UUID UUID Standards Guide

toolpilot.dev/cheatsheets/uuid-cheat-sheet/

UUID versions 1-5, format breakdown, generation examples in Python/JavaScript/Go, and when to use each version.

UUID Format

```
xxxxxxxx-xxxx-Mxxx-Nxxx-xxxxxxxxxxxx

Example: 550e8400-e29b-41d4-a716-446655440000

M = version digit (1-5)
N = variant bits (8, 9, a, or b for RFC 4122)
Total: 128 bits = 32 hex digits + 4 dashes
```

UUID Versions

Version	Algorithm	Unique by	Use case
v1	Time-based	MAC + timestamp	Sortable, DB primary keys
v2	DCE Security	MAC + POSIX UID	Legacy, rarely used
v3	MD5 hash	Namespace + name	Deterministic, reproducible IDs
v4	Random	128-bit random	General purpose (most common)
v5	SHA-1 hash	Namespace + name	Deterministic (preferred over v3)
v7	Unix time + random	Timestamp + random	Sortable, modern DBs (2023+)

Standard Namespaces (for v3/v5)

Namespace	UUID
DNS	6ba7b810-9dad-11d1-80b4-00c04fd430c8
URL	6ba7b811-9dad-11d1-80b4-00c04fd430c8
OID	6ba7b812-9dad-11d1-80b4-00c04fd430c8
X.500 DN	6ba7b814-9dad-11d1-80b4-00c04fd430c8

Generation Examples

```
# Python
import uuid
uuid.uuid4()           # v4 random
uuid.uuid1()           # v1 time-based
uuid.uuid5(uuid.NAMESPACE_DNS, 'example.com')  # v5

# JavaScript (browser/Node)
crypto.randomUUID()    # v4 (modern)

# Go
import "github.com/google/uuid"
uuid.New()             // v4

# SQL (PostgreSQL)
SELECT gen_random_uuid();   -- v4
```

When To Use Which Version

- v4: Default choice for most applications — simple, random, no dependencies
- v7: New projects needing sortable IDs (better DB index performance than v4)
- v5: Need same ID for same input (idempotent operations, content addressing)
- v1: Legacy systems requiring MAC address traceability

- Avoid v2/v3: v2 is rarely supported; v5 is strictly better than v3