

**NAME**

krb5 - Heimdal Kerberos 5 library

**SYNOPSIS**

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_add\_et\_list** (krb5\_context context, void(\*func)(struct et\_list \*\*))

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_password** (krb5\_context context, krb5\_creds \*creds, const char \*newpw, krb5\_principal targprinc, int \*result\_code, krb5\_data \*result\_code\_string, krb5\_data \*result\_string)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_context** (krb5\_context \*context)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_context** (krb5\_context context, krb5\_context \*out)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_free\_context** (krb5\_context context)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_config\_files** (krb5\_context context, char \*\*filenames)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_prepend\_config\_files\_default** (const char \*filelist, char \*\*\*pfilenames)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_default\_config\_files** (char \*\*\*pfilenames)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_free\_config\_files** (char \*\*filenames)

KRB5\_LIB\_FUNCTION const krb5\_etype \*KRB5\_LIB\_CALL **krb5\_kerberos\_etypes** (krb5\_context context)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_default\_in\_tkt\_etypes** (krb5\_context context, const krb5\_etype \*etypes)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_default\_in\_tkt\_etypes** (krb5\_context context, krb5\_pdu pdu\_type, krb5\_etype \*\*etypes)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_init\_ets** (krb5\_context context)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_set\_use\_admin\_kdc** (krb5\_context context, krb5\_boolean flag)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_get\_use\_admin\_kdc** (krb5\_context context)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_add\_extra\_addresses** (krb5\_context context, krb5\_addresses \*addresses)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_extra\_addresses** (krb5\_context context, const krb5\_addresses \*addresses)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_extra\_addresses** (krb5\_context context, krb5\_addresses \*addresses)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_add\_ignore\_addresses** (krb5\_context context, krb5\_addresses \*addresses)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_ignore\_addresses** (krb5\_context

context, const krb5\_addresses \*addresses)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_ignore\_addresses** (krb5\_context context, krb5\_addresses \*addresses)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_fcache\_version** (krb5\_context context, int version)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_fcache\_version** (krb5\_context context, int \*version)  
KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_is\_thread\_safe** (void)  
KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_set\_dns\_canonicalize\_hostname** (krb5\_context context, krb5\_boolean flag)  
KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_get\_dns\_canonicalize\_hostname** (krb5\_context context)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_kdc\_sec\_offset** (krb5\_context context, int32\_t \*sec, int32\_t \*usec)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_kdc\_sec\_offset** (krb5\_context context, int32\_t sec, int32\_t usec)  
KRB5\_LIB\_FUNCTION time\_t KRB5\_LIB\_CALL **krb5\_get\_max\_time\_skew** (krb5\_context context)  
KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_set\_max\_time\_skew** (krb5\_context context, time\_t t)  
KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_set\_home\_dir\_access** (krb5\_context context, krb5\_boolean allow)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_host\_realm** (krb5\_context context, const krb5\_realm \*from, krb5\_realm \*\*to)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_free\_cred\_contents** (krb5\_context context, krb5\_creds \*c)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_creds\_contents** (krb5\_context context, const krb5\_creds \*incred, krb5\_creds \*c)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_creds** (krb5\_context context, const krb5\_creds \*incred, krb5\_creds \*\*outcred)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_free\_creds** (krb5\_context context, krb5\_creds \*c)  
KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_compare\_creds** (krb5\_context context, krb5\_flags whichfields, const krb5\_creds \*mcreds, const krb5\_creds \*creds)  
KRB5\_LIB\_FUNCTION unsigned long KRB5\_LIB\_CALL **krb5\_creds\_get\_ticket\_flags** (krb5\_creds \*creds)  
KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_data\_zero** (krb5\_data \*p)  
KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_data\_free** (krb5\_data \*p)  
KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_free\_data** (krb5\_context context, krb5\_data \*p)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_data\_alloc** (krb5\_data \*p, int len)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_data\_realloc** (krb5\_data \*p, int len)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_data\_copy** (krb5\_data \*p, const void \*data, size\_t len)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_data** (krb5\_context context, const krb5\_data \*indata, krb5\_data \*\*outdata)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_data\_cmp** (const krb5\_data \*data1, const krb5\_data \*data2)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_data\_ct\_cmp** (const krb5\_data \*data1, const krb5\_data \*data2)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_krbhst\_get\_addrinfo** (krb5\_context context, krb5\_krbhst\_info \*host, struct addrinfo \*\*ai)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_free\_ticket** (krb5\_context context, krb5\_ticket \*ticket)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_ticket** (krb5\_context context, const krb5\_ticket \*from, krb5\_ticket \*\*to)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ticket\_get\_client** (krb5\_context context, const krb5\_ticket \*ticket, krb5\_principal \*client)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ticket\_get\_server** (krb5\_context context, const krb5\_ticket \*ticket, krb5\_principal \*server)

