NAME

fido_dev_set_io_functions, **fido_dev_set_sigmask**, **fido_dev_set_timeout**, **fido_dev_set_transport_functions**, **fido_dev_io_handle** - FIDO2 device I/O interface

SYNOPSIS

```
#include <fido.h>

typedef void *fido_dev_io_open_t(const char *);
typedef void  fido_dev_io_close_t(void *);
typedef int   fido_dev_io_read_t(void *, unsigned char *, size_t, int);
typedef int   fido_dev_io_write_t(void *, const unsigned char *, size_t);

typedef struct fido_dev_io {
        fido_dev_io_open_t  *open;
        fido_dev_io_close_t *close;
        fido_dev_io_read_t  *read;
        fido_dev_io_write_t *write;
} fido_dev_io_t;

#ifdef _WIN32
typedef int fido_sigset_t;
#else
typedef sigset_t fido_sigset_t;
#endif

typedef int   fido_dev_rx_t(struct fido_dev *,
          uint8_t, unsigned char *, size_t, int);
typedef int   fido_dev_tx_t(struct fido_dev *,
          uint8_t, const unsigned char *, size_t);

typedef struct fido_dev_transport {
        fido_dev_rx_t *rx;
        fido_dev_tx_t *tx;
} fido_dev_transport_t;

int
fido_dev_set_io_functions(fido_dev_t *dev, const fido_dev_io_t *io);

int
fido_dev_set_sigmask(fido_dev_t *dev, const fido_sigset_t *sigmask);
```

int
fido_dev_set_timeout(*fido_dev_t *dev*, *int ms*);

int
fido_dev_set_transport_functions(*fido_dev_t *dev*, *const fido_dev_transport_t *t*);

*void **
fido_dev_io_handle(*const fido_dev_t *dev*);

DESCRIPTION

The **fido_dev_set_io_functions**() function sets the I/O handlers used by *libfido2* to talk to *dev*. By default, these handlers are set to the operating system's native HID or NFC interfaces. They are defined as follows:

fido_dev_open_t
: Receives a *const char ** holding a path and opens the corresponding device, returning a non-NULL opaque pointer on success and NULL on error.

fido_dev_close_t
: Receives the opaque pointer returned by *fido_dev_open_t* and closes the device.

fido_dev_read_t
: Reads a single transmission unit (HID report, APDU) from a device. The first parameter is the opaque pointer returned by *fido_dev_open_t*. The second parameter is the read buffer, and the third parameter is the read buffer size. The fourth parameter is the number of milliseconds the caller is willing to sleep, should the call need to block. If this value holds -1, *fido_dev_read_t* may block indefinitely. On success, the number of bytes read is returned. On error, -1 is returned.

fido_dev_write_t
: Writes a single transmission unit (HID report, APDU) to *dev*. The first parameter is the opaque pointer returned by *fido_dev_open_t*. The second parameter is the write buffer, and the third parameter is the number of bytes to be written. A *fido_dev_write_t* may block. On success, the number of bytes written is returned. On error, -1 is returned.

When calling **fido_dev_set_io_functions**(), the *open*, *close*, *read*, and *write* fields of *io* may not be NULL.

No references to *io* are held by **fido_dev_set_io_functions**().

The **fido_dev_set_sigmask**() function may be used to specify a non-NULL signal mask *sigmask* to be used while *libfido2's* default I/O handlers wait on *dev*. On UNIX-like operating systems, *fido_sigset_t* is defined as *sigset_t*. On Windows, *fido_sigset_t* is defined as *int* and **fido_dev_set_sigmask**() is a no-op.

No references to *sigmask* are held by **fido_dev_set_sigmask**().

The **fido_dev_set_timeout**() function informs *libfido2* not to block for more than *ms* milliseconds while communicating with *dev*. If a timeout occurs, the corresponding *fido_dev_** function will fail with FIDO_ERR_RX. If *ms* is -1, then *libfido2* may block indefinitely. This is the default behaviour. When using the Windows Hello backend, *ms* is used as a guidance and may be overwritten by the platform.

The **fido_dev_set_transport_functions**() function sets the transport functions used by *libfido2* to talk to *dev*. While the I/O handlers are responsible for sending and receiving transmission units of initialization and continuation packets already formatted by *libfido2*, the transport handlers are responsible for sending and receiving the CTAPHID commands and data directly, as defined in the FIDO Client to Authenticator Protocol (CTAP) standard. They are defined as follows:

fido_dev_tx_t
: Receives a device, a CTAPHID command to transmit, a data buffer to transmit, and the length of the data buffer. On success, 0 is returned. On error, -1 is returned.

fido_dev_rx_t
: Receives a device, a CTAPHID command whose response the caller expects to receive, a data buffer to receive into, the size of the data buffer determining the maximum length of a response, and the maximum number of milliseconds to wait for a response. On success, the number of bytes read into the data buffer is returned. On error, -1 is returned.

When transport functions are specified, *libfido2* will use them instead of the read and write functions of the I/O handlers. However, the I/O handlers must still be specified to open and close the device.

The **fido_dev_io_handle**() function returns the opaque pointer returned by the open function of the I/O handlers. This is useful mainly for the transport functions, which unlike the I/O handlers are passed the *fido_dev_t* pointer instead of the opaque I/O handle.

RETURN VALUES

On success, **fido_dev_set_io_functions**(), **fido_dev_set_transport_functions**(), **fido_dev_set_sigmask**(), and **fido_dev_set_timeout**() return FIDO_OK. On error, a different error code defined in <*fido/err.h*> is returned.

SEE ALSO

fido_dev_info_manifest(3), fido_dev_open(3)

Client to Authenticator Protocol (CTAP),
https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-20210615.html,
FIDO Alliance, 2021-06-15, Proposed Standard, Version 2.1.