

**NAME**

Heimdal Kerberos 5 support functions -

**Functions**

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_acl\_match\_string** (krb5\_context context, const char \*string, const char \*format,...)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_acl\_match\_file** (krb5\_context context, const char \*file, const char \*format,...)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_config\_parse\_file\_multi** (krb5\_context context, const char \*fname, krb5\_config\_section \*\*res)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_config\_file\_free** (krb5\_context context, krb5\_config\_section \*s)

KRB5\_LIB\_FUNCTION const krb5\_config\_binding \*KRB5\_LIB\_CALL **krb5\_config\_get\_list** (krb5\_context context, const krb5\_config\_section \*c,...)

KRB5\_LIB\_FUNCTION const krb5\_config\_binding \*KRB5\_LIB\_CALL **krb5\_config\_vget\_list** (krb5\_context context, const krb5\_config\_section \*c, va\_list args)

KRB5\_LIB\_FUNCTION const char \*KRB5\_LIB\_CALL **krb5\_config\_get\_string** (krb5\_context context, const krb5\_config\_section \*c,...)

KRB5\_LIB\_FUNCTION const char \*KRB5\_LIB\_CALL **krb5\_config\_vget\_string** (krb5\_context context, const krb5\_config\_section \*c, va\_list args)

KRB5\_LIB\_FUNCTION const char \*KRB5\_LIB\_CALL **krb5\_config\_vget\_string\_default** (krb5\_context context, const krb5\_config\_section \*c, const char \*def\_value, va\_list args)

KRB5\_LIB\_FUNCTION const char \*KRB5\_LIB\_CALL **krb5\_config\_get\_string\_default** (krb5\_context context, const krb5\_config\_section \*c, const char \*def\_value,...)

KRB5\_LIB\_FUNCTION char \*\*KRB5\_LIB\_CALL **krb5\_config\_vget\_strings** (krb5\_context context, const krb5\_config\_section \*c, va\_list args)

KRB5\_LIB\_FUNCTION char \*\*KRB5\_LIB\_CALL **krb5\_config\_get\_strings** (krb5\_context context, const krb5\_config\_section \*c,...)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_config\_free\_strings** (char \*\*strings)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_config\_vget\_bool\_default** (krb5\_context context, const krb5\_config\_section \*c, krb5\_boolean def\_value, va\_list args)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_config\_vget\_bool** (krb5\_context context, const krb5\_config\_section \*c, va\_list args)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_config\_get\_bool\_default** (krb5\_context context, const krb5\_config\_section \*c, krb5\_boolean def\_value,...)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_config\_get\_bool** (krb5\_context context, const krb5\_config\_section \*c,...)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_config\_vget\_time\_default** (krb5\_context context, const krb5\_config\_section \*c, int def\_value, va\_list args)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_config\_vget\_time** (krb5\_context context, const

krb5\_config\_section \*c, va\_list args)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_config\_get\_time\_default** (krb5\_context context, const krb5\_config\_section \*c, int def\_value,...)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_config\_get\_time** (krb5\_context context, const krb5\_config\_section \*c,...)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_expand\_hostname** (krb5\_context context, const char \*orig\_hostname, char \*\*new\_hostname)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_expand\_hostname\_realms** (krb5\_context context, const char \*orig\_hostname, char \*\*new\_hostname, char \*\*\*realms)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_free\_host\_realm** (krb5\_context context, krb5\_realm \*realmlist)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_kuserok** (krb5\_context context, krb5\_principal principal, const char \*luser)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_plugin\_register** (krb5\_context context, enum krb5\_plugin\_type type, const char \*name, void \*symbol)

## Detailed Description

### Function Documentation

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_acl\_match\_file** (krb5\_context context, const char \* file, const char \* format, ...)

krb5\_acl\_match\_file matches ACL format against each line in a file using **krb5\_acl\_match\_string()**. Lines starting with # are treated like comments and ignored.

#### Parameters:

*context* Kerberos 5 context.

*file* file with acl listed in the file.

*format* format to match.

... parameter to format string.

#### Returns:

Return an error code or 0.

#### See also:

**krb5\_acl\_match\_string**

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_acl\_match\_string** (krb5\_context context, const char \* string, const char \* format, ...)

krb5\_acl\_match\_string matches ACL format against a string.