KRB5\_LIB\_FUNCTION time\_t KRB5\_LIB\_CALL **krb5\_ticket\_get\_endtime** (krb5\_context context, const krb5\_ticket \*ticket)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ticket\_get\_authorization\_data\_type** (krb5\_context context, krb5\_ticket \*ticket, int type, krb5\_data \*data)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_set\_real\_time** (krb5\_context context, krb5\_timestamp sec, int32\_t usec)

## Detailed Description

### Function Documentation

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_add\_et\_list** (krb5\_context context, void(\*)**(struct et\_list \*\*)** func)

Add a specified list of error messages to the et list in context. Call func (probably a comerr-generated function) with a pointer to the current et\_list.

#### Parameters:

*context* A kerberos context.  
*func* The generated com\_err et function.

#### Returns:

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_add\_extra\_addresses (krb5\_context context, krb5\_addresses \* addresses)**

Add extra address to the address list that the library will add to the client's address list when communicating with the KDC.

**Parameters:**

*context* Kerberos 5 context.

*addresses* addresses to add

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_add\_ignore\_addresses (krb5\_context context, krb5\_addresses \* addresses)**

Add extra addresses to ignore when fetching addresses from the underlying operating system.

**Parameters:**

*context* Kerberos 5 context.

*addresses* addresses to ignore

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_compare\_creds (krb5\_context context, krb5\_flags whichfields, const krb5\_creds \* mcreds, const krb5\_creds \* creds)**

Return TRUE if 'mcreds' and 'creds' are equal ('whichfields' determines what equal means).

The following flags, set in whichfields affects the comparison:

- ⊕ KRB5\_TC\_MATCH\_SRV\_NAMEONLY Consider all realms equal when comparing the service principal.
- ⊕ KRB5\_TC\_MATCH\_KEYTYPE Compare enctypees.
- ⊕ KRB5\_TC\_MATCH\_FLAGS\_EXACT Make sure that the ticket flags are identical.
- ⊕ KRB5\_TC\_MATCH\_FLAGS Make sure that all ticket flags set in mcreds are also present in creds .

- ⊕ `KRB5_TC_MATCH_TIMES_EXACT` Compares the ticket times exactly.
- ⊕ `KRB5_TC_MATCH_TIMES` Compares only the expiration times of the creds.
- ⊕ `KRB5_TC_MATCH_AUTHDATA` Compares the authdata fields.
- ⊕ `KRB5_TC_MATCH_2ND_TKT` Compares the second tickets (used by user-to-user authentication).
- ⊕ `KRB5_TC_MATCH_IS_SKEY` Compares the existence of the second ticket.

**Parameters:**

*context* Kerberos 5 context.  
*whichfields* which fields to compare.  
*mcreds* cred to compare with.  
*creds* cred to compare with.

**Returns:**

return TRUE if mcred and creds are equal, FALSE if not.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_context (krb5\_context context, krb5\_context \* out)**

Make a copy for the Kerberos 5 context, the new `krb5_context` should be freed with `krb5_free_context()`.

