NAME

krb5_auth_con_addflags, **krb5_auth_con_free**, **krb5_auth_con_genaddrs**, **krb5_auth_con_generatelocalsubkey**, **krb5_auth_con_getaddrs**, **krb5_auth_con_getauthenticator**, **krb5_auth_con_getflags**, **krb5_auth_con_getkey**, **krb5_auth_con_getlocalsubkey**, **krb5_auth_con_getrcache**, **krb5_auth_con_getremotesubkey**, **krb5_auth_con_getuserkey**, **krb5_auth_con_init**, **krb5_auth_con_initivector**, **krb5_auth_con_removeflags**, **krb5_auth_con_setaddrs**, **krb5_auth_con_setaddrs_from_fd**, **krb5_auth_con_setflags**, **krb5_auth_con_setivector**, **krb5_auth_con_setkey**, **krb5_auth_con_setlocalsubkey**, **krb5_auth_con_setrcache**, **krb5_auth_con_setremotesubkey**, **krb5_auth_con_setuserkey**, **krb5_auth_context**, **krb5_auth_getcksumtype**, **krb5_auth_getkeytype**, **krb5_auth_getlocalseqnumber**, **krb5_auth_getremoteseqnumber**, **krb5_auth_setcksumtype**, **krb5_auth_setkeytype**, **krb5_auth_setlocalseqnumber**, **krb5_auth_setremoteseqnumber**, **krb5_free_authenticator** - manage authentication on connection level

LIBRARY

Kerberos 5 Library (libkrb5, -lkrb5)

SYNOPSIS

#include <krb5.h>

krb5_error_code
krb5_auth_con_init(*krb5_context context*, *krb5_auth_context *auth_context*);

void
krb5_auth_con_free(*krb5_context context*, *krb5_auth_context auth_context*);

krb5_error_code
krb5_auth_con_setflags(*krb5_context context*, *krb5_auth_context auth_context*, *int32_t flags*);

krb5_error_code
krb5_auth_con_getflags(*krb5_context context*, *krb5_auth_context auth_context*, *int32_t *flags*);

krb5_error_code
krb5_auth_con_addflags(*krb5_context context*, *krb5_auth_context auth_context*, *int32_t addflags*, *int32_t *flags*);

krb5_error_code
krb5_auth_con_removeflags(*krb5_context context*, *krb5_auth_context auth_context*, *int32_t removelags*, *int32_t *flags*);

krb5_error_code
krb5_auth_con_setaddrs(*krb5_context context, krb5_auth_context auth_context, krb5_address *local_addr, krb5_address *remote_addr*);

krb5_error_code
krb5_auth_con_getaddrs(*krb5_context context, krb5_auth_context auth_context, krb5_address **local_addr, krb5_address **remote_addr*);

krb5_error_code
krb5_auth_con_genaddrs(*krb5_context context, krb5_auth_context auth_context, int fd, int flags*);

krb5_error_code
krb5_auth_con_setaddrs_from_fd(*krb5_context context, krb5_auth_context auth_context, void *p_fd*);

krb5_error_code
krb5_auth_con_getkey(*krb5_context context, krb5_auth_context auth_context, krb5_keyblock **keyblock*);

krb5_error_code
krb5_auth_con_getlocalsubkey(*krb5_context context, krb5_auth_context auth_context, krb5_keyblock **keyblock*);

krb5_error_code
krb5_auth_con_getremotesubkey(*krb5_context context, krb5_auth_context auth_context, krb5_keyblock **keyblock*);

krb5_error_code
krb5_auth_con_generatelocalsubkey(*krb5_context context, krb5_auth_context auth_context, krb5_keyblock, *key"*);

krb5_error_code
krb5_auth_con_initivector(*krb5_context context, krb5_auth_context auth_context*);

krb5_error_code
krb5_auth_con_setivector(*krb5_context context, krb5_auth_context *auth_context, krb5_pointer ivector*);

void
krb5_free_authenticator(*krb5_context context, krb5_authenticator *authenticator*);

DESCRIPTION

The **krb5_auth_context** structure holds all context related to an authenticated connection, in a similar way to **krb5_context** that holds the context for the thread or process. **krb5_auth_context** is used by various functions that are directly related to authentication between the server/client. Example of data that this structure contains are various flags, addresses of client and server, port numbers, keyblocks (and subkeys), sequence numbers, replay cache, and checksum-type.

krb5_auth_con_init() allocates and initializes the **krb5_auth_context** structure. Default values can be changed with **krb5_auth_con_setcksumtype**() and **krb5_auth_con_setflags**(). The **auth_context** structure must be freed by **krb5_auth_con_free**().

krb5_auth_con_getflags(), **krb5_auth_con_setflags**(), **krb5_auth_con_addflags**() and **krb5_auth_con_removeflags**() gets and modifies the flags for a **krb5_auth_context** structure. Possible flags to set are:

KRB5_AUTH_CONTEXT_DO_SEQUENCE
: Generate and check sequence-number on each packet.

