

**NAME**

**fido\_cbor\_info\_new**, **fido\_cbor\_info\_free**, **fido\_dev\_get\_cbor\_info**, **fido\_cbor\_info\_aaguid\_ptr**, **fido\_cbor\_info\_extensions\_ptr**, **fido\_cbor\_info\_protocols\_ptr**, **fido\_cbor\_info\_transports\_ptr**, **fido\_cbor\_info\_versions\_ptr**, **fido\_cbor\_info\_options\_name\_ptr**, **fido\_cbor\_info\_options\_value\_ptr**, **fido\_cbor\_info\_algorithm\_type**, **fido\_cbor\_info\_algorithm\_cose**, **fido\_cbor\_info\_algorithm\_count**, **fido\_cbor\_info\_certs\_name\_ptr**, **fido\_cbor\_info\_certs\_value\_ptr**, **fido\_cbor\_info\_certs\_len**, **fido\_cbor\_info\_aaguid\_len**, **fido\_cbor\_info\_extensions\_len**, **fido\_cbor\_info\_protocols\_len**, **fido\_cbor\_info\_transports\_len**, **fido\_cbor\_info\_versions\_len**, **fido\_cbor\_info\_options\_len**, **fido\_cbor\_info\_maxmsgsiz**, **fido\_cbor\_info\_maxcredbloblen**, **fido\_cbor\_info\_maxcredcntlst**, **fido\_cbor\_info\_maxcredidlen**, **fido\_cbor\_info\_maxlargeblob**, **fido\_cbor\_info\_maxrpid\_minpinlen**, **fido\_cbor\_info\_minpinlen**, **fido\_cbor\_info\_fwversion**, **fido\_cbor\_info\_uv\_attempts**, **fido\_cbor\_info\_uv\_modality**, **fido\_cbor\_info\_rk\_remaining**, **fido\_cbor\_info\_new\_pin\_required** - FIDO2 CBOR Info API

**SYNOPSIS**

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

```
fido_cbor_info_t *
```

```
fido_cbor_info_new(void);
```

```
void
```

```
fido_cbor_info_free(fido_cbor_info_t **ci_p);
```

```
int
```

```
fido_dev_get_cbor_info(fido_dev_t *dev, fido_cbor_info_t *ci);
```

```
const unsigned char *
```

```
fido_cbor_info_aaguid_ptr(const fido_cbor_info_t *ci);
```

```
char **
```

```
fido_cbor_info_extensions_ptr(const fido_cbor_info_t *ci);
```

```
const uint8_t *
```

```
fido_cbor_info_protocols_ptr(const fido_cbor_info_t *ci);
```

```
char **
```

```
fido_cbor_info_transports_ptr(const fido_cbor_info_t *ci);
```

```
char **
```

```
fido_cbor_info_versions_ptr(const fido_cbor_info_t *ci);
```

