 16.

Value

A character vector or a list of 16-byte raw vectors.

Examples

```
uuid_v5("dns", "example.com")
uuid_v5("dns", c("alpha", "beta"))
```

uuid_v6

Generate Version 6 UUIDs

Description

Version 6 UUIDs retain the timestamp-based lineage of version 1 while being more naturally sortable.

Usage

```
uuid_v6(n = 1L, output = c("string", "raw"))
```

Arguments

n	Number of UUIDs to generate. Must be a single non-negative integer.
output	Output format. "string" returns canonical UUID strings and "raw" returns a list of raw vectors, each of length 16.

Value

A character vector or a list of 16-byte raw vectors.

uuid_v7	<i>Generate Version 7 UUIDs</i>
---------	---------------------------------

Description

Version 7 UUIDs are time-ordered and are the default in `uuidx`.

Usage

```
uuid_v7(n = 1L, output = c("string", "raw"))
```

Arguments

<code>n</code>	Number of UUIDs to generate. Must be a single non-negative integer.
<code>output</code>	Output format. "string" returns canonical UUID strings and "raw" returns a list of raw vectors, each of length 16.

Value

A character vector or a list of 16-byte raw vectors.

uuid_validate	<i>Validate UUID Strings</i>
---------------	------------------------------

Description

Validate UUID Strings

Usage

```
uuid_validate(x)
```

Arguments

<code>x</code>	Vector to validate.
----------------	---------------------

Value

A logical vector where valid UUIDs are TRUE.

Examples

```
uuid_validate(c(uuid_v7(), "not-a-uuid"))
```

uuid_version	<i>Detect UUID Versions</i>
--------------	-----------------------------

Description

Detect UUID Versions

Usage

```
uuid_version(x)
```

Arguments

x Vector of UUID strings.

Value

An integer vector containing UUID version numbers, with NA for invalid inputs.

Examples

```
x = c(uuid_v7(), uuid_v4(), "not-a-uuid")
uuid_version(x)
```

Index

uuid_generate, [2](#)
uuid_max, [3](#)
uuid_nil, [3](#)
uuid_parse, [4](#)
uuid_v1, [5](#)
uuid_v4, [5](#)
uuid_v5, [6](#)
uuid_v6, [6](#)
uuid_v7, [7](#)
uuid_validate, [7](#)
uuid_version, [8](#)