NAME

HDB -

SYNOPSIS

#include <hdb.h>

Data Fields

char * **hdb_name**
krb5_error_code(* **hdb_open**)(krb5_context, struct **HDB** *, int, mode_t)
krb5_error_code(* **hdb_close**)(krb5_context, struct **HDB** *)
void(* **hdb_free**)(krb5_context, struct **HDB** *, **hdb_entry_ex** *)
krb5_error_code(* **hdb_fetch_kvno**)(krb5_context, struct **HDB** *, krb5_const_principal, unsigned, krb5_kvno, **hdb_entry_ex** *)
krb5_error_code(* **hdb_store**)(krb5_context, struct **HDB** *, unsigned, **hdb_entry_ex** *)
krb5_error_code(* **hdb_remove**)(krb5_context, struct **HDB** *, krb5_const_principal)
krb5_error_code(* **hdb_firstkey**)(krb5_context, struct **HDB** *, unsigned, **hdb_entry_ex** *)
krb5_error_code(* **hdb_nextkey**)(krb5_context, struct **HDB** *, unsigned, **hdb_entry_ex** *)
krb5_error_code(* **hdb_lock**)(krb5_context, struct **HDB** *, int)
krb5_error_code(* **hdb_unlock**)(krb5_context, struct **HDB** *)
krb5_error_code(* **hdb_rename**)(krb5_context, struct **HDB** *, const char *)
krb5_error_code(* **hdb__get**)(krb5_context, struct **HDB** *, krb5_data, krb5_data *)
krb5_error_code(* **hdb__put**)(krb5_context, struct **HDB** *, int, krb5_data, krb5_data)
krb5_error_code(* **hdb__del**)(krb5_context, struct **HDB** *, krb5_data)
krb5_error_code(* **hdb_destroy**)(krb5_context, struct **HDB** *)
krb5_error_code(* **hdb_get_realms**)(krb5_context, struct **HDB** *, krb5_realm **)
krb5_error_code(* **hdb_password**)(krb5_context, struct **HDB** *, **hdb_entry_ex** *, const char *, int)
krb5_error_code(* **hdb_auth_status**)(krb5_context, struct **HDB** *, **hdb_entry_ex** *, int)
krb5_error_code(* **hdb_check_constrained_delegation**)(krb5_context, struct **HDB** *, **hdb_entry_ex** *, krb5_const_principal)
krb5_error_code(* **hdb_check_pkinit_ms_upn_match**)(krb5_context, struct **HDB** *, **hdb_entry_ex** *, krb5_const_principal)
krb5_error_code(* **hdb_check_s4u2self**)(krb5_context, struct **HDB** *, **hdb_entry_ex** *, krb5_const_principal)

Detailed Description

HDB backend function pointer structure

The **HDB** structure is what the KDC and kadmind framework uses to query the backend database when talking about principals.

Field Documentation

char* HDB::hdb_name

don’t use, only for DB3

krb5_error_code(* HDB::hdb_open)(krb5_context, struct HDB *, int, mode_t)

Open (or create) the a Kerberos database.

Open (or create) the a Kerberos database that was resolved with hdb_create(). The third and fourth flag to the function are the same as open(), thus passing O_CREAT will create the data base if it doesn’t exists.

Then done the caller should call **hdb_close()**, and to release all resources **hdb_destroy()**.

krb5_error_code(* HDB::hdb_close)(krb5_context, struct HDB *)

Close the database for transaction

Closes the database for further transactions, wont release any permanant resources. the database can be ->hdb_open-ed again.

void(* HDB::hdb_free)(krb5_context, struct HDB *, hdb_entry_ex *)

Free an entry after use.

krb5_error_code(* HDB::hdb_fetch_kvno)(krb5_context, struct HDB *, krb5_const_principal, unsigned, krb5_kvno, hdb_entry_ex *)