*char \*\**

**fido\_cbor\_info\_options\_name\_ptr**(*const fido\_cbor\_info\_t \*ci*);

*const bool \**

**fido\_cbor\_info\_options\_value\_ptr**(*const fido\_cbor\_info\_t \*ci*);

*const char \**

**fido\_cbor\_info\_algorithm\_type**(*const fido\_cbor\_info\_t \*ci, size\_t idx*);

*int*

**fido\_cbor\_info\_algorithm\_cose**(*const fido\_cbor\_info\_t \*ci, size\_t idx*);

*size\_t*

**fido\_cbor\_info\_algorithm\_count**(*const fido\_cbor\_info\_t \*ci*);

*char \*\**

**fido\_cbor\_info\_certs\_name\_ptr**(*const fido\_cbor\_info\_t \*ci*);

*const uint64\_t \**

**fido\_cbor\_info\_certs\_value\_ptr**(*const fido\_cbor\_info\_t \*ci*);

*size\_t*

**fido\_cbor\_info\_certs\_len**(*const fido\_cbor\_info\_t \*ci*);

*size\_t*

**fido\_cbor\_info\_aaguid\_len**(*const fido\_cbor\_info\_t \*ci*);

*size\_t*

**fido\_cbor\_info\_extensions\_len**(*const fido\_cbor\_info\_t \*ci*);

*size\_t*

**fido\_cbor\_info\_protocols\_len**(*const fido\_cbor\_info\_t \*ci*);

*size\_t*

**fido\_cbor\_info\_transports\_len**(*const fido\_cbor\_info\_t \*ci*);

*size\_t*

**fido\_cbor\_info\_versions\_len**(*const fido\_cbor\_info\_t \*ci*);

*size\_t*

**fido\_cbor\_info\_options\_len**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_maxmsgsiz**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_maxcredbloblen**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_maxcredntlst**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_maxcredidlen**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_maxlargeblob**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_maxrpid\_minpinlen**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_minpinlen**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_fwversion**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_uv\_attempts**(*const fido\_cbor\_info\_t \*ci*);

*uint64\_t*

**fido\_cbor\_info\_uv\_modality**(*const fido\_cbor\_info\_t \*ci*);

*int64\_t*

**fido\_cbor\_info\_rk\_remaining**(*const fido\_cbor\_info\_t \*ci*);

*bool*

**fido\_cbor\_info\_new\_pin\_required**(*const fido\_cbor\_info\_t \*ci*);

## DESCRIPTION

The **fido\_cbor\_info\_new()** function returns a pointer to a newly allocated, empty *fido\_cbor\_info\_t* type.

If memory cannot be allocated, NULL is returned.

The **fido\_cbor\_info\_free()** function releases the memory backing *\*ci\_p*, where *\*ci\_p* must have been previously allocated by **fido\_cbor\_info\_new()**. On return, *\*ci\_p* is set to NULL. Either *ci\_p* or *\*ci\_p* may be NULL, in which case **fido\_cbor\_info\_free()** is a NOP.

The **fido\_dev\_get\_cbor\_info()** function transmits a CTAP\_CBOR\_GETINFO command to *dev* and fills *ci* with attributes retrieved from the command's response. The **fido\_dev\_get\_cbor\_info()** function may block.

The **fido\_cbor\_info\_aaguid\_ptr()**, **fido\_cbor\_info\_extensions\_ptr()**, **fido\_cbor\_info\_protocols\_ptr()**, **fido\_cbor\_info\_transports\_ptr()**, and **fido\_cbor\_info\_versions\_ptr()** functions return pointers to the authenticator attestation GUID, supported extensions, PIN protocol, transports, and CTAP version strings of *ci*. The corresponding length of a given attribute can be obtained by **fido\_cbor\_info\_aaguid\_len()**, **fido\_cbor\_info\_extensions\_len()**, **fido\_cbor\_info\_protocols\_len()**, **fido\_cbor\_info\_transports\_len()**, or **fido\_cbor\_info\_versions\_len()**.

The **fido\_cbor\_info\_options\_name\_ptr()** and **fido\_cbor\_info\_options\_value\_ptr()** functions return pointers to the array of option names and their respective values in *ci*. The length of the options array is returned by **fido\_cbor\_info\_options\_len()**.

The **fido\_cbor\_info\_algorithm\_count()** function returns the number of supported algorithms in *ci*. The **fido\_cbor\_info\_algorithm\_cose()** function returns the COSE identifier of algorithm *idx* in *ci*, or 0 if the COSE identifier is unknown or unset. The **fido\_cbor\_info\_algorithm\_type()** function returns the type of algorithm *idx* in *ci*, or NULL if the type is unset. Please note that the first algorithm in *ci* has an *idx* (index) value of 0.

The **fido\_cbor\_info\_certs\_name\_ptr()** and **fido\_cbor\_info\_certs\_value\_ptr()** functions return pointers to the array of certification names and their respective values in *ci*. The length of the certifications array is returned by **fido\_cbor\_info\_certs\_len()**.

The **fido\_cbor\_info\_maxmsgsiz()** function returns the maximum message size attribute of *ci*.