**Parameters:**

*context* the Kerberos context to copy  
*out* the copy of the Kerberos, set to NULL error.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_creds (krb5\_context context, const krb5\_creds \* incred, krb5\_creds \*\* outcred)**

Copy `krb5_creds`.

**Parameters:**

*context* Kerberos 5 context.  
*incred* source credential  
*outcred* destination credential, free with `krb5_free_creds()`.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_creds\_contents (krb5\_context context, const krb5\_creds \* incred, krb5\_creds \* c)**

Copy content of `krb5_creds`.

**Parameters:**

*context* Kerberos 5 context.

*incred* source credential

*c* destination credential, free with `krb5_free_cred_contents()`.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_data (krb5\_context context, const krb5\_data \* indata, krb5\_data \*\* outdata)**

Copy the data into a newly allocated `krb5_data`.

**Parameters:**

*context* Kerberos 5 context.

*indata* the `krb5_data` data to copy

*outdata* new `krb5_data` to copy too. Free with `krb5_free_data()`.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_host\_realm (krb5\_context context, const krb5\_realm \* from, krb5\_realm \*\* to)**

Copy the list of realms from 'from' to 'to'.

**Parameters:**

*context* Kerberos 5 context.

*from* list of realms to copy from.

*to* list of realms to copy to, free list of `krb5_free_host_realm()`.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see

krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION** **krb5\_error\_code** **KRB5\_LIB\_CALL** **krb5\_copy\_ticket** (**krb5\_context** context,  
**const krb5\_ticket \* from, krb5\_ticket \*\* to**)

Copy ticket and content

**Parameters:**

*context* a Kerberos 5 context

*from* ticket to copy

*to* new copy of ticket, free with **krb5\_free\_ticket()**

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see

**krb5\_get\_error\_message()**.

**KRB5\_LIB\_FUNCTION** **unsigned long** **KRB5\_LIB\_CALL** **krb5\_creds\_get\_ticket\_flags** (**krb5\_creds \***  
**creds**)

Returns the ticket flags for the credentials in creds. See also **krb5\_ticket\_get\_flags()**.

**Parameters:**

*creds* credential to get ticket flags from

**Returns:**

ticket flags

**KRB5\_LIB\_FUNCTION** **krb5\_error\_code** **KRB5\_LIB\_CALL** **krb5\_data\_alloc** (**krb5\_data \* p, int len**)

Allocate data of and **krb5\_data**.

**Parameters:**

*p* **krb5\_data** to allocate.

*len* size to allocate.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned.

**KRB5\_LIB\_FUNCTION** **int** **KRB5\_LIB\_CALL** **krb5\_data\_cmp** (**const krb5\_data \* data1, const**  
**krb5\_data \* data2**)

Compare to data.

**Parameters:**

*data1* krb5\_data to compare  
*data2* krb5\_data to compare

**Returns:**

return the same way as memcmp(), useful when sorting.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_data\_copy (krb5\_data \* p, const void \* data, size\_t len)**

Copy the data of len into the krb5\_data.

**Parameters:**

*p* krb5\_data to copy into.  
*data* data to copy..  
*len* new size.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned.

**KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL krb5\_data\_ct\_cmp (const krb5\_data \* data1, const krb5\_data \* data2)**

Compare to data not exposing timing information from the checksum data

**Parameters:**

*data1* krb5\_data to compare  
*data2* krb5\_data to compare

**Returns:**

returns zero for same data, otherwise non zero.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_data\_free (krb5\_data \* p)**

Free the content of krb5\_data structure, its ok to free a zeroed structure (with memset() or **krb5\_data\_zero()**). When done, the structure will be zeroed. The same function is called **krb5\_free\_data\_contents()** in MIT Kerberos.

**Parameters:**

*p* krb5\_data to free.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_data\_realloc (krb5\_data \* p, int len)**

Grow (or shrink) the content of krb5\_data to a new size.



