

**NAME**

Heimdal Kerberos 5 storage functions -

**Functions**

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_storage\_set\_flags** (krb5\_storage \*sp, krb5\_flags flags)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_storage\_clear\_flags** (krb5\_storage \*sp, krb5\_flags flags)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_storage\_is\_flags** (krb5\_storage \*sp, krb5\_flags flags)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_storage\_set\_byteorder** (krb5\_storage \*sp, krb5\_flags byteorder)

KRB5\_LIB\_FUNCTION krb5\_flags KRB5\_LIB\_CALL **krb5\_storage\_get\_byteorder** (krb5\_storage \*sp)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_storage\_set\_max\_alloc** (krb5\_storage \*sp, size\_t size)

KRB5\_LIB\_FUNCTION off\_t KRB5\_LIB\_CALL **krb5\_storage\_seek** (krb5\_storage \*sp, off\_t offset, int whence)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_storage\_truncate** (krb5\_storage \*sp, off\_t offset)

KRB5\_LIB\_FUNCTION krb5\_ssize\_t KRB5\_LIB\_CALL **krb5\_storage\_read** (krb5\_storage \*sp, void \*buf, size\_t len)

KRB5\_LIB\_FUNCTION krb5\_ssize\_t KRB5\_LIB\_CALL **krb5\_storage\_write** (krb5\_storage \*sp, const void \*buf, size\_t len)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_storage\_set\_eof\_code** (krb5\_storage \*sp, int code)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_storage\_get\_eof\_code** (krb5\_storage \*sp)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_storage\_free** (krb5\_storage \*sp)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_storage\_to\_data** (krb5\_storage \*sp, krb5\_data \*data)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_int32** (krb5\_storage \*sp, int32\_t value)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_uint32** (krb5\_storage \*sp, uint32\_t value)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_int32** (krb5\_storage \*sp, int32\_t \*value)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_uint32** (krb5\_storage \*sp, uint32\_t \*value)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_int16** (krb5\_storage \*sp, int16\_t value)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_uint16** (krb5\_storage \*sp, uint16\_t value)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_int16** (krb5\_storage \*sp, int16\_t

\*value)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_uint16** (krb5\_storage \*sp, uint16\_t  
\*value)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_int8** (krb5\_storage \*sp, int8\_t  
value)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_uint8** (krb5\_storage \*sp, uint8\_t  
value)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_int8** (krb5\_storage \*sp, int8\_t  
\*value)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_uint8** (krb5\_storage \*sp, uint8\_t  
\*value)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_data** (krb5\_storage \*sp,  
krb5\_data data)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_data** (krb5\_storage \*sp, krb5\_data  
\*data)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_string** (krb5\_storage \*sp, const  
char \*s)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_string** (krb5\_storage \*sp, char  
\*\*string)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_stringz** (krb5\_storage \*sp, const  
char \*s)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_stringz** (krb5\_storage \*sp, char  
\*\*string)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_principal** (krb5\_storage \*sp,  
krb5\_const\_principal p)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_principal** (krb5\_storage \*sp,  
krb5\_principal \*princ)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_keyblock** (krb5\_storage \*sp,  
krb5\_keyblock p)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_keyblock** (krb5\_storage \*sp,  
krb5\_keyblock \*p)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_times** (krb5\_storage \*sp,  
krb5\_times times)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_times** (krb5\_storage \*sp,  
krb5\_times \*times)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_address** (krb5\_storage \*sp,  
krb5\_address p)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_ret\_address** (krb5\_storage \*sp,  
krb5\_address \*adr)  
KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_store\_addrs** (krb5\_storage \*sp,

```

krb5_addresses p)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_ret_addrs (krb5_storage *sp,
krb5_addresses *adr)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_store_authdata (krb5_storage *sp,
krb5_authdata auth)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_ret_authdata (krb5_storage *sp,
krb5_authdata *auth)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_store_creds (krb5_storage *sp,
krb5_creds *creds)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_ret_creds (krb5_storage *sp,
krb5_creds *creds)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_store_creds_tag (krb5_storage *sp,
krb5_creds *creds)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_ret_creds_tag (krb5_storage *sp,
krb5_creds *creds)
KRB5_LIB_FUNCTION krb5_storage *KRB5_LIB_CALL krb5_storage_emem (void)
KRB5_LIB_FUNCTION krb5_storage *KRB5_LIB_CALL krb5_storage_from_fd (krb5_socket_t fd_in)
KRB5_LIB_FUNCTION krb5_storage *KRB5_LIB_CALL krb5_storage_from_mem (void *buf, size_t
len)
KRB5_LIB_FUNCTION krb5_storage *KRB5_LIB_CALL krb5_storage_from_data (krb5_data *data)
KRB5_LIB_FUNCTION krb5_storage *KRB5_LIB_CALL krb5_storage_from_readonly_mem (const
void *buf, size_t len)

```

## Detailed Description

### Function Documentation

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL *krb5\_ret\_address*** (*krb5\_storage \* sp*,  
*krb5\_address \* adr*)

Read a address block from the storage.

