

**NAME**

Heimdal Kerberos 5 credential handing functions -

**Functions**

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_fwd\_tgt\_creds** (krb5\_context context, krb5\_auth\_context auth\_context, const char \*hostname, krb5\_principal client, krb5\_principal server, krb5\_ccache ccache, int forwardable, krb5\_data \*out\_data)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_forwarded\_creds** (krb5\_context context, krb5\_auth\_context auth\_context, krb5\_ccache ccache, krb5\_flags flags, const char \*hostname, krb5\_creds \*in\_creds, krb5\_data \*out\_data)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_init\_creds\_opt\_alloc** (krb5\_context context, krb5\_get\_init\_creds\_opt \*\*opt)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_get\_init\_creds\_opt\_free** (krb5\_context context, krb5\_get\_init\_creds\_opt \*opt)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_creds\_init** (krb5\_context context, krb5\_principal client, krb5\_prompter\_fct prompter, void \*prompter\_data, krb5\_deltat start\_time, krb5\_get\_init\_creds\_opt \*options, krb5\_init\_creds\_context \*rctx)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_creds\_set\_service** (krb5\_context context, krb5\_init\_creds\_context ctx, const char \*service)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_creds\_set\_password** (krb5\_context context, krb5\_init\_creds\_context ctx, const char \*password)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_creds\_set\_keytab** (krb5\_context context, krb5\_init\_creds\_context ctx, krb5\_keytab keytab)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_creds\_step** (krb5\_context context, krb5\_init\_creds\_context ctx, krb5\_data \*in, krb5\_data \*out, krb5\_krbhst\_info \*hostinfo, unsigned int \*flags)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_creds\_get\_error** (krb5\_context context, krb5\_init\_creds\_context ctx, KRB\_ERROR \*error)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_init\_creds\_free** (krb5\_context context, krb5\_init\_creds\_context ctx)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_init\_creds\_get** (krb5\_context context, krb5\_init\_creds\_context ctx)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_init\_creds\_password** (krb5\_context context, krb5\_creds \*creds, krb5\_principal client, const char \*password, krb5\_prompter\_fct prompter, void \*data, krb5\_deltat start\_time, const char \*in\_tkt\_service, krb5\_get\_init\_creds\_opt \*options)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_init\_creds\_keyblock** (krb5\_context context, krb5\_creds \*creds, krb5\_principal client, krb5\_keyblock \*keyblock, krb5\_deltat start\_time, const char \*in\_tkt\_service, krb5\_get\_init\_creds\_opt \*options)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_get\_init\_creds\_keytab** (krb5\_context context, krb5\_creds \*creds, krb5\_principal client, krb5\_keytab keytab, krb5\_deltat start\_time, const char

\*in\_tkt\_service, krb5\_get\_init\_creds\_opt \*options)

## Detailed Description

### Function Documentation

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_fwd\_tgt\_creds (krb5\_context context, krb5\_auth\_context auth\_context, const char \* hostname, krb5\_principal client, krb5\_principal server, krb5\_ccache ccache, int forwardable, krb5\_data \* out\_data)**

Forward credentials for client to host hostname , making them forwardable if forwardable, and returning the blob of data to sent in out\_data. If hostname == NULL, pick it from server.

#### Parameters:

*context* A kerberos 5 context.  
*auth\_context* the auth context with the key to encrypt the out\_data.  
*hostname* the host to forward the tickets too.  
*client* the client to delegate from.  
*server* the server to delegate the credential too.  
*ccache* credential cache to use.  
*forwardable* make the forwarded ticket forwardable.  
*out\_data* the resulting credential.

#### Returns:

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_get\_forwarded\_creds (krb5\_context context, krb5\_auth\_context auth\_context, krb5\_ccache ccache, krb5\_flags flags, const char \* hostname, krb5\_creds \* in\_creds, krb5\_data \* out\_data)**

Gets tickets forwarded to hostname. If the tickets that are forwarded are address-less, the forwarded tickets will also be address-less.

If the ticket have any address, hostname will be used for figure out the address to forward the ticket too. This since this might use DNS, its insecure and also doesn't represent configured all addresses of the host. For example, the host might have two addresses, one IPv4 and one IPv6 address where the later is not published in DNS. This IPv6 address might be used communications and thus the resulting ticket useless.

#### Parameters:

*context* A kerberos 5 context.  
*auth\_context* the auth context with the key to encrypt the out\_data.  
*ccache* credential cache to use  
*flags* the flags to control the resulting ticket flags

*hostname* the host to forward the tickets too.

*in\_creds* the in client and server ticket names. The client and server components forwarded to the remote host.

*out\_data* the resulting credential.

### Returns:

Return an error code or 0.