**Parameters:**

*p* krb5\_data to free.  
*len* new size.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_data\_zero (krb5\_data \* p)**

Reset the (potentially uninitialized) krb5\_data structure.

**Parameters:**

*p* krb5\_data to reset.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_free\_config\_files (char \*\* filenames)**

Free a list of configuration files.

**Parameters:**

*filenames* list, terminated with a NULL pointer, to be freed. NULL is an valid argument.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_free\_context (krb5\_context context)**

Frees the krb5\_context allocated by krb5\_init\_context().

**Parameters:**

*context* context to be freed.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_free\_cred\_contents (krb5\_context context, krb5\_creds \* c)**

Free content of krb5\_creds.

**Parameters:**

*context* Kerberos 5 context.  
*c* krb5\_creds to free.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION** `krb5_error_code KRB5_LIB_CALL krb5_free_creds (krb5_context context, krb5_creds * c)`

Free `krb5_creds`.

**Parameters:**

*context* Kerberos 5 context.

*c* `krb5_creds` to free.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION** `void KRB5_LIB_CALL krb5_free_data (krb5_context context, krb5_data * p)`

Free `krb5_data` (and its content).

**Parameters:**

*context* Kerberos 5 context.

*p* `krb5_data` to free.

**KRB5\_LIB\_FUNCTION** `krb5_error_code KRB5_LIB_CALL krb5_free_ticket (krb5_context context, krb5_ticket * ticket)`

Free ticket and content

**Parameters:**

*context* a Kerberos 5 context

*ticket* ticket to free

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION** `krb5_error_code KRB5_LIB_CALL krb5_get_default_config_files (char *** pfilenames)`

Get the global configuration list.

**Parameters:**

*pfilenames* return array of filenames, should be freed with `krb5_free_config_files()`.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see

krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_get\_default\_in\_tkt\_etypes**  
(krb5\_context context, krb5\_pdu pdu\_type, krb5\_etype \*\* etypes)

Get the default encryption types that will be use in communcation with the KDC, clients and servers.

**Parameters:**

*context* Kerberos 5 context.

*etypes* Encryption types, array terminated with ETYPE\_NULL(0), caller should free array with  
krb5\_xfree():

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see  
krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_get\_dns\_canonicalize\_hostname**  
(krb5\_context context)

Get if the library uses DNS to canonicalize hostnames.

**Parameters:**

*context* Kerberos 5 context.

**Returns:**

return non zero if the library uses DNS to canonicalize hostnames.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_get\_extra\_addresses** (krb5\_context  
context, krb5\_addresses \* addresses)

Get extra address to the address list that the library will add to the client's address list when  
communicating with the KDC.

**Parameters:**

*context* Kerberos 5 context.

*addresses* addreses to set

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see  
krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_get\_fcache\_version** (krb5\_context  
context, int \* version)

Get version of fcache that the library should use.

**Parameters:**

*context* Kerberos 5 context.

*version* version number.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_get\_ignore\_addresses (krb5\_context context, krb5\_addresses \* addresses)**

Get extra addresses to ignore when fetching addresses from the underlying operating system.

**Parameters:**

*context* Kerberos 5 context.

*addresses* list addresses ignored

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_get\_kdc\_sec\_offset (krb5\_context context, int32\_t \* sec, int32\_t \* usec)**

Get current offset in time to the KDC.

**Parameters:**

*context* Kerberos 5 context.

*sec* seconds part of offset.

*usec* micro seconds part of offset.

**Returns:**

returns zero

**KRB5\_LIB\_FUNCTION time\_t KRB5\_LIB\_CALL krb5\_get\_max\_time\_skew (krb5\_context context)**

Get max time skew allowed.

**Parameters:**

*context* Kerberos 5 context.

**Returns:**

timeskew in seconds.

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_get\_use\_admin\_kdc (krb5\_context context)**

Make the kerberos library default to the admin KDC.

