

NAME

fido_dev_open, **fido_dev_open_with_info**, **fido_dev_close**, **fido_dev_cancel**, **fido_dev_new**,
fido_dev_new_with_info, **fido_dev_free**, **fido_dev_force_fido2**, **fido_dev_force_u2f**, **fido_dev_is_fido2**,
fido_dev_is_winhello, **fido_dev_supports_credman**, **fido_dev_supports_cred_prot**,
fido_dev_supports_permissions, **fido_dev_supports_pin**, **fido_dev_supports_uv**, **fido_dev_has_pin**,
fido_dev_has_uv, **fido_dev_protocol**, **fido_dev_build**, **fido_dev_flags**, **fido_dev_major**, **fido_dev_minor**
- FIDO2 device open/close and related functions

SYNOPSIS

```
#include <fido.h>
```

int

```
fido_dev_open(fido_dev_t *dev, const char *path);
```

int

```
fido_dev_open_with_info(fido_dev_t *dev);
```

int

```
fido_dev_close(fido_dev_t *dev);
```

int

```
fido_dev_cancel(fido_dev_t *dev);
```

fido_dev_t *

```
fido_dev_new(void);
```

fido_dev_t *

```
fido_dev_new_with_info(const fido_dev_info_t *);
```

void

```
fido_dev_free(fido_dev_t **dev_p);
```

void

```
fido_dev_force_fido2(fido_dev_t *dev);
```

void

```
fido_dev_force_u2f(fido_dev_t *dev);
```

bool

```
fido_dev_is_fido2(const fido_dev_t *dev);
```

bool

fido_dev_is_winhello(*const fido_dev_t *dev*);

bool

fido_dev_supports_credman(*const fido_dev_t *dev*);

bool

fido_dev_supports_cred_prot(*const fido_dev_t *dev*);

bool

fido_dev_supports_permissions(*const fido_dev_t *dev*);

bool

fido_dev_supports_pin(*const fido_dev_t *dev*);

bool

fido_dev_supports_uv(*const fido_dev_t *dev*);

bool

fido_dev_has_pin(*const fido_dev_t *dev*);

bool

fido_dev_has_uv(*const fido_dev_t *dev*);

uint8_t

fido_dev_protocol(*const fido_dev_t *dev*);

uint8_t

fido_dev_build(*const fido_dev_t *dev*);

uint8_t

fido_dev_flags(*const fido_dev_t *dev*);

uint8_t

fido_dev_major(*const fido_dev_t *dev*);

uint8_t

fido_dev_minor(*const fido_dev_t *dev*);

DESCRIPTION

The **fido_dev_open()** function opens the device pointed to by *path*, where *dev* is a freshly allocated or otherwise closed *fido_dev_t*. If *dev* claims to be FIDO2, *libfido2* will attempt to speak FIDO2 to *dev*. If that fails, *libfido2* will fallback to U2F unless the FIDO_DISABLE_U2F_FALLBACK flag was set in *fido_init(3)*.

The **fido_dev_open_with_info()** function opens *dev* as previously allocated using **fido_dev_new_with_info()**.

The **fido_dev_close()** function closes the device represented by *dev*. If *dev* is already closed, **fido_dev_close()** is a NOP.

The **fido_dev_cancel()** function cancels any pending requests on *dev*.

The **fido_dev_new()** function returns a pointer to a newly allocated, empty *fido_dev_t*. If memory cannot be allocated, NULL is returned.

The **fido_dev_new_with_info()** function returns a pointer to a newly allocated *fido_dev_t* with *fido_dev_info_t* parameters, for use with *fido_dev_info_manifest(3)* and **fido_dev_open_with_info()**. If memory cannot be allocated, NULL is returned.

The **fido_dev_free()** function releases the memory backing **dev_p*, where **dev_p* must have been previously allocated by **fido_dev_new()**. On return, **dev_p* is set to NULL. Either *dev_p* or **dev_p* may be NULL, in which case **fido_dev_free()** is a NOP.

The **fido_dev_force_fido2()** function can be used to force CTAP2 communication with *dev*, where *dev* is an open device.

The **fido_dev_force_u2f()** function can be used to force CTAP1 (U2F) communication with *dev*, where *dev* is an open device.

The **fido_dev_is_fido2()** function returns true if *dev* is a FIDO2 device.

The **fido_dev_is_winhello()** function returns true if *dev* is a Windows Hello device.

The **fido_dev_supports_credman()** function returns true if *dev* supports CTAP 2.1 Credential Management.

The **fido_dev_supports_cred_prot()** function returns true if *dev* supports CTAP 2.1 Credential Protection.

The **fido_dev_supports_permissions()** function returns true if *dev* supports CTAP 2.1 UV token permissions.

The **fido_dev_supports_pin()** function returns true if *dev* supports CTAP 2.0 Client PINs.

The **fido_dev_supports_uv()** function returns true if *dev* supports a built-in user verification method.

The **fido_dev_has_pin()** function returns true if *dev* has a CTAP 2.0 Client PIN set.

The **fido_dev_has_uv()** function returns true if *dev* supports built-in user verification and its user verification feature is configured.

The **fido_dev_protocol()** function returns the CTAPHID protocol version identifier of *dev*.

The **fido_dev_build()** function returns the CTAPHID build version number of *dev*.

The **fido_dev_flags()** function returns the CTAPHID capabilities flags of *dev*.

The **fido_dev_major()** function returns the CTAPHID major version number of *dev*.

The **fido_dev_minor()** function returns the CTAPHID minor version number of *dev*.

For the format and meaning of the CTAPHID parameters returned by functions above, please refer to the FIDO Client to Authenticator Protocol (CTAP) specification.

RETURN VALUES

On success, **fido_dev_open()**, **fido_dev_open_with_info()**, and **fido_dev_close()** 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_set_io_functions(3)`, `fido_init(3)`