#### Parameters:

*sp* the storage buffer to write to  
*adr* the address block read from storage

#### Returns:

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL *krb5\_ret\_addrs*** (*krb5\_storage \* sp*,  
*krb5\_addresses \* adr*)

Read a addresses block from the storage.

**Parameters:**

*sp* the storage buffer to write to  
*adr* the addresses block read from storage

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_authdata (krb5\_storage \* sp, krb5\_authdata \* auth)**

Read a auth data from the storage.

**Parameters:**

*sp* the storage buffer to write to  
*auth* the auth data block read from storage

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_creds (krb5\_storage \* sp, krb5\_creds \* creds)**

Read a credentials block from the storage.

**Parameters:**

*sp* the storage buffer to write to  
*creds* the credentials block read from storage

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_creds\_tag (krb5\_storage \* sp, krb5\_creds \* creds)**

Read a tagged credentials block from the storage.

**Parameters:**

*sp* the storage buffer to write to  
*creds* the credentials block read from storage

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_data (krb5\_storage \* sp, krb5\_data \* data)**

Parse a data from the storage.

**Parameters:**

*sp* the storage buffer to read from

*data* the parsed data

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_int16 (krb5\_storage \* sp, int16\_t \* value)**

Read a int16 from storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too

*value* the value read from the buffer

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_int32 (krb5\_storage \* sp, int32\_t \* value)**

Read a int32 from storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too

*value* the value read from the buffer

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_int8 (krb5\_storage \* sp, int8\_t \* value)**

Read a int8 from storage

**Parameters:**

*sp* the storage to write too  
*value* the value read from the buffer

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_keyblock (krb5\_storage \* sp, krb5\_keyblock \* p)**

Read a keyblock from the storage.

**Parameters:**

*sp* the storage buffer to write to  
*p* the keyblock read from storage, free using **krb5\_free\_keyblock()**

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_principal (krb5\_storage \* sp, krb5\_principal \* princ)**

Parse principal from the storage.

**Parameters:**

*sp* the storage buffer to read from  
*princ* the parsed principal

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_string (krb5\_storage \* sp, char \*\* string)**

Parse a string from the storage.

**Parameters:**

*sp* the storage buffer to read from  
*string* the parsed string

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_stringz (krb5\_storage \* sp, char**

**\*\* string)**

Parse zero terminated string from the storage.

**Parameters:**

*sp* the storage buffer to read from

*string* the parsed string

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_times (krb5\_storage \* sp, krb5\_times \* times)**

Read a times block from the storage.

**Parameters:**

*sp* the storage buffer to write to

*times* the times block read from storage

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_uint16 (krb5\_storage \* sp, uint16\_t \* value)**

Read a int16 from storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too

*value* the value read from the buffer

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_uint32 (krb5\_storage \* sp, uint32\_t \* value)**

Read a uint32 from storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too

*value* the value read from the buffer

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_ret\_uint8 (krb5\_storage \* sp, uint8\_t \* value)**

Read a uint8 from storage

**Parameters:**

*sp* the storage to write too

*value* the value read from the buffer

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_storage\_clear\_flags (krb5\_storage \* sp, krb5\_flags flags)**

Clear the flags on a storage buffer

**Parameters:**

*sp* the storage buffer to clear the flags on

*flags* the flags to clear

**KRB5\_LIB\_FUNCTION krb5\_storage\* KRB5\_LIB\_CALL krb5\_storage\_emem (void)**

Create a elastic (allocating) memory storage backend. Memory is allocated on demand. Free returned krb5\_storage with **krb5\_storage\_free()**.

**Returns:**

A krb5\_storage on success, or NULL on out of memory error.

**See also:**

**krb5\_storage\_from\_mem()**

**krb5\_storage\_from\_readonly\_mem()**

**krb5\_storage\_from\_fd()**

**krb5\_storage\_from\_data()**



**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_storage\_free (krb5\_storage \* sp)**

Free a krb5 storage.

**Parameters:**

*sp* the storage to free.

