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_LARGEBLOB_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 **fido_cred_set_rk**() and **fido_cred_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 **fido_cred_set_fmt**() function sets the attestation statement format identifier of *cred* to *fmt*, where *fmt* must be *packed* (the format used in FIDO2), *fido-u2f* (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 *fido-u2f* attestation statements.

The **fido_cred_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 **fido_cred_set_authdata** set of functions may happen in two distinct situations: when generating a new credential on a FIDO2 device, prior to fido_dev_make_cred(3) (i.e, in the context of a FIDO2 client), or when validating a generated credential using fido_cred_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 **fido_cred_set_authdata** set of functions can be found in the *cred.c* example shipped with *libfido2*.

RETURN VALUES

The error codes returned by the **fido_cred_set_authdata** set of functions are defined in <*fido/err.h*>. On success, FIDO_OK is returned.

SEE ALSO

fido_cred_exclude(3), fido_cred_verify(3), fido_dev_make_cred(3)