

**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 **fid\_dev\_supports\_permissions()** function returns true if *dev* supports CTAP 2.1 UV token permissions.

The **fid\_dev\_supports\_pin()** function returns true if *dev* supports CTAP 2.0 Client PINs.

The **fid\_dev\_supports\_uv()** function returns true if *dev* supports a built-in user verification method.

The **fid\_dev\_has\_pin()** function returns true if *dev* has a CTAP 2.0 Client PIN set.

The **fid\_dev\_has\_uv()** function returns true if *dev* supports built-in user verification and its user verification feature is configured.

The **fid\_dev\_protocol()** function returns the CTAPHID protocol version identifier of *dev*.

The **fid\_dev\_build()** function returns the CTAPHID build version number of *dev*.

The **fid\_dev\_flags()** function returns the CTAPHID capabilities flags of *dev*.

The **fid\_dev\_major()** function returns the CTAPHID major version number of *dev*.

The **fid\_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, **fid\_dev\_open()**, **fid\_dev\_open\_with\_info()**, and **fid\_dev\_close()** return FIDO\_OK. On error, a different error code defined in `<fido/err.h>` is returned.

## SEE ALSO

`fid_dev_info_manifest(3)`, `fid_dev_set_io_functions(3)`, `fid_init(3)`