**Returns:**

An Kerberos 5 error code.

**KRB5\_LIB\_FUNCTION krb5\_storage\* KRB5\_LIB\_CALL krb5\_storage\_from\_data (krb5\_data \* data)**

Create a fixed size memory storage block

**Returns:**

A krb5\_storage on success, or NULL on out of memory error.

**See also:**

krb5\_storage\_mem()

krb5\_storage\_from\_mem()

krb5\_storage\_from\_readonly\_mem()

krb5\_storage\_from\_fd()

**KRB5\_LIB\_FUNCTION krb5\_storage\* KRB5\_LIB\_CALL krb5\_storage\_from\_fd (krb5\_socket\_t fd\_in)**

**Returns:**

A krb5\_storage on success, or NULL on out of memory error.

**See also:**

krb5\_storage\_emem()

krb5\_storage\_from\_mem()

krb5\_storage\_from\_readonly\_mem()

krb5\_storage\_from\_data()

**KRB5\_LIB\_FUNCTION krb5\_storage\* KRB5\_LIB\_CALL krb5\_storage\_from\_mem (void \* buf, size\_t len)**

Create a fixed size memory storage block

**Returns:**

A `krb5_storage` on success, or `NULL` on out of memory error.

**See also:**

`krb5_storage_mem()`

`krb5_storage_from_readonly_mem()`

`krb5_storage_from_data()`

`krb5_storage_from_fd()`

**KRB5\_LIB\_FUNCTION** `krb5_storage*` **KRB5\_LIB\_CALL** `krb5_storage_from_readonly_mem` (**const void \* buf**, **size\_t len**)

Create a fixed size memory storage block that is read only

**Returns:**

A `krb5_storage` on success, or `NULL` on out of memory error.

**See also:**

`krb5_storage_mem()`

`krb5_storage_from_mem()`

`krb5_storage_from_data()`

`krb5_storage_from_fd()`

**KRB5\_LIB\_FUNCTION** `krb5_flags` **KRB5\_LIB\_CALL** `krb5_storage_get_byteorder` (`krb5_storage * sp`)

Return the current byteorder for the buffer. See `krb5_storage_set_byteorder()` for the list of byte order constants.

**KRB5\_LIB\_FUNCTION** `int` **KRB5\_LIB\_CALL** `krb5_storage_get_eof_code` (`krb5_storage * sp`)

Get the return code that will be used when end of storage is reached.

**Parameters:**

*sp* the storage

**Returns:**

storage error code

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_storage\_is\_flags (krb5\_storage \* sp, krb5\_flags flags)**

Return true or false depending on if the storage flags is set or not. NB testing for the flag 0 always return true.

**Parameters:**

*sp* the storage buffer to check flags on  
*flags* The flags to test for

**Returns:**

true if all the flags are set, false if not.

**KRB5\_LIB\_FUNCTION krb5\_ssize\_t KRB5\_LIB\_CALL krb5\_storage\_read (krb5\_storage \* sp, void \* buf, size\_t len)**

Read to the storage buffer.

**Parameters:**

*sp* the storage buffer to read from  
*buf* the buffer to store the data in  
*len* the length to read

**Returns:**

The length of data read (can be shorter then len), or negative on error.

**KRB5\_LIB\_FUNCTION off\_t KRB5\_LIB\_CALL krb5\_storage\_seek (krb5\_storage \* sp, off\_t offset, int whence)**

Seek to a new offset.

**Parameters:**

*sp* the storage buffer to seek in.  
*offset* the offset to seek  
*whence* relative searching, SEEK\_CUR from the current position, SEEK\_END from the end, SEEK\_SET absolute from the start.

**Returns:**

The new current offset

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_storage\_set\_byteorder (krb5\_storage \* sp, krb5\_flags byteorder)**

Set the new byte order of the storage buffer.

**Parameters:**

*sp* the storage buffer to set the byte order for.  
*byteorder* the new byte order.

The byte order are: KRB5\_STORAGE\_BYTEORDER\_BE, KRB5\_STORAGE\_BYTEORDER\_LE and KRB5\_STORAGE\_BYTEORDER\_HOST.

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_storage\_set\_eof\_code (krb5\_storage \* sp, int code)**

Set the return code that will be used when end of storage is reached.

**Parameters:**

*sp* the storage  
*code* the error code to return on end of storage

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_storage\_set\_flags (krb5\_storage \* sp, krb5\_flags flags)**

Add the flags on a storage buffer by or-ing in the flags to the buffer.

