

**NAME**

**krb5\_krbhst\_init**, **krb5\_krbhst\_init\_flags**, **krb5\_krbhst\_next**, **krb5\_krbhst\_next\_as\_string**,  
**krb5\_krbhst\_reset**, **krb5\_krbhst\_free**, **krb5\_krbhst\_format\_string**, **krb5\_krbhst\_get\_addrinfo** - lookup  
Kerberos KDC hosts

**LIBRARY**

Kerberos 5 Library (libkrb5, -lkrb5)

**SYNOPSIS**

```
#include <krb5.h>
```

*krb5\_error\_code*

```
krb5_krbhst_init(krb5_context context, const char *realm, unsigned int type,  
                krb5_krbhst_handle *handle);
```

*krb5\_error\_code*

```
krb5_krbhst_init_flags(krb5_context context, const char *realm, unsigned int type, int flags,  
                      krb5_krbhst_handle *handle);
```

*krb5\_error\_code*

```
krb5_krbhst_next(krb5_context context, krb5_krbhst_handle handle, krb5_krbhst_info **host);
```

*krb5\_error\_code*

```
krb5_krbhst_next_as_string(krb5_context context, krb5_krbhst_handle handle, char *hostname,  
                          size_t hostlen);
```

*void*

```
krb5_krbhst_reset(krb5_context context, krb5_krbhst_handle handle);
```

*void*

```
krb5_krbhst_free(krb5_context context, krb5_krbhst_handle handle);
```

*krb5\_error\_code*

```
krb5_krbhst_format_string(krb5_context context, const krb5_krbhst_info *host, char *hostname,  
                          size_t hostlen);
```

*krb5\_error\_code*

```
krb5_krbhst_get_addrinfo(krb5_context context, krb5_krbhst_info *host, struct addrinfo **ai);
```

**DESCRIPTION**

These functions are used to sequence through all Kerberos hosts of a particular realm and service. The service type can be the KDCs, the administrative servers, the password changing servers, or the servers for Kerberos 4 ticket conversion.

First a handle to a particular service is obtained by calling **krb5\_krbhst\_init()** (or **krb5\_krbhst\_init\_flags()**) with the *realm* of interest and the type of service to lookup. The *type* can be one of:

```
KRB5_KRBHST_KDC
KRB5_KRBHST_ADMIN
KRB5_KRBHST_CHANGEPW
KRB5_KRBHST_KRB524
```

The *handle* is returned to the caller, and should be passed to the other functions.

The *flag* argument to **krb5\_krbhst\_init\_flags** is the same flags as **krb5\_send\_to\_kdc\_flags()** uses. Possible values are:

```
KRB5_KRBHST_FLAGS_MASTER    only talk to master (readwrite) KDC
KRB5_KRBHST_FLAGS_LARGE_MSG this is a large message, so use transport that can
                             handle that.
```

For each call to **krb5\_krbhst\_next()** information on a new host is returned. The former function returns in *host* a pointer to a structure containing information about the host, such as protocol, hostname, and port:

```
typedef struct krb5_krbhst_info {
    enum { KRB5_KRBHST_UDP,
          KRB5_KRBHST_TCP,
          KRB5_KRBHST_HTTP } proto;
    unsigned short port;
    struct addrinfo *ai;
    struct krb5_krbhst_info *next;
    char hostname[1];
} krb5_krbhst_info;
```

The related function, **krb5\_krbhst\_next\_as\_string()**, return the same information as a URL-like string.

When there are no more hosts, these functions return KRB5\_KDC\_UNREACH.

To re-iterate over all hosts, call **krb5\_krbhst\_reset()** and the next call to **krb5\_krbhst\_next()** will return the first host.

When done with the handle, **krb5\_krbhst\_free()** should be called.

To use a *krb5\_krbhst\_info*, there are two functions: **krb5\_krbhst\_format\_string()** that will return a printable representation of that struct and **krb5\_krbhst\_get\_addrinfo()** that will return a *struct addrinfo* that can then be used for communicating with the server mentioned.

## EXAMPLES

The following code will print the KDCs of the realm "MY.REALM":

```
krb5_krbhst_handle handle;
char host[MAXHOSTNAMELEN];
krb5_krbhst_init(context, "MY.REALM", KRB5_KRBHST_KDC, &handle);
while(krb5_krbhst_next_as_string(context, handle,
                                host, sizeof(host)) == 0)
    printf("%s\n", host);
krb5_krbhst_free(context, handle);
```

## SEE ALSO

getaddrinfo(3), krb5\_get\_krbhst(3), krb5\_send\_to\_kdc\_flags(3)

## HISTORY

These functions first appeared in Heimdal 0.3g.