Some older of the MIT gssapi library used clear-text tickets (warped inside AP-REQ encryption), use the `krb5_auth_context` flag `KRB5_AUTH_CONTEXT_CLEAR_FORWARDED_CRED` to support those tickets. The session key is used otherwise to encrypt the forwarded ticket.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_get_init_creds_keyblock`  
(`krb5_context` context, `krb5_creds` \* creds, `krb5_principal` client, `krb5_keyblock` \* keyblock,  
`krb5_deltat` start\_time, `const char` \* in\_tkt\_service, `krb5_get_init_creds_opt` \* options)  
Get new credentials using keyblock.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_get_init_creds_keytab` (`krb5_context`  
context, `krb5_creds` \* creds, `krb5_principal` client, `krb5_keytab` keytab, `krb5_deltat` start\_time, `const`  
`char` \* in\_tkt\_service, `krb5_get_init_creds_opt` \* options)  
Get new credentials using keytab.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_get_init_creds_opt_alloc`  
(`krb5_context` context, `krb5_get_init_creds_opt` \*\* opt)  
Allocate a new `krb5_get_init_creds_opt` structure, free with `krb5_get_init_creds_opt_free()`.

**KRB5\_LIB\_FUNCTION** `void` **KRB5\_LIB\_CALL** `krb5_get_init_creds_opt_free` (`krb5_context` context,  
`krb5_get_init_creds_opt` \* opt)  
Free `krb5_get_init_creds_opt` structure.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_get_init_creds_password`  
(`krb5_context` context, `krb5_creds` \* creds, `krb5_principal` client, `const char` \* password,  
`krb5_prompter_fct` prompter, `void` \* data, `krb5_deltat` start\_time, `const char` \* in\_tkt\_service,  
`krb5_get_init_creds_opt` \* options)  
Get new credentials using password.

**KRB5\_LIB\_FUNCTION** `void` **KRB5\_LIB\_CALL** `krb5_init_creds_free` (`krb5_context` context,  
`krb5_init_creds_context` ctx)  
Free the `krb5_init_creds_context` allocated by `krb5_init_creds_init()`.

**Parameters:**

*context* A Kerberos 5 context.  
*ctx* The `krb5_init_creds_context` to free.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_init_creds_get` (`krb5_context context, krb5_init_creds_context ctx`)

Get new credentials as setup by the `krb5_init_creds_context`.

**Parameters:**

*context* A Kerberos 5 context.  
*ctx* The `krb5_init_creds_context` to process.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_init_creds_get_error` (`krb5_context context, krb5_init_creds_context ctx, KRB_ERROR * error`)

Get the last error from the transaction.

**Returns:**

Returns 0 or an error code

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_init_creds_init` (`krb5_context context, krb5_principal client, krb5_prompfter_fct prompfter, void * prompfter_data, krb5_deltat start_time, krb5_get_init_creds_opt * options, krb5_init_creds_context * rctx`)

Start a new context to get a new initial credential.

**Parameters:**

*context* A Kerberos 5 context.  
*client* The Kerberos principal to get the credential for, if NULL is given, the default principal is used as determined by `krb5_get_default_principal()`.  
*prompfter*  
*prompfter\_data*  
*start\_time* the time the ticket should start to be valid or 0 for now.  
*options* a options structure, can be NULL for default options.  
*rctx* A new allocated free with `krb5_init_creds_free()`.

**Returns:**

0 for success or an Kerberos 5 error code, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_init_creds_set_keytab` (`krb5_context context, krb5_init_creds_context ctx, krb5_keytab keytab`)

Set the keytab to use for authentication.

**Parameters:**

*context* a Kerberos 5 context.  
*ctx* `ctx krb5_init_creds_context` context.  
*keytab* the keytab to read the key from.

**Returns:**

0 for success, or an Kerberos 5 error code, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_init\_creds\_set\_password  
(krb5\_context context, krb5\_init\_creds\_context ctx, const char \* password)**

Sets the password that will use for the request.

**Parameters:**

*context* a Kerberos 5 context.  
*ctx* `ctx krb5_init_creds_context` context.  
*password* the password to use.

**Returns:**

0 for success, or an Kerberos 5 error code, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_init\_creds\_set\_service (krb5\_context  
context, krb5\_init\_creds\_context ctx, const char \* service)**

Sets the service that the is requested. This call is only neede for special initial tickets, by default the a `krbtgt` is fetched in the default realm.

**Parameters:**

*context* a Kerberos 5 context.  
*ctx* a `krb5_init_creds_context` context.  
*service* the service given as a string, for example `'kadmin/admin'`. If `NULL`, the default `krbtgt` in the clients realm is set.

**Returns:**

0 for success, or an Kerberos 5 error code, see `krb5_get_error_message()`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_init\_creds\_step (krb5\_context  
context, krb5\_init\_creds\_context ctx, krb5\_data \* in, krb5\_data \* out, krb5\_krbhst\_info \* hostinfo,  
unsigned int \* flags)**

The core loop if `krb5_get_init_creds()` function family. Create the packets and have the caller send them off to the KDC.

If the caller want all work been done for them, use **krb5\_init\_creds\_get()** instead.

**Parameters:**

*context* a Kerberos 5 context.

*ctx* ctx krb5\_init\_creds\_context context.

*in* input data from KDC, first round it should be reset by krb5\_data\_zer().

*out* reply to KDC.

*hostinfo* KDC address info, first round it can be NULL.

*flags* status of the round, if KRB5\_INIT\_CREDS\_STEP\_FLAG\_CONTINUE is set, continue one more round.

**Returns:**

0 for success, or an Kerberos 5 error code, see krb5\_get\_error\_message().