NAME

krb5_digest, **krb5_digest_alloc**, **krb5_digest_free**, **krb5_digest_set_server_cb**, **krb5_digest_set_type**, **krb5_digest_set_hostname**, **krb5_digest_get_server_nonce**, **krb5_digest_set_server_nonce**, **krb5_digest_get_opaque**, **krb5_digest_set_opaque**, **krb5_digest_get_identifier**, **krb5_digest_set_identifier**, **krb5_digest_init_request**, **krb5_digest_set_client_nonce**, **krb5_digest_set_digest**, **krb5_digest_set_username**, **krb5_digest_set_authid**, **krb5_digest_set_authentication_user**, **krb5_digest_set_realm**, **krb5_digest_set_method**, **krb5_digest_set_uri**, **krb5_digest_set_nonceCount**, **krb5_digest_set_qop**, **krb5_digest_request**, **krb5_digest_get_responseData**, **krb5_digest_get_rsp**, **krb5_digest_get_tickets**, **krb5_digest_get_client_binding**, **krb5_digest_get_a1_hash** **-** remote digest (HTTP-DIGEST, SASL, CHAP) suppport

LIBRARY

Kerberos 5 Library (libkrb5, -lkrb5)

SYNOPSIS

#include <krb5.h>

typedef struct krb5_digest *krb5_digest;

krb5_error_code
krb5_digest_alloc(*krb5_context context*, *krb5_digest *digest*);

void
krb5_digest_free(*krb5_digest digest*);

krb5_error_code
krb5_digest_set_type(*krb5_context context*, *krb5_digest digest*, *const char *type*);

krb5_error_code
krb5_digest_set_server_cb(*krb5_context context*, *krb5_digest digest*, *const char *type*,
  *const char *binding*);

krb5_error_code
krb5_digest_set_hostname(*krb5_context context*, *krb5_digest digest*, *const char *hostname*);

*const char **
krb5_digest_get_server_nonce(*krb5_context context*, *krb5_digest digest*);

krb5_error_code

krb5_digest_set_server_nonce(*krb5_context context*, *krb5_digest digest*, *const char *nonce*);

*const char **
krb5_digest_get_opaque(*krb5_context context*, *krb5_digest digest*);

krb5_error_code
krb5_digest_set_opaque(*krb5_context context*, *krb5_digest digest*, *const char *opaque*);

*const char **
krb5_digest_get_identifier(*krb5_context context*, *krb5_digest digest*);

krb5_error_code
krb5_digest_set_identifier(*krb5_context context*, *krb5_digest digest*, *const char *id*);

krb5_error_code
krb5_digest_init_request(*krb5_context context*, *krb5_digest digest*, *krb5_realm realm*,
    krb5_ccache ccache);

krb5_error_code
krb5_digest_set_client_nonce(*krb5_context context*, *krb5_digest digest*, *const char *nonce*);

krb5_error_code
krb5_digest_set_digest(*krb5_context context*, *krb5_digest digest*, *const char *dgst*);

krb5_error_code
krb5_digest_set_username(*krb5_context context*, *krb5_digest digest*, *const char *username*);

krb5_error_code
krb5_digest_set_authid(*krb5_context context*, *krb5_digest digest*, *const char *authid*);

krb5_error_code
krb5_digest_set_authentication_user(*krb5_context context*, *krb5_digest digest*,
    krb5_principal authentication_user);

krb5_error_code
krb5_digest_set_realm(*krb5_context context*, *krb5_digest digest*, *const char *realm*);

krb5_error_code
krb5_digest_set_method(*krb5_context context*, *krb5_digest digest*, *const char *method*);

krb5_error_code
krb5_digest_set_uri(*krb5_context context*, *krb5_digest digest*, *const char *uri*);

krb5_error_code
krb5_digest_set_nonceCount(*krb5_context context*, *krb5_digest digest*, *const char *nonce_count*);

krb5_error_code
krb5_digest_set_qop(*krb5_context context*, *krb5_digest digest*, *const char *qop*);

krb5_error_code
krb5_digest_request(*krb5_context context*, *krb5_digest digest*, *krb5_realm realm*, *krb5_ccache ccache*);

*const char **
krb5_digest_get_responseData(*krb5_context context*, *krb5_digest digest*);

*const char **
krb5_digest_get_rsp(*krb5_context context*, *krb5_digest digest*);

krb5_error_code
krb5_digest_get_tickets(*krb5_context context*, *krb5_digest digest*, *Ticket **tickets*);

krb5_error_code
krb5_digest_get_client_binding(*krb5_context context*, *krb5_digest digest*, *char **type*, *char **binding*);

krb5_error_code
krb5_digest_get_a1_hash(*krb5_context context*, *krb5_digest digest*, *krb5_data *data*);

DESCRIPTION

The **krb5_digest_alloc**() function allocatates the *digest* structure. The structure should be freed with **krb5_digest_free**() when it is no longer being used.

krb5_digest_alloc() returns 0 to indicate success. Otherwise an kerberos code is returned and the pointer that *digest* points to is set to NULL.

krb5_digest_free() free the structure *digest*.

SEE ALSO

krb5(3), kerberos(8)