KRB5_AUTH_CONTEXT_DO_TIME
: Check timestamp on incoming packets.

KRB5_AUTH_CONTEXT_RET_SEQUENCE, KRB5_AUTH_CONTEXT_RET_TIME
: Return sequence numbers and time stamps in the outdata parameters.

KRB5_AUTH_CONTEXT_CLEAR_FORWARDED_CRED
: will force **krb5_get_forwarded_creds**() and **krb5_fwd_tgt_creds**() to create unencrypted) ENCTYPE_NULL) credentials. This is for use with old MIT server and JAVA based servers as they can't handle encrypted KRB-CRED. Note that sending such KRB-CRED is clear exposes crypto keys and tickets and is insecure, make sure the packet is encrypted in the protocol. krb5_rd_cred(3), krb5_rd_priv(3), krb5_rd_safe(3), krb5_mk_priv(3) and krb5_mk_safe(3). Setting this flag requires that parameter to be passed to these functions.

  The flags KRB5_AUTH_CONTEXT_DO_TIME also modifies the behavior the function **krb5_get_forwarded_creds**() by removing the timestamp in the forward credential message, this have backward compatibility problems since not all versions of the heimdal supports timeless credentional messages. Is very useful since it always the sender of the message to cache forward message and thus avoiding a round trip to the KDC for each time a credential is forwarded. The same functionality can be obtained by using address-less tickets.

krb5_auth_con_setaddrs(), **krb5_auth_con_setaddrs_from_fd**() and **krb5_auth_con_getaddrs**() gets and

sets the addresses that are checked when a packet is received. It is mandatory to set an address for the remote host. If the local address is not set, it iss deduced from the underlaying operating system. **krb5_auth_con_getaddrs**() will call **krb5_free_address**() on any address that is passed in *local_addr* or *remote_addr*. **krb5_auth_con_setaddr**() allows passing in a NULL pointer as *local_addr* and *remote_addr*, in that case it will just not set that address.

krb5_auth_con_setaddrs_from_fd() fetches the addresses from a file descriptor.

krb5_auth_con_genaddrs() fetches the address information from the given file descriptor *fd* depending on the bitmap argument *flags*.

Possible values on *flags* are:

KRB5_AUTH_CONTEXT_GENERATE_LOCAL_ADDR
fetches the local address from *fd*.

KRB5_AUTH_CONTEXT_GENERATE_REMOTE_ADDR
fetches the remote address from *fd*.

krb5_auth_con_setkey(), **krb5_auth_con_setuserkey**() and **krb5_auth_con_getkey**() gets and sets the key used for this auth context. The keyblock returned by **krb5_auth_con_getkey**() should be freed with **krb5_free_keyblock**(). The keyblock send into **krb5_auth_con_setkey**() is copied into the **krb5_auth_context**, and thus no special handling is needed. NULL is not a valid keyblock to **krb5_auth_con_setkey**().

krb5_auth_con_setuserkey() is only useful when doing user to user authentication. **krb5_auth_con_setkey**() is equivalent to **krb5_auth_con_setuserkey**().

krb5_auth_con_getlocalsubkey(), **krb5_auth_con_setlocalsubkey**(), **krb5_auth_con_getremotesubkey**() and **krb5_auth_con_setremotesubkey**() gets and sets the keyblock for the local and remote subkey. The keyblock returned by **krb5_auth_con_getlocalsubkey**() and **krb5_auth_con_getremotesubkey**() must be freed with **krb5_free_keyblock**().

krb5_auth_setcksumtype() and **krb5_auth_getcksumtype**() sets and gets the checksum type that should be used for this connection.

krb5_auth_con_generatelocalsubkey() generates a local subkey that have the same encryption type as *key*.

krb5_auth_getremoteseqnumber() **krb5_auth_setremoteseqnumber**(), **krb5_auth_getlocalseqnumber**()

and **krb5_auth_setlocalseqnumber**() gets and sets the sequence-number for the local and remote sequence-number counter.

krb5_auth_setkeytype() and **krb5_auth_getkeytype**() gets and gets the keytype of the keyblock in **krb5_auth_context**.

krb5_auth_con_getauthenticator() Retrieves the authenticator that was used during mutual authentication. The authenticator returned should be freed by calling **krb5_free_authenticator**().

krb5_auth_con_getrcache() and **krb5_auth_con_setrcache**() gets and sets the replay-cache.

krb5_auth_con_initivector() allocates memory for and zeros the initial vector in the *auth_context* keyblock.

krb5_auth_con_setivector() sets the i_vector portion of *auth_context* to *ivector*.

krb5_free_authenticator() free the content of *authenticator* and *authenticator* itself.

SEE ALSO

krb5_context(3), kerberos(8)