# UUID Functions

**13 functions**

Preliminary note - look http://en.wikipedia.org/wiki/UUID

Create - 6 functions

Transform - 2 functions

Read - 5 functions

---

returns <null> instead of 0, '' (empty string) or '17.11.1858'

Output RETURN mechanism if nothing other is published: FREE_IT TestSQLs with NULL run only in FireBird 2.0

---

**UUID functions: Preliminary note**

A Universally Unique Identifier is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). The intent of UUIDs is to enable distributed systems to uniquely identify information without significant central coordination. Thus, anyone can create a UUID and use it to identify something with reasonable confidence that the identifier will never be unintentionally used by anyone for anything else. Information labelled with UUIDs can therefore be later combined into a single database without need to resolve name conflicts. The most widespread use of this standard is in Microsoft's Globally Unique Identifiers (GUIDs) which implement this standard.

A UUID is essentially a 16-byte (128 bit) number and in its canonical form a UUID may look like this: 550e8400-e29b-41d4-a716-446655440000

The number of theoretically possible UUIDs is therefore 25616 or about 3.4 × 1038. This means that during a 10 billion year lifetime of the Earth, about 1 trillion UUIDs have to be created every nanosecond to exhaust the number of UUIDs.

The term GUID usually references Microsoft's implementation of the UUID standard, however, many other pieces of software use the term GUID including Oracle Database and Novell eDirectory.

Conceptually, the original (version 1) generation scheme for UUIDs was to concatenate the UUID version with the MAC address of the computer that is generating the UUID, and with the number of 100-nanosecond intervals since the adoption of the Gregorian calendar. In practice, the actual algorithm is more complicated. This scheme has been criticized in that it is not sufficiently 'opaque'; it reveals both the identity of the computer that generated the UUID and the time at which it did so.

In the RFC4122 there are 5 UUID-versions descriped:

1. time-based
    1. with a real MAC-adress descriped in the IEEE 802-standard
    2. with a random-generated MAC-Adresse descriped in the IEEE 802-standard
2. DCE-security-version (with POSIX UIDs)
3. namespace-based (with MD5)
4. random created
5. namespace-based (SHA-1)

For purpose of the locally generated unique IDs for records with compatibility under Windows and Linux with the same algorithm only the versions 1 and 4 can be used (version 3 and 5 not because there is not surely a name-space on the computer): Because you prefer a decoding or a non-decoding UUID we made 3 different kinds of UUIDs:

F_UUID1MAC

```
  version 1 with a real MAC-Adresse
  you can decode the creating-time and the MAC-adress of the computer
```

F_UUID1RAND

```
  version 1 with a randomly generated MAC-adresse
  only decode of the creating-time possible
```

F_UUID4

```
  version 4
  no decoding possible
```

In the uuidlib from Ian Newby we found a modification of the version 1b (with random generated MAC-adress) which was compressed instead of the output-mask from the RFC4122. This algorithm has advantages on created UUIDs on the same server because it first give out the MAC-adress and then the timestamp. So the index could work quicker. We also took this algorithm, incl. the transforming-functions, so we have a compressed version of every of the 3 basic-versions.

F_UUID1MACCOMPR, UUID1RANDCOMPR, UUID4COMPR.

From:
http://ibexpert.com/docu/ - **IBExpert**

Permanent link:
**http://ibexpert.com/docu/doku.php?id=04-ibexpert-udf-functions:04-05-uuid-functions**

Last update: **2023/07/06 18:04**