The **fido\_cbor\_info\_maxcredbloblen()** function returns the maximum "credBlob" length in bytes supported by the authenticator as reported in *ci*.

The **fido\_cbor\_info\_maxcredntlst()** function returns the maximum supported number of credentials in a single credential ID list as reported in *ci*.

The **fido\_cbor\_info\_maxcredidlen()** function returns the maximum supported length of a credential ID

as reported in *ci*.

The **fido\_cbor\_info\_maxrpid\_minpinlen()** function returns the maximum number of RP IDs that may be passed to `fido_dev_set_pin_minlen_rpid(3)`, as reported in *ci*. The minimum PIN length attribute is a CTAP 2.1 addition. If the attribute is not advertised by the authenticator, the **fido\_cbor\_info\_maxrpid\_minpinlen()** function returns zero.

The **fido\_cbor\_info\_maxlargeblob()** function returns the maximum length in bytes of an authenticator's serialized largeBlob array as reported in *ci*.

The **fido\_cbor\_info\_minpinlen()** function returns the minimum PIN length enforced by the authenticator as reported in *ci*. The minimum PIN length attribute is a CTAP 2.1 addition. If the attribute is not advertised by the authenticator, the **fido\_cbor\_info\_minpinlen()** function returns zero.

The **fido\_cbor\_info\_fwversion()** function returns the firmware version attribute of *ci*.

The **fido\_cbor\_info\_uv\_attempts()** function returns the number of UV attempts that the platform may attempt before falling back to PIN authentication. If 1, then all `fido_dev_get_uv_retry_count(3)` retries are handled internally by the authenticator and the platform may only attempt non-PIN UV once. The UV attempts attribute is a CTAP 2.1 addition. If the attribute is not advertised by the authenticator, the **fido\_cbor\_info\_uv\_attempts()** function returns zero.

The **fido\_cbor\_info\_uv\_modality()** function returns a bitmask representing different UV modes supported by the authenticator, as defined in the FIDO Registry of Predefined Values and reported in *ci*. See the `FIDO_UV_MODE_*` definitions in `<fido/param.h>` for the set of values defined by libfido2 and a brief description of each. The UV modality attribute is a CTAP 2.1 addition. If the attribute is not advertised by the authenticator, the **fido\_cbor\_info\_uv\_modality()** function returns zero.

The **fido\_cbor\_info\_rk\_remaining()** function returns the estimated number of additional resident/discoverable credentials that can be stored on the authenticator as reported in *ci*. The estimated number of remaining resident credentials is a CTAP 2.1 addition. If the attribute is not advertised by the authenticator, the **fido\_cbor\_info\_rk\_remaining()** function returns -1.

The **fido\_cbor\_info\_new\_pin\_required()** function returns whether a new PIN is required by the authenticator as reported in *ci*. If **fido\_cbor\_info\_new\_pin\_required()** returns true, operations requiring PIN authentication will fail until a new PIN is set on the authenticator. The `fido_dev_set_pin(3)` function can be used to set a new PIN.

A complete example of how to use these functions can be found in the `example/info.c` file shipped with `libfido2`.

**RETURN VALUES**

The `fido_cbor_info_aaguid_ptr()`, `fido_cbor_info_extensions_ptr()`, `fido_cbor_info_protocols_ptr()`, `fido_cbor_info_transports_ptr()`, `fido_cbor_info_versions_ptr()`, `fido_cbor_info_options_name_ptr()`, and `fido_cbor_info_options_value_ptr()` functions return `NULL` if the respective field in *ci* is absent. If not `NULL`, returned pointers are guaranteed to exist until any API function that takes *ci* without the *const* qualifier is invoked.

**SEE ALSO**

`fido_dev_get_uv_retry_count(3)`, `fido_dev_open(3)`, `fido_dev_set_pin(3)`,  
`fido_dev_set_pin_minlen_rpid(3)`

FIDO Registry of Predefined Values,

<https://fidoalliance.org/specs/common-specs/fido-registry-v2.2-rd-20210525.html>, FIDO Alliance,  
2021-05-25, Review Draft, Version 2.2.