## NAME

RAND_add, RAND_poll, RAND_seed, RAND_status, RAND_event, RAND_screen, RAND_keep_random_devices_open - add randomness to the PRNG or get its status

## SYNOPSIS

```
#include <openssl/rand.h>

int RAND_status(void);
int RAND_poll();

void RAND_add(const void *buf, int num, double randomness);
void RAND_seed(const void *buf, int num);

void RAND_keep_random_devices_open(int keep);
```

The following functions have been deprecated since OpenSSL 1.1.0, and can be hidden entirely by defining **OPENSSL_API_COMPAT** with a suitable version value, see **openssl_user_macros**(7):

```
int RAND_event(UINT iMsg, WPARAM wParam, LPARAM lParam);
void RAND_screen(void);
```

## DESCRIPTION

These functions can be used to seed the random generator and to check its seeded state. In general, manual (re-)seeding of the default OpenSSL random generator (**RAND_OpenSSL**(3)) is not necessary (but allowed), since it does (re-)seed itself automatically using trusted system entropy sources. This holds unless the default RAND_METHOD has been replaced or OpenSSL was built with automatic reseeding disabled, see **RAND**(7) for more details.

**RAND_status()** indicates whether or not the random generator has been sufficiently seeded. If not, functions such as **RAND_bytes**(3) will fail.

**RAND_poll()** uses the system's capabilities to seed the random generator using random input obtained from polling various trusted entropy sources. The default choice of the entropy source can be modified at build time, see **RAND**(7) for more details.

**RAND_add()** mixes the **num** bytes at **buf** into the internal state of the random generator. This function will not normally be needed, as mentioned above. The **randomness** argument is an estimate of how much randomness is contained in **buf**, in bytes, and should be a number between zero and **num**. Details about sources of randomness and how to estimate their randomness can be found in the literature; for example [NIST SP 800-90B]. The content of **buf** cannot be recovered from subsequent random

generator output. Applications that intend to save and restore random state in an external file should consider using **RAND_load_file**(3) instead.

NOTE: In FIPS mode, random data provided by the application is not considered to be a trusted entropy source. It is mixed into the internal state of the RNG as additional data only and this does not count as a full reseed. For more details, see **EVP_RAND**(7).

**RAND_seed()** is equivalent to **RAND_add()** with **randomness** set to **num**.

**RAND_keep_random_devices_open()** is used to control file descriptor usage by the random seed sources. Some seed sources maintain open file descriptors by default, which allows such sources to operate in a **chroot**(2) jail without the associated device nodes being available. When the **keep** argument is zero, this call disables the retention of file descriptors. Conversely, a nonzero argument enables the retention of file descriptors. This function is usually called during initialization and it takes effect immediately. This capability only applies to the default provider.

**RAND_event()** and **RAND_screen()** are equivalent to **RAND_poll()** and exist for compatibility reasons only. See HISTORY section below.

## RETURN VALUES

**RAND_status()** returns 1 if the random generator has been seeded with enough data, 0 otherwise.

**RAND_poll()** returns 1 if it generated seed data, 0 otherwise.

**RAND_event()** returns **RAND_status()**.

The other functions do not return values.

## SEE ALSO

**RAND_bytes**(3), **RAND_egd**(3), **RAND_load_file**(3), **RAND**(7) **EVP_RAND**(7)

## HISTORY

**RAND_event()** and **RAND_screen()** were deprecated in OpenSSL 1.1.0 and should not be used.

## COPYRIGHT