The ACL format has three format specifiers: s, f, and r. Each specifier will retrieve one argument from

the variable arguments for either matching or storing data. The input string is split up using ' ' (space) and '\t' (tab) as a delimiter; multiple and '\t' in a row are considered to be the same.

List of format specifiers:

- ⊕ *s* Matches a string using `strcmp(3)` (case sensitive).
- ⊕ *f* Matches the string with `fnmatch(3)`. The `flags` argument (the last argument) passed to the `fnmatch` function is 0.
- ⊕ *r* Returns a copy of the string in the `char **` passed in; the copy must be freed with `free(3)`. There is no need to `free(3)` the string on error: the function will clean up and set the pointer to `NULL`.

**Parameters:**

*context* Kerberos 5 context  
*string* string to match with  
*format* format to match  
 ... parameter to format string

**Returns:**

Return an error code or 0.

```
char *s;
```

```
ret = krb5_acl_match_string(context, 'foo', 's', 'foo');
if (ret)
    krb5_errx(context, 1, 'acl didn't match');
ret = krb5_acl_match_string(context, 'foo foo baz/kaka',
    'ss', 'foo', &s, 'foo/*');
if (ret) {
    // no need to free(s) on error
    assert(s == NULL);
    krb5_errx(context, 1, 'acl didn't match');
}
free(s);
```

**See also:**

**`krb5_acl_match_file`**

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_config_file_free` (`krb5_context`

**context, krb5\_config\_section \* s)**

Free configuration file section, the result of `krb5_config_parse_file()` and `krb5_config_parse_file_multi()`.

**Parameters:**

*context* A Kerberos 5 context  
*s* the configuration section to free

**Returns:**

returns 0 on successes, otherwise an error code, see `krb5_get_error_message()`

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_config\_free\_strings (char \*\* strings)**

Free the resulting strings from `krb5_config_get_strings()` and `krb5_config_vget_strings()`.

**Parameters:**

*strings* strings to free

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_config\_get\_bool (krb5\_context context, const krb5\_config\_section \* c, ...)**

Like `krb5_config_get_bool()` but with a `va_list` list of configuration selection.

Configuration value to a boolean value, where yes/true and any non-zero number means TRUE and other value is FALSE.

**Parameters:**

*context* A Kerberos 5 context.  
*c* a configuration section, or NULL to use the section from context  
 ... a list of names, terminated with NULL.

**Returns:**

TRUE or FALSE

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_config\_get\_bool\_default (krb5\_context context, const krb5\_config\_section \* c, krb5\_boolean def\_value, ...)**

`krb5_config_get_bool_default()` will convert the configuration option value to a boolean value, where yes/true and any non-zero number means TRUE and other value is FALSE.

**Parameters:**

*context* A Kerberos 5 context.  
*c* a configuration section, or NULL to use the section from context

*def\_value* the default value to return if no configuration found in the database.  
 ... a list of names, terminated with NULL.

**Returns:**

TRUE or FALSE

**KRB5\_LIB\_FUNCTION** **const krb5\_config\_binding\*** **KRB5\_LIB\_CALL** **krb5\_config\_get\_list**  
**(krb5\_context context, const krb5\_config\_section \* c, ...)**

Get a list of configuration binding list for more processing

**Parameters:**

*context* A Kerberos 5 context.  
*c* a configuration section, or NULL to use the section from context  
 ... a list of names, terminated with NULL.

**Returns:**

NULL if configuration list is not found, a list otherwise

**KRB5\_LIB\_FUNCTION** **const char\*** **KRB5\_LIB\_CALL** **krb5\_config\_get\_string** **(krb5\_context context,**  
**const krb5\_config\_section \* c, ...)**

Returns a 'const char \*' to a string in the configuration database. The string may not be valid after a reload of the configuration database so a caller should make a local copy if it needs to keep the string.

**Parameters:**

*context* A Kerberos 5 context.  
*c* a configuration section, or NULL to use the section from context  
 ... a list of names, terminated with NULL.

