

**NAME**

HDB -

**SYNOPSIS**

#include &lt;hdb.h&gt;

**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. kvno (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 synchroning 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 userPrincipalName 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.