

**NAME**

**fid\_assert\_new**, **fid\_assert\_free**, **fid\_assert\_count**, **fid\_assert\_rp\_id**, **fid\_assert\_user\_display\_name**, **fid\_assert\_user\_icon**, **fid\_assert\_user\_name**, **fid\_assert\_authdata\_ptr**, **fid\_assert\_blob\_ptr**, **fid\_assert\_clientdata\_hash\_ptr**, **fid\_assert\_hmac\_secret\_ptr**, **fid\_assert\_largeblob\_key\_ptr**, **fid\_assert\_user\_id\_ptr**, **fid\_assert\_sig\_ptr**, **fid\_assert\_id\_ptr**, **fid\_assert\_authdata\_len**, **fid\_assert\_blob\_len**, **fid\_assert\_clientdata\_hash\_len**, **fid\_assert\_hmac\_secret\_len**, **fid\_assert\_largeblob\_key\_len**, **fid\_assert\_user\_id\_len**, **fid\_assert\_sig\_len**, **fid\_assert\_id\_len**, **fid\_assert\_sigcount**, **fid\_assert\_flags** - FIDO2 assertion API

**SYNOPSIS**

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

```
fid_assert_t *
```

```
fid_assert_new(void);
```

```
void
```

```
fid_assert_free(fid_assert_t **assert_p);
```

```
size_t
```

```
fid_assert_count(const fid_assert_t *assert);
```

```
const char *
```

```
fid_assert_rp_id(const fid_assert_t *assert);
```

```
const char *
```

```
fid_assert_user_display_name(const fid_assert_t *assert, size_t idx);
```

```
const char *
```

```
fid_assert_user_icon(const fid_assert_t *assert, size_t idx);
```

```
const char *
```

```
fid_assert_user_name(const fid_assert_t *assert, size_t idx);
```

```
const unsigned char *
```

```
fid_assert_authdata_ptr(const fid_assert_t *assert, size_t idx);
```

```
const unsigned char *
```

```
fid_assert_clientdata_hash_ptr(const fid_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 **fid\_assert\_blob\_ptr()** and **fid\_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 **fid\_assert\_authdata\_len()**, **fid\_assert\_clientdata\_hash\_len()**, **fid\_assert\_id\_len()**, **fid\_assert\_user\_id\_len()**, **fid\_assert\_sig\_len()**, **fid\_assert\_hmac\_secret\_len()**, **fid\_assert\_blob\_len()**, and **fid\_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 **fid\_assert\_authdata\_ptr()** is a CBOR-encoded byte string, as obtained from the authenticator.

The **fid\_assert\_rp\_id()**, **fid\_assert\_user\_display\_name()**, **fid\_assert\_user\_icon()**, **fid\_assert\_user\_name()**, **fid\_assert\_authdata\_ptr()**, **fid\_assert\_clientdata\_hash\_ptr()**, **fid\_assert\_id\_ptr()**, **fid\_assert\_user\_id\_ptr()**, **fid\_assert\_sig\_ptr()**, **fid\_assert\_hmac\_secret\_ptr()**, **fid\_assert\_blob\_ptr()**, and **fid\_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

**fid\_assert\_allow\_cred(3)**, **fid\_assert\_set\_authdata(3)**, **fid\_assert\_verify(3)**, **fid\_dev\_get\_assert(3)**, **fid\_dev\_largeblob\_get(3)**