**Returns:**

NULL if configuration string not found, a string otherwise

**KRB5\_LIB\_FUNCTION** **const char\*** **KRB5\_LIB\_CALL** **krb5\_config\_get\_string\_default** **(krb5\_context**  
**context, const krb5\_config\_section \* c, const char \* def\_value, ...)**

Like **krb5\_config\_get\_string()**, but instead of returning NULL, instead return a default value.

**Parameters:**

*context* A Kerberos 5 context.  
*c* a configuration section, or NULL to use the section from context  
*def\_value* the default value to return if no configuration found in the database.  
 ... a list of names, terminated with NULL.

**Returns:**

a configuration string

**KRB5\_LIB\_FUNCTION char\*\* KRB5\_LIB\_CALL krb5\_config\_get\_strings (krb5\_context context, const krb5\_config\_section \* c, ...)**

Get a list of configuration strings, free the result with **krb5\_config\_free\_strings()**.

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

... a list of names, terminated with NULL.

**Returns:**

TRUE or FALSE

**KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL krb5\_config\_get\_time (krb5\_context context, const krb5\_config\_section \* c, ...)**

Get the time from the configuration file using a relative time, for example: 1h30s

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

... a list of names, terminated with NULL.

**Returns:**

parsed the time or -1 on error

**KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL krb5\_config\_get\_time\_default (krb5\_context context, const krb5\_config\_section \* c, int def\_value, ...)**

Get the time from the configuration file using a relative time, for example: 1h30s

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*def\_value* the default value to return if no configuration found in the database.

... a list of names, terminated with NULL.

**Returns:**

parsed the time (or *def\_value* on parse error)

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_config\_parse\_file\_multi**  
**(krb5\_context context, const char \* fname, krb5\_config\_section \*\* res)**

Parse a configuration file and add the result into res. This interface can be used to parse several configuration files into one resulting krb5\_config\_section by calling it repeatably.

**Parameters:**

*context* a Kerberos 5 context.

*fname* a file name to a Kerberos configuration file

*res* the returned result, must be free with **krb5\_free\_config\_files()**.

**Returns:**

Return an error code or 0, see **krb5\_get\_error\_message()**.

If the *fname* starts with '~/' parse configuration file in the current users home directory. The behavior can be disabled and enabled by calling **krb5\_set\_home\_dir\_access()**.

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_config\_vget\_bool (krb5\_context context, const krb5\_config\_section \* c, va\_list args)**

**krb5\_config\_get\_bool()** will convert the configuration option value to a boolean value, where yes/true and any non-zero number means TRUE and other value is FALSE.

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*args* a va\_list of arguments

**Returns:**

TRUE or FALSE

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_config\_vget\_bool\_default**  
**(krb5\_context context, const krb5\_config\_section \* c, krb5\_boolean def\_value, va\_list args)**

Like **krb5\_config\_get\_bool\_default()** but with a va\_list list of configuration selection.

Configuration value to a boolean value, where yes/true and any non-zero number means TRUE and other value is FALSE.

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*def\_value* the default value to return if no configuration found in the database.

*args* a va\_list of arguments

**Returns:**

TRUE or FALSE

**KRB5\_LIB\_FUNCTION const krb5\_config\_binding\* KRB5\_LIB\_CALL krb5\_config\_vget\_list (krb5\_context context, const krb5\_config\_section \* c, va\_list args)**

Get a list of configuration binding list for more processing

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*args* a va\_list of arguments

**Returns:**

NULL if configuration list is not found, a list otherwise

**KRB5\_LIB\_FUNCTION const char\* KRB5\_LIB\_CALL krb5\_config\_vget\_string (krb5\_context context, const krb5\_config\_section \* c, va\_list args)**

Like `krb5_config_get_string()`, but uses a va\_list instead of ...

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*args* a va\_list of arguments

**Returns:**

NULL if configuration string not found, a string otherwise

**KRB5\_LIB\_FUNCTION const char\* KRB5\_LIB\_CALL krb5\_config\_vget\_string\_default (krb5\_context context, const krb5\_config\_section \* c, const char \* def\_value, va\_list args)**

Like `krb5_config_vget_string()`, but instead of returning NULL, instead return a default value.

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*def\_value* the default value to return if no configuration found in the database.

