

**NAME**

**uuid\_compare**, **uuid\_create**, **uuid\_create\_nil**, **uuid\_equal**, **uuid\_from\_string**, **uuid\_hash**, **uuid\_is\_nil**, **uuid\_to\_string** - DCE 1.1 compliant UUID functions

**LIBRARY**

Standard C Library (libc, -lc)

**SYNOPSIS**

```
#include <uuid.h>
```

```
int32_t
```

```
uuid_compare(const uuid_t *uuid1, const uuid_t *uuid2, uint32_t *status);
```

```
void
```

```
uuid_create(uuid_t *uuid, uint32_t *status);
```

```
void
```

```
uuid_create_nil(uuid_t *uuid, uint32_t *status);
```

```
int32_t
```

```
uuid_equal(const uuid_t *uuid1, const uuid_t *uuid2, uint32_t *status);
```

```
void
```

```
uuid_from_string(const char *str, uuid_t *uuid, uint32_t *status);
```

```
uint16_t
```

```
uuid_hash(const uuid_t *uuid, uint32_t *status);
```

```
int32_t
```

```
uuid_is_nil(const uuid_t *uuid, uint32_t *status);
```

```
void
```

```
uuid_to_string(const uuid_t *uuid, char **str, uint32_t *status);
```

```
void
```

```
uuid_enc_le(void *buf, const uuid_t *uuid);
```

```
void
```

```
uuid_dec_le(const void *buf, uuid_t *);
```

*void*

```
uuid_enc_be(void *buf, const uuid_t *uuid);
```

*void*

```
uuid_dec_be(const void *buf, uuid_t *);
```

## DESCRIPTION

The family of DCE 1.1 compliant UUID functions allow applications to operate on universally unique identifiers, or UUIDs. The **uuid\_create()** and **uuid\_create\_nil()** functions create UUIDs. To convert from the binary representation to the string representation or vice versa, use **uuid\_to\_string()** or **uuid\_from\_string()** respectively.

The **uuid\_to\_string()** function set *\*str* to be a pointer to a buffer sufficiently large to hold the string. This pointer should be passed to **free(3)** to release the allocated storage when it is no longer needed.

The **uuid\_enc\_le()** and **uuid\_enc\_be()** functions encode a binary representation of a UUID into an octet stream in little-endian and big-endian byte-order, respectively. The destination buffer must be pre-allocated by the caller, and must be large enough to hold the 16-octet binary UUID. These routines are not part of the DCE RPC API. They are provided for convenience.

The **uuid\_dec\_le()** and **uuid\_dec\_be()** functions decode a UUID from an octet stream in little-endian and big-endian byte-order, respectively. These routines are not part of the DCE RPC API. They are provided for convenience.

The **uuid\_compare()** and **uuid\_equal()** functions compare two UUIDs for equality. UUIDs are equal if pointers *a* and *b* are equal or both NULL, or if the structures *a* and *b* point to are equal. **uuid\_compare()** returns 0 if the UUIDs are equal, -1 if *a* is less than *b*, and 1 if *a* is greater than *b*. **uuid\_equal()** returns 1 if the UUIDs are equal, 0 if they are not equal.

The **uuid\_is\_nil()** function compares a UUID to NULL. The function returns 1 if *u* is NULL or if the UUID consists of all zeros, and zero otherwise.

The **uuid\_hash()** function returns a 16-bit hash value for the specified UUID.

## RETURN VALUES

The successful or unsuccessful completion of the function is returned in the *status* argument. Possible values are:

**uuid\_s\_ok**                      The function completed successfully.

- `uuid_s_bad_version`      The UUID does not have a known version.
- `uuid_s_invalid_string_uuid`      The string representation of an UUID is not valid.
- `uuid_s_no_memory`      The function can not allocate memory to store an UUID representation.

**`uuid_compare()`, `uuid_equal()`, `uuid_is_nil()`, and `uuid_hash()`** always set *status* to `uuid_s_ok`.

### SEE ALSO

`uuidgen(1)`, `uuidgen(2)`

### STANDARDS

The UUID functions conform to the DCE 1.1 RPC specification.

### BUGS

This manpage can be improved.