## NAME

**fido_dev_largeblob_get**, **fido_dev_largeblob_set**, **fido_dev_largeblob_remove**, **fido_dev_largeblob_get_array**, **fido_dev_largeblob_set_array** - FIDO2 large blob API

## SYNOPSIS

**#include <fido.h>**

*int*
**fido_dev_largeblob_get**(*fido_dev_t *dev, const unsigned char *key_ptr, size_t key_len, unsigned char **blob_ptr, size_t *blob_len*);

*int*
**fido_dev_largeblob_set**(*fido_dev_t *dev, const unsigned char *key_ptr, size_t key_len, const unsigned char *blob_ptr, size_t blob_len, const char *pin*);

*int*
**fido_dev_largeblob_remove**(*fido_dev_t *dev, const unsigned char *key_ptr, size_t key_len, const char *pin*);

*int*
**fido_dev_largeblob_get_array**(*fido_dev_t *dev, unsigned char **cbor_ptr, size_t *cbor_len*);

*int*
**fido_dev_largeblob_set_array**(*fido_dev_t *dev, const unsigned char *cbor_ptr, size_t cbor_len, const char *pin*);

## DESCRIPTION

The "largeBlobs" API of *libfido2* allows binary blobs residing on a CTAP 2.1 authenticator to be read, written, and inspected. "largeBlobs" is a CTAP 2.1 extension.

"largeBlobs" are stored as elements of a CBOR array. Confidentiality is ensured by encrypting each element with a distinct, credential-bound 256-bit AES-GCM key. The array is otherwise shared between different credentials and FIDO2 relying parties.

Retrieval of a credential’s encryption key is possible during enrollment with fido_cred_set_extensions(3) and fido_cred_largeblob_key_ptr(3), during assertion with fido_assert_set_extensions(3) and fido_assert_largeblob_key_ptr(3), or, in the case of a resident credential, via *libfido2’s* credential management API.

The "largeBlobs" CBOR array is opaque to the authenticator. Management of the array is left at the

discretion of FIDO2 clients. For further details on CTAP 2.1's "largeBlobs" extension, please refer to the CTAP 2.1 spec.

The **fido_dev_largeblob_get**() function retrieves the authenticator's "largeBlobs" CBOR array and, on success, returns the first blob (iterating from array index zero) that can be decrypted by *key_ptr*, where *key_ptr* points to *key_len* bytes. On success, **fido_dev_largeblob_get**() sets *blob_ptr* to the body of the decrypted blob, and *blob_len* to the length of the decrypted blob in bytes. It is the caller's responsibility to free *blob_ptr*.

The **fido_dev_largeblob_set**() function uses *key_ptr* to encrypt *blob_ptr* and inserts the result in the authenticator's "largeBlobs" CBOR array. Insertion happens at the end of the array if no existing element can be decrypted by *key_ptr*, or at the position of the first element (iterating from array index zero) that can be decrypted by *key_ptr*. *key_len* holds the length of *key_ptr* in bytes, and *blob_len* the length of *blob_ptr* in bytes. A *pin* or equivalent user-verification gesture is required.

The **fido_dev_largeblob_remove**() function retrieves the authenticator's "largeBlobs" CBOR array and, on success, drops the first blob (iterating from array index zero) that can be decrypted by *key_ptr*, where *key_ptr* points to *key_len* bytes. A *pin* or equivalent user-verification gesture is required.

The **fido_dev_largeblob_get_array**() function retrieves the authenticator's "largeBlobs" CBOR array and, on success, sets *cbor_ptr* to the body of the CBOR array, and *cbor_len* to its corresponding length in bytes. It is the caller's responsibility to free *cbor_ptr*.

Finally, the **fido_dev_largeblob_set_array**() function sets the authenticator's "largeBlobs" CBOR array to the data pointed to by *cbor_ptr*, where *cbor_ptr* points to *cbor_len* bytes. A *pin* or equivalent user-verification gesture is required.

## RETURN VALUES

The functions **fido_dev_largeblob_set**(), **fido_dev_largeblob_get**(), **fido_dev_largeblob_remove**(), **fido_dev_largeblob_get_array**(), and **fido_dev_largeblob_set_array**() return FIDO_OK on success. On error, an error code defined in <*fido/err.h*> is returned.

## SEE ALSO

fido_assert_largeblob_key_len(3), fido_assert_largeblob_key_ptr(3), fido_assert_set_extensions(3), fido_cred_largeblob_key_len(3), fido_cred_largeblob_key_ptr(3), fido_cred_set_extensions(3), fido_credman_get_dev_rk(3), fido_credman_get_dev_rp(3), fido_dev_get_assert(3), fido_dev_make_cred(3)

## CAVEATS

The "largeBlobs" extension is not meant to be used to store sensitive data. When retrieved, a

credential’s "largeBlobs" encryption key is transmitted in the clear, and an authenticator’s "largeBlobs" CBOR array can be read without user interaction or verification.