**Parameters:**

*context* Kerberos 5 context.

**Returns:**

boolean flag to telling the context will use admin KDC as the default KDC.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_init\_context (krb5\_context \* context)**

Initializes the context structure and reads the configuration file /etc/krb5.conf. The structure should be freed by calling **krb5\_free\_context()** when it is no longer being used.

**Parameters:**

*context* pointer to returned context

**Returns:**

Returns 0 to indicate success. Otherwise an errno code is returned. Failure means either that something bad happened during initialization (typically ENOMEM) or that Kerberos should not be used ENXIO.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_init\_ets (krb5\_context context)**

Init the built-in ets in the Kerberos library.

**Parameters:**

*context* kerberos context to add the ets too

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_is\_thread\_safe (void)**

Runtime check if the Kerberos library was compiled with thread support.

**Returns:**

TRUE if the library was compiled with thread support, FALSE if not.

**KRB5\_LIB\_FUNCTION const krb5\_enctype\* KRB5\_LIB\_CALL krb5\_kerberos\_enctypes (krb5\_context context)**

Returns the list of Kerberos encryption types sorted in order of most preferred to least preferred encryption type. Note that some encryption types might be disabled, so you need to check with **krb5\_etype\_valid()** before using the encryption type.

**Returns:**

list of entypes, terminated with ETYPE\_NULL. Its a static array completed into the Kerberos library so the content doesn't need to be freed.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_krbhst\_get\_addrinfo (krb5\_context context, krb5\_krbhst\_info \* host, struct addrinfo \*\* ai)**

Return an 'struct addrinfo \*' for a KDC host.

Returns an the struct addrinfo in in that corresponds to the information in 'host'. free:ing is handled by **krb5\_krbhst\_free**, so the returned ai must not be released.

First try this as an IP address, this allows us to add a dot at the end to stop using the search domains.

If the hostname contains a dot, assumes it's a FQDN and don't use search domains since that might be painfully slow when machine is disconnected from that network.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_prepend\_config\_files\_default (const char \* filelist, char \*\*\* pfilenames)**

Prepend the filename to the global configuration list.

**Parameters:**

*filelist* a filename to add to the default list of filename

*pfilenames* return array of filenames, should be freed with **krb5\_free\_config\_files()**.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see **krb5\_get\_error\_message()**.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_config\_files (krb5\_context context, char \*\* filenames)**

Reinit the context from a new set of filenames.

**Parameters:**

*context* context to add configuration too.

*filenames* array of filenames, end of list is indicated with a NULL filename.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_default\_in\_tkt\_etypes (krb5\_context context, const krb5\_enctype \* etypes)**

Set the default encryption types that will be use in communcation with the KDC, clients and servers.

**Parameters:**

*context* Kerberos 5 context.

*etypes* Encryption types, array terminated with ETYPE\_NULL (0).

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_set\_dns\_canonicalize\_hostname (krb5\_context context, krb5\_boolean flag)**

Set if the library should use DNS to canonicalize hostnames.

**Parameters:**

*context* Kerberos 5 context.

*flag* if its dns canonicalizion is used or not.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_extra\_addresses (krb5\_context context, const krb5\_addresses \* addresses)**

Set extra address to the address list that the library will add to the client's address list when communicating with the KDC.

**Parameters:**

*context* Kerberos 5 context.

*addresses* addreses to set

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_fcache\_version (krb5\_context context, int version)**

Set version of fcache that the library should use.

**Parameters:**

*context* Kerberos 5 context.

*version* version number.

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_set\_home\_dir\_access (krb5\_context context, krb5\_boolean allow)**

Enable and disable home directory access on either the global state or the `krb5_context` state. By calling `krb5_set_home_dir_access()` with `context` set to `NULL`, the global state is configured otherwise the state for the `krb5_context` is modified.

For home directory access to be allowed, both the global state and the `krb5_context` state have to be allowed.

Administrator (root user), never uses the home directory.

