

**NAME**

**fido\_cred\_set\_authdata**, **fido\_cred\_set\_authdata\_raw**, **fido\_cred\_set\_attstmt**, **fido\_cred\_set\_x509**, **fido\_cred\_set\_sig**, **fido\_cred\_set\_id**, **fido\_cred\_set\_clientdata**, **fido\_cred\_set\_clientdata\_hash**, **fido\_cred\_set\_rp**, **fido\_cred\_set\_user**, **fido\_cred\_set\_extensions**, **fido\_cred\_set\_blob**, **fido\_cred\_set\_pin\_minlen**, **fido\_cred\_set\_prot**, **fido\_cred\_set\_rk**, **fido\_cred\_set\_uv**, **fido\_cred\_set\_fmt**, **fido\_cred\_set\_type** - set parameters of a FIDO2 credential

**SYNOPSIS**

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

```
typedef enum {
```

```
    FIDO_OPT_OMIT = 0, /* use authenticator's default */
```

```
    FIDO_OPT_FALSE, /* explicitly set option to false */
```

```
    FIDO_OPT_TRUE, /* explicitly set option to true */
```

```
} fido_opt_t;
```

```
int
```

```
fido_cred_set_authdata(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_authdata_raw(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_attstmt(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_x509(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_sig(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_id(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_clientdata(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_clientdata_hash(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

```
int
```

```
fido_cred_set_rp(fido_cred_t *cred, const char *id, const char *name);
```

*int*

```
fido_cred_set_user(fido_cred_t *cred, const unsigned char *user_id, size_t user_id_len,
    const char *name, const char *display_name, const char *icon);
```

*int*

```
fido_cred_set_extensions(fido_cred_t *cred, int flags);
```

*int*

```
fido_cred_set_blob(fido_cred_t *cred, const unsigned char *ptr, size_t len);
```

*int*

