

Struktur einer UUID

Grundstruktur einer UUID

Eine UUID besteht immer aus **128 Bits = 16 Bytes**.

Typische Darstellung:

```
123e4567-e89b-12d3-a456-426614174000
```

Aufgeteilt:

```
8-4-4-4-12 Hex-Zeichen
```

Das entspricht:

- 32 Bit – 16 Bit – 16 Bit – 16 Bit – 48 Bit

Bit-Level Aufbau (allgemein)

Eine UUID hat zwei wichtige feste Bestandteile:

1. Version (4 Bits)

- Gibt an, **wie die UUID erzeugt wurde (v1-v7)**
- Sitzt im **3. Block** (die ersten 4 Bits davon)

Beispiel:

```
xxxxxxxx-xxxx-1xxx-....
```

↑

Version

2. Variant (2–3 Bits)

- Gibt das Format der UUID an (meist RFC 4122 / 9562)
- Sitzt im **4. Block am Anfang**

Typische Werte:

- `10xx` → Standard UUID (das, was du fast immer siehst)

Beispiel:

```
xxxxxxxx-xxxx-xxxx-8xxx-....  
                ↑  
            Variant
```

Beispiel analysiert

UUID:

```
123e4567-e89b-12d3-a456-426614174000
```

 Zerlegt:

- `123e4567` → Teil 1
- `e89b` → Teil 2
- `12d3` → Teil 3 → **Version = 1**
- `a456` → Teil 4 → **Variant = a (= 1010 → RFC-konform)**
- `426614174000` → Teil 5

Unterschied je nach Version

Jetzt wird's interessant:

Der Inhalt der 128 Bits hängt stark von der Version ab.

☐☐ UUID v1 (Zeitbasiert)

Aufbau:

```
time_low (32)
time_mid (16)
time_high + version (16)
variant + clock_seq (16)
node (48)
```

Enthält:

- Zeitstempel
- Clock Sequence (gegen Kollisionen)
- MAC-Adresse

☐☐ Deshalb:

- eindeutig
 - aber privacy-problematisch
-

☐☐ UUID v4 (Random)

Aufbau:

```
random (122 bits)
+ version (4 bits)
+ variant (2-3 bits)
```

☐☐ Fast alles ist Zufall

Nur:

- Version
- Variant

sind fest codiert

UUID v7 (modern UUID)

Aufbau (vereinfacht):

```
unix_ts_ms (48 bits)
version (4 bits)
rand (12 bits)
variant (2-3 bits)
rand (62 bits)
```

UUID Mischung aus:

- Zeit (Millisekunden seit 1970)
- Zufall

Vorteil:

- Sortierbar
- Keine MAC-Adresse
- Gute Verteilung

UUID Visuelle Darstellung (v7 Beispiel)

```
|----- Zeit (48) -----|ver|--- rand ---|var|----- rand -----|
```

UUID Wichtige Erkenntnisse

- UUID ist **kein reiner Zufallsstring**
- Einige Bits sind immer reserviert:
 - Version
 - Variant

- Je nach Version:
 - Zeit enthalten ☐
 - Hash enthalten ☐
 - MAC enthalten ☐
 - oder komplett random ☐
-

Kleine Stolperfallen

- Hex-Darstellung $\neq$ echte interne Reihenfolge (Endianness kann verwirren)
 - v1/v6 können Infos leaken (Zeit, Gerät)
 - v4 ist nicht sortierbar → schlecht für DB-Indizes
 - v7 löst viele dieser Probleme
-

Created 2026-06-19 07:41:24 UTC by Mich3l

Updated 2026-06-20 20:48:33 UTC by Mich3l