**Parameters:**

*context* a Kerberos 5 context or `NULL`

*allow* allow if `TRUE` home directory

**Returns:**

the old value

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_ignore\_addresses (krb5\_context context, const krb5\_addresses \* addresses)**

Set extra addresses to ignore when fetching addresses from the underlying operating system.

**Parameters:**

*context* Kerberos 5 context.

*addresses* addreses to ignore

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_kdc\_sec\_offset (krb5\_context context, int32\_t sec, int32\_t usec)**



Set current offset in time to the KDC.

**Parameters:**

*context* Kerberos 5 context.

*sec* seconds part of offset.

*usec* micro seconds part of offset.

**Returns:**

returns zero

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_set\_max\_time\_skew (krb5\_context context, time\_t t)**

Set max time skew allowed.

**Parameters:**

*context* Kerberos 5 context.

*t* timeskew in seconds.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_password (krb5\_context context, krb5\_creds \* creds, const char \* newpw, krb5\_principal targprinc, int \* result\_code, krb5\_data \* result\_code\_string, krb5\_data \* result\_string)**

Change password using creds.

**Parameters:**

*context* a Keberos context

*creds* The initial kadmin/passwd for the principal or an admin principal

*newpw* The new password to set

*targprinc* if unset, the default principal is used.

*result\_code* Result code, KRB5\_KPASSWD\_SUCCESS is when password is changed.

*result\_code\_string* binary message from the server, contains at least the result\_code.

*result\_string* A message from the kpasswd service or the library in human printable form. The string is NUL terminated.

**Returns:**

On success and \*result\_code is KRB5\_KPASSWD\_SUCCESS, the password is changed.

@

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_set\_real\_time (krb5\_context context, krb5\_timestamp sec, int32\_t usec)**

Set the absolute time that the caller knows the kdc has so the kerberos library can calculate the relative difference between the KDC time and local system time.

**Parameters:**

*context* Kerberos 5 context.

*sec* The applications new of 'now' in seconds

*usec* The applications new of 'now' in micro seconds

**Returns:**

Kerberos 5 error code, see `krb5_get_error_message()`.

If the caller passes in a negative usec, its assumed to be unknown and the function will use the current time usec.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_set\_use\_admin\_kdc (krb5\_context context, krb5\_boolean flag)**

Make the kerberos library default to the admin KDC.

**Parameters:**

*context* Kerberos 5 context.

*flag* boolean flag to select if the use the admin KDC or not.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ticket\_get\_authorization\_data\_type (krb5\_context context, krb5\_ticket \* ticket, int type, krb5\_data \* data)**

Extract the authorization data type of type from the ticket. Store the field in data. This function is to use for kerberos applications.

**Parameters:**

*context* a Kerberos 5 context

*ticket* Kerberos ticket

*type* type to fetch

*data* returned data, free with `krb5_data_free()`

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ticket\_get\_client (krb5\_context context, const krb5\_ticket \* ticket, krb5\_principal \* client)**

Return client principal in ticket

**Parameters:**

*context* a Kerberos 5 context

*ticket* ticket to copy

*client* client principal, free with **krb5\_free\_principal()**

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see **krb5\_get\_error\_message()**.

**KRB5\_LIB\_FUNCTION time\_t KRB5\_LIB\_CALL krb5\_ticket\_get\_endtime (krb5\_context context, const krb5\_ticket \* ticket)**

Return end time of ticket

**Parameters:**

*context* a Kerberos 5 context

*ticket* ticket to copy

**Returns:**

end time of ticket

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ticket\_get\_server (krb5\_context context, const krb5\_ticket \* ticket, krb5\_principal \* server)**

Return server principal in ticket

**Parameters:**

*context* a Kerberos 5 context

*ticket* ticket to copy

*server* server principal, free with **krb5\_free\_principal()**

**Returns:**

Returns 0 to indicate success. Otherwise an kerberos et error code is returned, see **krb5\_get\_error\_message()**.