```
fido_cred_set_pin_minlen(fido_cred_t *cred, size_t len);
```

*int*

```
fido_cred_set_prot(fido_cred_t *cred, int prot);
```

*int*

```
fido_cred_set_rk(fido_cred_t *cred, fido_opt_t rk);
```

*int*

```
fido_cred_set_uv(fido_cred_t *cred, fido_opt_t uv);
```

*int*

```
fido_cred_set_fmt(fido_cred_t *cred, const char *ptr);
```

*int*

```
fido_cred_set_type(fido_cred_t *cred, int cose_alg);
```

## DESCRIPTION

The **fido\_cred\_set\_authdata** set of functions define the various parameters of a FIDO2 credential, allowing a *fido\_cred\_t* type to be prepared for a subsequent call to **fido\_dev\_make\_cred(3)** or **fido\_cred\_verify(3)**. For the complete specification of a FIDO2 credential and the format of its constituent parts, please refer to the Web Authentication (webauthn) standard.

The **fido\_cred\_set\_authdata()**, **fido\_cred\_set\_attstmt()**, **fido\_cred\_set\_x509()**, **fido\_cred\_set\_sig()**, **fido\_cred\_set\_id()**, and **fido\_cred\_set\_clientdata\_hash()** functions set the authenticator data, attestation statement, attestation certificate, attestation signature, id, and client data hash parts of *cred* to *ptr*, where *ptr* points to *len* bytes. A copy of *ptr* is made, and no references to the passed pointer are kept.

The authenticator data passed to **fido\_cred\_set\_authdata()** must be a CBOR-encoded byte string, as obtained from **fido\_cred\_authdata\_ptr()**. Alternatively, a raw binary blob may be passed to **fido\_cred\_set\_authdata\_raw()**. An application calling **fido\_cred\_set\_authdata()** does not need to call **fido\_cred\_set\_id()**. The latter is meant to be used in contexts where the credential's authenticator data is not available.

The attestation statement passed to **fido\_cred\_set\_attstmt()** must be a CBOR-encoded map, as obtained from **fido\_cred\_attstmt\_ptr()**. An application calling **fido\_cred\_set\_attstmt()** does not need to call **fido\_cred\_set\_x509()** or **fido\_cred\_set\_sig()**. The latter two are meant to be used in contexts where the credential's complete attestation statement is not available or required.

The **fido\_cred\_set\_clientdata()** function allows an application to set the client data hash of *cred* by specifying the credential's unhashed client data. This is required by Windows Hello, which calculates the client data hash internally. For compatibility with Windows Hello, applications should use **fido\_cred\_set\_clientdata()** instead of **fido\_cred\_set\_clientdata\_hash()**.

The **fido\_cred\_set\_rp()** function sets the relying party *id* and *name* parameters of *cred*, where *id* and *name* are NUL-terminated UTF-8 strings. The contents of *id* and *name* are copied, and no references to the passed pointers are kept.

The **fido\_cred\_set\_user()** function sets the user attributes of *cred*, where *user\_id* points to *user\_id\_len* bytes and *name*, *display\_name*, and *icon* are NUL-terminated UTF-8 strings. The contents of *user\_id*, *name*, *display\_name*, and *icon* are copied, and no references to the passed pointers are kept. Previously set user attributes are flushed. The *user\_id*, *name*, *display\_name*, and *icon* parameters may be NULL.

The **fido\_cred\_set\_extensions()** function sets the extensions of *cred* to the bitmask *flags*. At the moment, only the FIDO\_EXT\_CRED\_BLOB, FIDO\_EXT\_CRED\_PROTECT, FIDO\_EXT\_HMAC\_SECRET, FIDO\_EXT\_MINPINLEN, and FIDO\_EXT\_LARGELOB\_KEY extensions are supported. If *flags* is zero, the extensions of *cred* are cleared.

The **fido\_cred\_set\_blob()** function sets the "credBlob" to be stored with *cred* to the data pointed to by *ptr*, which must be *len* bytes long.

The **fido\_cred\_set\_pin\_minlen()** function enables the CTAP 2.1 FIDO\_EXT\_MINPINLEN extension on *cred* and sets the expected minimum PIN length of *cred* to *len*, where *len* is greater than zero. If *len* is zero, the FIDO\_EXT\_MINPINLEN extension is disabled on *cred*.

The **fido\_cred\_set\_prot()** function enables the CTAP 2.1 FIDO\_EXT\_CRED\_PROTECT extension on *cred* and sets the protection of *cred* to the scalar *prot*. At the moment, only the FIDO\_CRED\_PROT\_UV\_OPTIONAL, FIDO\_CRED\_PROT\_UV\_OPTIONAL\_WITH\_ID, and

FIDO\_CRED\_PROT\_UV\_REQUIRED protections are supported. If *prot* is zero, the protection of *cred* is cleared.

The **fidocred\_set\_rk()** and **fidocred\_set\_uv()** functions set the *rk* (resident/discoverable key) and *uv* (user verification) attributes of *cred*. Both are FIDO\_OPT\_OMIT by default, allowing the authenticator to use its default settings.

The **fidocred\_set\_fmt()** function sets the attestation statement format identifier of *cred* to *fmt*, where *fmt* must be *packed* (the format used in FIDO2), *fidou2f* (the format used in U2F), *tpm* (the format used by TPM-based authenticators), or *none*. A copy of *fmt* is made, and no references to the passed pointer are kept. Note that not all authenticators support FIDO2 and therefore may only be able to generate *fidou2f* attestation statements.

The **fidocred\_set\_type()** function sets the type of *cred* to *cose\_alg*, where *cose\_alg* is COSE\_ES256, COSE\_ES384, COSE\_RS256, or COSE\_EDDSA. The type of a credential may only be set once. Note that not all authenticators support COSE\_RS256, COSE\_ES384, or COSE\_EDDSA.

Use of the **fidocred\_set\_authdata** set of functions may happen in two distinct situations: when generating a new credential on a FIDO2 device, prior to `fidocred_dev_make_cred(3)` (i.e, in the context of a FIDO2 client), or when validating a generated credential using `fidocred_verify(3)` (i.e, in the context of a FIDO2 server).

For a complete description of the generation of a FIDO2 credential and its verification, please refer to the FIDO2 specification. A concrete utilisation example of the **fidocred\_set\_authdata** set of functions can be found in the *cred.c* example shipped with *libfido2*.

## RETURN VALUES

The error codes returned by the **fidocred\_set\_authdata** set of functions are defined in `<fido/err.h>`. On success, FIDO\_OK is returned.

## SEE ALSO

`fidocred_exclude(3)`, `fidocred_verify(3)`, `fidocred_dev_make_cred(3)`