

**NAME**

`add_key` - add a key to the kernel's key management facility

**SYNOPSIS**

```
#include <keyutils.h>
```

```
key_serial_t add_key(const char *type, const char *description,
                    const void *payload, size_t plen,
                    key_serial_t keyring);
```

**DESCRIPTION**

`add_key()` asks the kernel to create or update a key of the given *type* and *description*, instantiate it with the *payload* of length *plen*, and to attach it to the nominated *keyring* and to return its serial number.

The key type may reject the data if it's in the wrong format or in some other way invalid.

If the destination *keyring* already contains a key that matches the specified *type* and *description*, then, if the key type supports it, that key will be updated rather than a new key being created; if not, a new key will be created and it will displace the link to the extant key from the keyring.

The destination *keyring* serial number may be that of a valid keyring to which the caller has write permission, or it may be a special keyring ID:

**KEY\_SPEC\_THREAD\_KEYRING**

This specifies the caller's thread-specific keyring.

**KEY\_SPEC\_PROCESS\_KEYRING**

This specifies the caller's process-specific keyring.

**KEY\_SPEC\_SESSION\_KEYRING**

This specifies the caller's session-specific keyring.

**KEY\_SPEC\_USER\_KEYRING**

This specifies the caller's UID-specific keyring.

**KEY\_SPEC\_USER\_SESSION\_KEYRING**

This specifies the caller's UID-session keyring.

**KEY TYPES**

There are a number of key types available in the core key management code, and these can be specified to this function:

**“user”** Keys of the user-defined key type may contain a blob of arbitrary data, and the *description* may be any valid string, though it is preferred that the description be prefixed with a string representing the service to which the key is of interest and a colon (for instance **“afs:mykey”**). The *payload* may be empty or NULL for keys of this type.

**“keyring”**

Keyrings are special key types that may contain links to sequences of other keys of any type. If this interface is used to create a keyring, then a NULL *payload* should be specified, and *plen* should be zero.

**RETURN VALUE**

On success `add_key()` returns the serial number of the key it created or updated. On error, the value -1 will be returned and `errno` will have been set to an appropriate error.

**ERRORS****ENOKEY**

The keyring doesn't exist.

**EKEYEXPIRED**

The keyring has expired.

**EKEYREVOKED**

The keyring has been revoked.

**EINVAL**

The payload data was invalid.

**ENOMEM**

Insufficient memory to create a key.

**EDQUOT**

The key quota for this user would be exceeded by creating this key or linking it to the keyring.

**EACCES**

The keyring wasn't available for modification by the user.

**LINKING**

Although this is a Linux system call, it is not present in *libc* but can be found rather in *libkeyutils*. When linking, **-lkeyutils** should be specified to the linker.

**SEE ALSO**

**keyctl(1)**, **keyctl(2)**, **request\_key(2)**

**COLOPHON**

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <http://www.kernel.org/doc/man-pages/>.