

**NAME**

**fido\_credman\_metadata\_new**, **fido\_credman\_rk\_new**, **fido\_credman\_rp\_new**,  
**fido\_credman\_metadata\_free**, **fido\_credman\_rk\_free**, **fido\_credman\_rp\_free**, **fido\_credman\_rk\_existing**,  
**fido\_credman\_rk\_remaining**, **fido\_credman\_rk**, **fido\_credman\_rk\_count**, **fido\_credman\_rp\_id**,  
**fido\_credman\_rp\_name**, **fido\_credman\_rp\_count**, **fido\_credman\_rp\_id\_hash\_ptr**,  
**fido\_credman\_rp\_id\_hash\_len**, **fido\_credman\_get\_dev\_metadata**, **fido\_credman\_get\_dev\_rk**,  
**fido\_credman\_set\_dev\_rk**, **fido\_credman\_del\_dev\_rk**, **fido\_credman\_get\_dev\_rp** - FIDO2 credential  
management API

**SYNOPSIS**

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

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

```
fido_credman_metadata_t *
```

```
fido_credman_metadata_new(void);
```

```
fido_credman_rk_t *
```

```
fido_credman_rk_new(void);
```

```
fido_credman_rp_t *
```

```
fido_credman_rp_new(void);
```

```
void
```

```
fido_credman_metadata_free(fido_credman_metadata_t **metadata_p);
```

```
void
```

```
fido_credman_rk_free(fido_credman_rk_t **rk_p);
```

```
void
```

```
fido_credman_rp_free(fido_credman_rp_t **rp_p);
```

```
uint64_t
```

```
fido_credman_rk_existing(const fido_credman_metadata_t *metadata);
```

```
uint64_t
```

```
fido_credman_rk_remaining(const fido_credman_metadata_t *metadata);
```

```
const fido_cred_t *
```

```
fido_credman_rk(const fido_credman_rk_t *rk, size_t idx);
```

*size\_t***fido\_credman\_rk\_count**(*const fido\_credman\_rk\_t \*rk*);*const char \****fido\_credman\_rp\_id**(*const fido\_credman\_rp\_t \*rp, size\_t idx*);*const char \****fido\_credman\_rp\_name**(*const fido\_credman\_rp\_t \*rp, size\_t idx*);*size\_t***fido\_credman\_rp\_count**(*const fido\_credman\_rp\_t \*rp*);*const unsigned char \****fido\_credman\_rp\_id\_hash\_ptr**(*const fido\_credman\_rp\_t \*rp, size\_t idx*);*size\_t***fido\_credman\_rp\_id\_hash\_len**(*const fido\_credman\_rp\_t \*, size\_t idx*);*int***fido\_credman\_get\_dev\_metadata**(*fido\_dev\_t \*dev, fido\_credman\_metadata\_t \*metadata,*  
*const char \*pin*);*int***fido\_credman\_get\_dev\_rk**(*fido\_dev\_t \*dev, const char \*rp\_id, fido\_credman\_rk\_t \*rk, const char \*pin*);*int***fido\_credman\_set\_dev\_rk**(*fido\_dev\_t \*dev, fido\_cred\_t \*cred, const char \*pin*);*int***fido\_credman\_del\_dev\_rk**(*fido\_dev\_t \*dev, const unsigned char \*cred\_id, size\_t cred\_id\_len,*  
*const char \*pin*);*int***fido\_credman\_get\_dev\_rp**(*fido\_dev\_t \*dev, fido\_credman\_rp\_t \*rp, const char \*pin*);

## DESCRIPTION

The credential management API of *libfido2* allows resident credentials on a FIDO2 authenticator to be listed, inspected, modified, and removed. Please note that not all FIDO2 authenticators support credential management. To obtain information on what an authenticator supports, please refer to `fido_cbor_info_new(3)`.

The *fido\_credman\_metadata\_t* type abstracts credential management metadata.

The **fido\_credman\_metadata\_new()** function returns a pointer to a newly allocated, empty *fido\_credman\_metadata\_t* type. If memory cannot be allocated, NULL is returned.

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

The **fido\_credman\_get\_dev\_metadata()** function populates *metadata* with information retrieved from *dev*. A valid *pin* must be provided.

The **fido\_credman\_rk\_existing()** function inspects *metadata* and returns the number of resident credentials on the authenticator. The **fido\_credman\_rk\_remaining()** function inspects *metadata* and returns the estimated number of resident credentials that can be created on the authenticator.

The *fido\_credman\_rk\_t* type abstracts the set of resident credentials belonging to a given relying party.

The **fido\_credman\_rk\_new()** function returns a pointer to a newly allocated, empty *fido\_credman\_rk\_t* type. If memory cannot be allocated, NULL is returned.

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

The **fido\_credman\_get\_dev\_rk()** function populates *rk* with the set of resident credentials belonging to *rp\_id* in *dev*. A valid *pin* must be provided.

The **fido\_credman\_rk\_count()** function returns the number of resident credentials in *rk*. The **fido\_credman\_rk()** function returns a pointer to the credential at index *idx* in *rk*. Please note that the first credential in *rk* has an *idx* (index) value of 0.

The **fido\_credman\_set\_dev\_rk()** function updates the credential pointed to by *cred* in *dev*. The credential id and user id attributes of *cred* must be set. See [fido\\_cred\\_set\\_id\(3\)](#) and [fido\\_cred\\_set\\_user\(3\)](#) for details. Only a credential's user attributes (name, display name) may be updated at this time.

The **fido\_credman\_del\_dev\_rk()** function deletes the resident credential identified by *cred\_id* from *dev*, where *cred\_id* points to *cred\_id\_len* bytes. A valid *pin* must be provided.

The *fidoman\_credman\_rp\_t* type abstracts information about a relying party.

The **fidoman\_credman\_rp\_new()** function returns a pointer to a newly allocated, empty *fidoman\_credman\_rp\_t* type. If memory cannot be allocated, NULL is returned.

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

The **fidoman\_credman\_get\_dev\_rp()** function populates *rp* with information about relying parties with resident credentials in *dev*. A valid *pin* must be provided.

The **fidoman\_credman\_rp\_count()** function returns the number of relying parties in *rp*.

The **fidoman\_credman\_rp\_id()** and **fidoman\_credman\_rp\_name()** functions return pointers to the id and name of relying party *idx* in *rp*. If not NULL, the values returned by these functions point to NUL-terminated UTF-8 strings. Please note that the first relying party in *rp* has an *idx* (index) value of 0.

The **fidoman\_credman\_rp\_id\_hash\_ptr()** function returns a pointer to the hashed id of relying party *idx* in *rp*. The corresponding length can be obtained by **fidoman\_credman\_rp\_id\_hash\_len()**. Please note that the first relying party in *rp* has an *idx* (index) value of 0.

## RETURN VALUES

The **fidoman\_credman\_get\_dev\_metadata()**, **fidoman\_credman\_get\_dev\_rk()**, **fidoman\_credman\_set\_dev\_rk()**, **fidoman\_credman\_del\_dev\_rk()**, and **fidoman\_credman\_get\_dev\_rp()** functions return FIDO\_OK on success. On error, a different error code defined in *<fidoman\_err.h>* is returned. Functions returning pointers are not guaranteed to succeed, and should have their return values checked for NULL.

## SEE ALSO

*fidoman\_cbor\_info\_new(3)*, *fidoman\_cred\_new(3)*, *fidoman\_dev\_supports\_credman(3)*

## CAVEATS

Resident credentials are called "discoverable credentials" in CTAP 2.1.