NAME

fido_assert_new, **fido_assert_free**, **fido_assert_count**, **fido_assert_rp_id**, **fido_assert_user_display_name**, **fido_assert_user_icon**, **fido_assert_user_name**, **fido_assert_authdata_ptr**, **fido_assert_blob_ptr**, **fido_assert_clientdata_hash_ptr**, **fido_assert_hmac_secret_ptr**, **fido_assert_largeblob_key_ptr**, **fido_assert_user_id_ptr**, **fido_assert_sig_ptr**, **fido_assert_id_ptr**, **fido_assert_authdata_len**, **fido_assert_blob_len**, **fido_assert_clientdata_hash_len**, **fido_assert_hmac_secret_len**, **fido_assert_largeblob_key_len**, **fido_assert_user_id_len**, **fido_assert_sig_len**, **fido_assert_id_len**, **fido_assert_sigcount**, **fido_assert_flags** - FIDO2 assertion API

SYNOPSIS

#include <fido.h>

*fido_assert_t **
fido_assert_new(*void*);

void
fido_assert_free(*fido_assert_t **assert_p*);

size_t
fido_assert_count(*const fido_assert_t *assert*);

*const char **
fido_assert_rp_id(*const fido_assert_t *assert*);

*const char **
fido_assert_user_display_name(*const fido_assert_t *assert*, *size_t idx*);

*const char **
fido_assert_user_icon(*const fido_assert_t *assert*, *size_t idx*);

*const char **
fido_assert_user_name(*const fido_assert_t *assert*, *size_t idx*);

*const unsigned char **
fido_assert_authdata_ptr(*const fido_assert_t *assert*, *size_t idx*);

*const unsigned char **
fido_assert_clientdata_hash_ptr(*const fido_assert_t *assert*);

*const unsigned char **

fido_assert_blob_ptr(*const fido_assert_t *assert, size_t idx*);

*const unsigned char **
fido_assert_hmac_secret_ptr(*const fido_assert_t *assert, size_t idx*);

*const unsigned char **
fido_assert_largeblob_key_ptr(*const fido_assert_t *assert, size_t idx*);

*const unsigned char **
fido_assert_user_id_ptr(*const fido_assert_t *assert, size_t idx*);

*const unsigned char **
fido_assert_sig_ptr(*const fido_assert_t *assert, size_t idx*);

*const unsigned char **
fido_assert_id_ptr(*const fido_assert_t *assert, size_t idx*);

size_t
fido_assert_authdata_len(*const fido_assert_t *assert, size_t idx*);

size_t
fido_assert_clientdata_hash_len(*const fido_assert_t *assert*);

size_t
fido_assert_blob_len(*const fido_assert_t *assert, size_t idx*);

size_t
fido_assert_hmac_secret_len(*const fido_assert_t *assert, size_t idx*);

size_t
fido_assert_largeblob_key_len(*const fido_assert_t *assert, size_t idx*);

size_t
fido_assert_user_id_len(*const fido_assert_t *assert, size_t idx*);

size_t
fido_assert_sig_len(*const fido_assert_t *assert, size_t idx*);

size_t
fido_assert_id_len(*const fido_assert_t *assert, size_t idx*);

uint32_t
fido_assert_sigcount(*const fido_assert_t *assert, size_t idx*);

uint8_t
fido_assert_flags(*const fido_assert_t *assert, size_t idx*);

DESCRIPTION

A FIDO2 assertion is a collection of statements, each statement a map between a challenge, a credential, a signature, and ancillary attributes. In *libfido2*, a FIDO2 assertion is abstracted by the *fido_assert_t* type. The functions described in this page allow a *fido_assert_t* type to be allocated, deallocated, and inspected. For other operations on *fido_assert_t*, please refer to fido_assert_set_authdata(3), fido_assert_allow_cred(3), fido_assert_verify(3), and fido_dev_get_assert(3).

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

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

The **fido_assert_count**() function returns the number of statements in *assert*.

The **fido_assert_rp_id**() function returns a pointer to a NUL-terminated string holding the relying party ID of *assert*.

The **fido_assert_user_display_name**(), **fido_assert_user_icon**(), and **fido_assert_user_name**(), functions return pointers to the user display name, icon, and name attributes of statement *idx* in *assert*. If not NULL, the values returned by these functions point to NUL-terminated UTF-8 strings. The user display name, icon, and name attributes will typically only be returned by the authenticator if user verification was performed by the authenticator and multiple resident/discoverable credentials were involved in the assertion.

The **fido_assert_authdata_ptr**(), **fido_assert_clientdata_hash_ptr**(), **fido_assert_id_ptr**(), **fido_assert_user_id_ptr**(), **fido_assert_sig_ptr**(), **fido_assert_sigcount**(), and **fido_assert_flags**() functions return pointers to the CBOR-encoded authenticator data, client data hash, credential ID, user ID, signature, signature count, and authenticator data flags of statement *idx* in *assert*.

The **fido_assert_hmac_secret_ptr**() function returns a pointer to the hmac-secret attribute of statement *idx* in *assert*. The HMAC Secret Extension (hmac-secret) is a CTAP 2.0 extension. Note that the resulting hmac-secret varies according to whether user verification was performed by the authenticator.

The **fido_assert_blob_ptr**() and **fido_assert_largeblob_key_ptr**() functions return pointers to the "credBlob" and "largeBlobKey" attributes of statement *idx* in *assert*. Credential Blob (credBlob) and Large Blob Key (largeBlobKey) are CTAP 2.1 extensions.

The **fido_assert_authdata_len**(), **fido_assert_clientdata_hash_len**(), **fido_assert_id_len**(), **fido_assert_user_id_len**(), **fido_assert_sig_len**(), **fido_assert_hmac_secret_len**(), **fido_assert_blob_len**(), and **fido_assert_largeblob_key_len**() functions return the length of a given attribute.

Please note that the first statement in *assert* has an *idx* (index) value of 0.

The authenticator data and signature parts of an assertion statement are typically passed to a FIDO2 server for verification.

RETURN VALUES

The authenticator data returned by **fido_assert_authdata_ptr**() is a CBOR-encoded byte string, as obtained from the authenticator.

The **fido_assert_rp_id**(), **fido_assert_user_display_name**(), **fido_assert_user_icon**(), **fido_assert_user_name**(), **fido_assert_authdata_ptr**(), **fido_assert_clientdata_hash_ptr**(), **fido_assert_id_ptr**(), **fido_assert_user_id_ptr**(), **fido_assert_sig_ptr**(), **fido_assert_hmac_secret_ptr**(), **fido_assert_blob_ptr**(), and **fido_assert_largeblob_key_ptr**() functions may return NULL if the respective field in *assert* is not set. If not NULL, returned pointers are guaranteed to exist until any API function that takes *assert* without the *const* qualifier is invoked.

SEE ALSO

fido_assert_allow_cred(3), fido_assert_set_authdata(3), fido_assert_verify(3), fido_dev_get_assert(3), fido_dev_largeblob_get(3)