*args* a va\_list of arguments

**Returns:**



a configuration string

**KRB5\_LIB\_FUNCTION char\*\* KRB5\_LIB\_CALL krb5\_config\_vget\_strings (krb5\_context context, const krb5\_config\_section \* c, va\_list args)**

Get a list of configuration strings, free the result with **krb5\_config\_free\_strings()**.

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*args* a va\_list of arguments

**Returns:**

TRUE or FALSE

**KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL krb5\_config\_vget\_time (krb5\_context context, const krb5\_config\_section \* c, va\_list args)**

Get the time from the configuration file using a relative time, for example: 1h30s

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*args* a va\_list of arguments

**Returns:**

parsed the time or -1 on error

**KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL krb5\_config\_vget\_time\_default (krb5\_context context, const krb5\_config\_section \* c, int def\_value, va\_list args)**

Get the time from the configuration file using a relative time.

Like **krb5\_config\_get\_time\_default()** but with a va\_list list of configuration selection.

**Parameters:**

*context* A Kerberos 5 context.

*c* a configuration section, or NULL to use the section from context

*def\_value* the default value to return if no configuration found in the database.

*args* a va\_list of arguments

**Returns:**

parsed the time (or def\_value on parse error)

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_expand\_hostname (krb5\_context context, const char \* orig\_hostname, char \*\* new\_hostname)**

**krb5\_expand\_hostname()** tries to make orig\_hostname into a more canonical one in the newly allocated space returned in new\_hostname.

**Parameters:**

*context* a Keberos context

*orig\_hostname* hostname to canonicalise.

*new\_hostname* output hostname, caller must free hostname with krb5\_xfree().

**Returns:**

Return an error code or 0, see krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_expand\_hostname\_realms (krb5\_context context, const char \* orig\_hostname, char \*\* new\_hostname, char \*\*\* realms)**

**krb5\_expand\_hostname\_realms()** expands orig\_hostname to a name we believe to be a hostname in newly allocated space in new\_hostname and return the realms new\_hostname is believed to belong to in realms.

**Parameters:**

*context* a Keberos context

*orig\_hostname* hostname to canonicalise.

*new\_hostname* output hostname, caller must free hostname with krb5\_xfree().

*realms* output possible realms, is an array that is terminated with NULL. Caller must free with **krb5\_free\_host\_realm()**.

**Returns:**

Return an error code or 0, see krb5\_get\_error\_message().

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_free\_host\_realm (krb5\_context context, krb5\_realm \* realmlist)**

Free all memory allocated by 'realmlist'

**Parameters:**

*context* A Kerberos 5 context.

*realmlist* realmlist to free, NULL is ok

**Returns:**

a Kerberos error code, always 0.

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_kuserok (krb5\_context context, krb5\_principal principal, const char \* luser)**

This function takes the name of a local user and checks if principal is allowed to log in as that user.

The user may have a `~/.k5login` file listing principals that are allowed to login as that user. If that file does not exist, all principals with a first component identical to the username, and a realm considered local, are allowed access.

The `.k5login` file must contain one principal per line, be owned by user and not be writable by group or other (but must be readable by anyone).

Note that if the file exists, no implicit access rights are given to `user@LOCALREALM`.

Optionally, a set of files may be put in `~/.k5login.d` (a directory), in which case they will all be checked in the same manner as `.k5login`. The files may be called anything, but files starting with a hash (`#`), or ending with a tilde (`~`) are ignored. Subdirectories are not traversed. Note that this directory may not be checked by other Kerberos implementations.

If no configuration file exists, match user against local domains, ie `luser@LOCAL-REALMS-IN-CONFIGURATION-FILES`.

**Parameters:**

*context* Kerberos 5 context.  
*principal* principal to check if allowed to login  
*luser* local user id

**Returns:**

returns TRUE if access should be granted, FALSE otherwise.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_plugin\_register (krb5\_context context, enum krb5\_plugin\_type type, const char \* name, void \* symbol)**

Register a plugin symbol name of specific type.

**Parameters:**

*context* a Keberos context  
*type* type of plugin symbol  
*name* name of plugin symbol  
*symbol* a pointer to the named symbol

**Returns:**

In case of error a non zero error com\_err error is returned and the Kerberos error string is set.