**Parameters:**

*sp* the storage buffer to set the flags on  
*flags* the flags to set

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_storage\_set\_max\_alloc (krb5\_storage \* sp, size\_t size)**

Set the max alloc value

**Parameters:**

*sp* the storage buffer set the max allow for  
*size* maximum size to allocate, use 0 to remove limit

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_storage\_to\_data (krb5\_storage \* sp, krb5\_data \* data)**

Copy the content of storage

**Parameters:**

*sp* the storage to copy to a data  
*data* the copied data, free with **krb5\_data\_free()**

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL krb5\_storage\_truncate (krb5\_storage \* sp, off\_t offset)**

Truncate the storage buffer in sp to offset.

**Parameters:**

*sp* the storage buffer to truncate.

*offset* the offset to truncate too.

**Returns:**

An Kerberos 5 error code.

**KRB5\_LIB\_FUNCTION krb5\_ssize\_t KRB5\_LIB\_CALL krb5\_storage\_write (krb5\_storage \* sp, const void \* buf, size\_t len)**

Write to the storage buffer.

**Parameters:**

*sp* the storage buffer to write to

*buf* the buffer to write to the storage buffer

*len* the length to write

**Returns:**

The length of data written (can be shorter than len), or negative on error.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_address (krb5\_storage \* sp, krb5\_address p)**

Write a address block to storage.

**Parameters:**

*sp* the storage buffer to write to

*p* the address block to write.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_addrs (krb5\_storage \* sp, krb5\_addresses p)**

Write a addresses block to storage.

**Parameters:**

*sp* the storage buffer to write to  
*p* the addresses block to write.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_authdata (krb5\_storage \* sp, krb5\_authdata auth)**

Write a auth data block to storage.

**Parameters:**

*sp* the storage buffer to write to  
*auth* the auth data block to write.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_creds (krb5\_storage \* sp, krb5\_creds \* creds)**

Write a credentials block to storage.

**Parameters:**

*sp* the storage buffer to write to  
*creds* the creds block to write.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_creds\_tag (krb5\_storage \* sp, krb5\_creds \* creds)**

Write a tagged credentials block to storage.

**Parameters:**

*sp* the storage buffer to write to  
*creds* the creds block to write.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_data (krb5\_storage \* sp,**

**krb5\_data data)**

Store a data to the storage. The data is stored with an int32 as length plus the data (not padded).

**Parameters:**

*sp* the storage buffer to write to  
*data* the buffer to store.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_int16 (krb5\_storage \* sp, int16\_t value)**

Store a int16 to storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too  
*value* the value to store

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_int32 (krb5\_storage \* sp, int32\_t value)**

Store a int32 to storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too  
*value* the value to store

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_int8 (krb5\_storage \* sp, int8\_t value)**

Store a int8 to storage.

**Parameters:**

*sp* the storage to write too

*value* the value to store

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_keyblock (krb5\_storage \* sp, krb5\_keyblock p)**

Store a keyblock to the storage.

**Parameters:**

*sp* the storage buffer to write to

*p* the keyblock to write

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_principal (krb5\_storage \* sp, krb5\_const\_principal p)**

Write a principal block to storage.

**Parameters:**

*sp* the storage buffer to write to

*p* the principal block to write.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_string (krb5\_storage \* sp, const char \* s)**

Store a string to the buffer. The data is formatted as an len:uint32 plus the string itself (not padded).

**Parameters:**

*sp* the storage buffer to write to

*s* the string to store.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_stringz (krb5\_storage \* sp, const char \* s)**



Store a zero terminated string to the buffer. The data is stored one character at a time until a NUL is stored.

**Parameters:**

*sp* the storage buffer to write to  
*s* the string to store.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_times (krb5\_storage \* sp, krb5\_times times)**

Write a times block to storage.

**Parameters:**

*sp* the storage buffer to write to  
*times* the times block to write.

**Returns:**

0 on success, a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_uint16 (krb5\_storage \* sp, uint16\_t value)**

Store a uint16 to storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too  
*value* the value to store

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_uint32 (krb5\_storage \* sp, uint32\_t value)**

Store a uint32 to storage, byte order is controlled by the settings on the storage, see **krb5\_storage\_set\_byteorder()**.

**Parameters:**

*sp* the storage to write too

*value* the value to store

**Returns:**

0 for success, or a Kerberos 5 error code on failure.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_store\_uint8 (krb5\_storage \* sp, uint8\_t value)**

Store a uint8 to storage.

**Parameters:**

*sp* the storage to write too

*value* the value to store

**Returns:**

0 for success, or a Kerberos 5 error code on failure.