Fetch an entry from the backend

Fetch an entry from the backend, flags are what type of entry should be fetch: client, server, krbtgt. knvo (if specified and flags HDB_F_KVNO_SPECIFIED set) is the kvno to get

krb5_error_code(* HDB::hdb_store)(krb5_context, struct HDB *, unsigned, hdb_entry_ex *)

Store an entry to database

krb5_error_code(* HDB::hdb_remove)(krb5_context, struct HDB *, krb5_const_principal)

Remove an entry from the database.

krb5_error_code(* HDB::hdb_firstkey)(krb5_context, struct HDB *, unsigned, hdb_entry_ex *)

As part of iteration, fetch one entry

krb5_error_code(* HDB::hdb_nextkey)(krb5_context, struct HDB *, unsigned, hdb_entry_ex *)

As part of iteration, fetch next entry

krb5_error_code(* HDB::hdb_lock)(krb5_context, struct HDB *, int)
Lock database

A lock can only be held by one consumers. Transaction can still happen on the database while the lock is held, so the entry is only useful for syncroning creation of the database and renaming of the database.

krb5_error_code(* HDB::hdb_unlock)(krb5_context, struct HDB *)
Unlock database

krb5_error_code(* HDB::hdb_rename)(krb5_context, struct HDB *, const char *)
Rename the data base.

Assume that the database is not hdb_open'ed and not locked.

krb5_error_code(* HDB::hdb__get)(krb5_context, struct HDB *, krb5_data, krb5_data *)
Get an hdb_entry from a classical DB backend

If the database is a classical DB (ie BDB, NDBM, GDBM, etc) backend, this function will take a principal key (krb5_data) and return all data related to principal in the return krb5_data. The returned encoded entry is of type hdb_entry or hdb_entry_alias.

krb5_error_code(* HDB::hdb__put)(krb5_context, struct HDB *, int, krb5_data, krb5_data)
Store an hdb_entry from a classical DB backend

Same discussion as in **HDB::hdb__get**

krb5_error_code(* HDB::hdb__del)(krb5_context, struct HDB *, krb5_data)
Delete and hdb_entry from a classical DB backend

Same discussion as in **HDB::hdb__get**

krb5_error_code(* HDB::hdb_destroy)(krb5_context, struct HDB *)
Destroy the handle to the database.

Destroy the handle to the database, deallocate all memory and related resources. Does not remove any permanent data. Its the logical reverse of hdb_create() function that is the entry point for the module.

krb5_error_code(* HDB::hdb_get_realms)(krb5_context, struct HDB *, krb5_realm **)
Get the list of realms this backend handles. This call is optional to support. The returned realms are used for announcing the realms over bonjour. Free returned array with krb5_free_host_realm().

krb5_error_code(* HDB::hdb_password)(krb5_context, struct HDB *, hdb_entry_ex *, const char *, int)
Change password.

Will update keys for the entry when given password. The new keys must be written into the entry and will then later be ->**hdb_store()** into the database. The backend will still perform all other operations, increasing the kvno, and update modification timestamp.

The backend needs to call _kadm5_set_keys() and perform password quality checks.

krb5_error_code(* HDB::hdb_auth_status)(krb5_context, struct HDB *, hdb_entry_ex *, int)
Auth feedback

This is a feedback call that allows backends that provides lockout functionality to register failure and/or successes.

In case the entry is locked out, the backend should set the hdb_entry.flags.locked-out flag.

krb5_error_code(* HDB::hdb_check_constrained_delegation)(krb5_context, struct HDB *, hdb_entry_ex *, krb5_const_principal)
Check if delegation is allowed.

krb5_error_code(* HDB::hdb_check_pkinit_ms_upn_match)(krb5_context, struct HDB *, hdb_entry_ex *, krb5_const_principal)
Check if this name is an alias for the supplied client for PKINIT userPrinicpalName logins

krb5_error_code(* HDB::hdb_check_s4u2self)(krb5_context, struct HDB *, hdb_entry_ex *, krb5_const_principal)
Check if s4u2self is allowed from this client to this server

Author

Generated automatically by Doxygen for Heimdalhdblibrary from the source code.