

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_rp_t* type abstracts information about a relying party.

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

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

The **fidoman_get_dev_rp()** function populates *rp* with information about relying parties with resident credentials in *dev*. A valid *pin* must be provided.

The **fidoman_rp_count()** function returns the number of relying parties in *rp*.

The **fidoman_rp_id()** and **fidoman_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_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_rp_id_hash_len()**. Please note that the first relying party in *rp* has an *idx* (index) value of 0.

RETURN VALUES

The **fidoman_get_dev_metadata()**, **fidoman_get_dev_rk()**, **fidoman_set_dev_rk()**, **fidoman_del_dev_rk()**, and **fidoman_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.