

**NAME**

Heimdal Kerberos 5 address functions -

**Functions**

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_sockaddr2address** (krb5\_context context, const struct sockaddr \*sa, krb5\_address \*addr)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_sockaddr2port** (krb5\_context context, const struct sockaddr \*sa, int16\_t \*port)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_addr2sockaddr** (krb5\_context context, const krb5\_address \*addr, struct sockaddr \*sa, krb5\_socklen\_t \*sa\_size, int port)

KRB5\_LIB\_FUNCTION size\_t KRB5\_LIB\_CALL **krb5\_max\_sockaddr\_size** (void)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_sockaddr\_uninteresting** (const struct sockaddr \*sa)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_h\_addr2sockaddr** (krb5\_context context, int af, const char \*addr, struct sockaddr \*sa, krb5\_socklen\_t \*sa\_size, int port)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_h\_addr2addr** (krb5\_context context, int af, const char \*haddr, krb5\_address \*addr)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_anyaddr** (krb5\_context context, int af, struct sockaddr \*sa, krb5\_socklen\_t \*sa\_size, int port)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_print\_address** (const krb5\_address \*addr, char \*str, size\_t len, size\_t \*ret\_len)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_parse\_address** (krb5\_context context, const char \*string, krb5\_addresses \*addresses)

KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL **krb5\_address\_order** (krb5\_context context, const krb5\_address \*addr1, const krb5\_address \*addr2)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_address\_compare** (krb5\_context context, const krb5\_address \*addr1, const krb5\_address \*addr2)

KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL **krb5\_address\_search** (krb5\_context context, const krb5\_address \*addr, const krb5\_addresses \*addrlist)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_free\_address** (krb5\_context context, krb5\_address \*address)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_free\_addresses** (krb5\_context context, krb5\_addresses \*addresses)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_address** (krb5\_context context, const krb5\_address \*inaddr, krb5\_address \*outaddr)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_addresses** (krb5\_context context, const krb5\_addresses \*inaddr, krb5\_addresses \*outaddr)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_append\_addresses** (krb5\_context context, krb5\_addresses \*dest, const krb5\_addresses \*source)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_make\_addrport** (krb5\_context context,

```
krb5_address **res, const krb5_address *addr, int16_t port)
KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL krb5_address_prefixlen_boundary
(krb5_context context, const krb5_address *inaddr, unsigned long prefixlen, krb5_address *low,
krb5_address *high)
```

## Detailed Description

### Function Documentation

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_addr2sockaddr (krb5\_context context, const krb5\_address \* addr, struct sockaddr \* sa, krb5\_socklen\_t \* sa\_size, int port)**

krb5\_addr2sockaddr sets the 'struct sockaddr sockaddr' from addr and port. The argument sa\_size should initially contain the size of the sa and after the call, it will contain the actual length of the address. In case of the sa is too small to fit the whole address, the up to \*sa\_size will be stored, and then \*sa\_size will be set to the required length.

#### Parameters:

*context* a Keberos context  
*addr* the address to copy the from  
*sa* the struct sockaddr that will be filled in  
*sa\_size* pointer to length of sa, and after the call, it will contain the actual length of the address.  
*port* set port in sa.

#### Returns:

Return an error code or 0. Will return KRB5\_PROG\_ATYPE\_NOSUPP in case address type is not supported.

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_address\_compare (krb5\_context context, const krb5\_address \* addr1, const krb5\_address \* addr2)**

krb5\_address\_compare compares the addresses addr1 and addr2. Returns TRUE if the two addresses are the same.

#### Parameters:

*context* a Keberos context  
*addr1* address to compare  
*addr2* address to compare

#### Returns:

Return an TRUE is the address are the same FALSE if not

**KRB5\_LIB\_FUNCTION int KRB5\_LIB\_CALL krb5\_address\_order (krb5\_context context, const krb5\_address \* addr1, const krb5\_address \* addr2)**

`krb5_address_order` compares the addresses `addr1` and `addr2` so that it can be used for sorting addresses. If the addresses are the same address `krb5_address_order` will return 0. Behaves like `memcmp(2)`.

**Parameters:**

*context* a Keberos context  
*addr1* `krb5_address` to compare  
*addr2* `krb5_address` to compare

**Returns:**

< 0 if address `addr1` in 'less' then `addr2`. 0 if `addr1` and `addr2` is the same address, > 0 if `addr2` is 'less' then `addr1`.

**KRB5\_LIB\_FUNCTION** `krb5_error_code` **KRB5\_LIB\_CALL** `krb5_address_prefixlen_boundary`  
(`krb5_context` *context*, `const krb5_address *` *inaddr*, `unsigned long` *prefixlen*, `krb5_address *` *low*,  
`krb5_address *` *high*)

Calculate the boundary addresses of '`inaddr`'/'`prefixlen`' and store them in '`low`' and '`high`'.

**Parameters:**

*context* a Keberos context  
*inaddr* address in `prefixlen` that the bondery searched  
*prefixlen* width of boundary  
*low* lowest address  
*high* highest address

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION** `krb5_boolean` **KRB5\_LIB\_CALL** `krb5_address_search` (`krb5_context` *context*,  
`const krb5_address *` *addr*, `const krb5_addresses *` *addrlist*)

`krb5_address_search` checks if the address `addr` is a member of the address set list `addrlist`.

**Parameters:**

*context* a Keberos context.  
*addr* address to search for.  
*addrlist* list of addresses to look in for `addr`.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_anyaddr (krb5\_context context, int af, struct sockaddr \* sa, krb5\_socklen\_t \* sa\_size, int port)**

krb5\_anyaddr fills in a 'struct sockaddr sa' that can be used to bind(2) to. The argument sa\_size should initially contain the size of the sa, and after the call, it will contain the actual length of the address.

**Parameters:**

*context* a Keberos context

*af* address family

*sa* sockaddr

*sa\_size* length of sa.

*port* for to fill into sa.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_append\_addresses (krb5\_context context, krb5\_addresses \* dest, const krb5\_addresses \* source)**

krb5\_append\_addresses adds the set of addresses in source to dest. While copying the addresses, duplicates are also sorted out.

**Parameters:**

*context* a Keberos context

*dest* destination of copy operation

*source* addresses that are going to be added to dest

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_address (krb5\_context context, const krb5\_address \* inaddr, krb5\_address \* outaddr)**

krb5\_copy\_address copies the content of address inaddr to outaddr.

**Parameters:**

*context* a Keberos context

*inaddr* pointer to source address

*outaddr* pointer to destination address

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_addresses (krb5\_context context, const krb5\_addresses \* inaddr, krb5\_addresses \* outaddr)**  
krb5\_copy\_addresses copies the content of addresses inaddr to outaddr.

**Parameters:**

*context* a Keberos context  
*inaddr* pointer to source addresses  
*outaddr* pointer to destination addresses

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_free\_address (krb5\_context context, krb5\_address \* address)**  
krb5\_free\_address frees the data stored in the address that is allocated with any of the krb5\_address functions.

**Parameters:**

*context* a Keberos context  
*address* address to be freed.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_free\_addresses (krb5\_context context, krb5\_addresses \* addresses)**  
krb5\_free\_addresses frees the data stored in the address that is allocated with any of the krb5\_address functions.

**Parameters:**

*context* a Keberos context  
*addresses* addressses to be freed.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_h\_addr2addr (krb5\_context context, int af, const char \* haddr, krb5\_address \* addr)**  
krb5\_h\_addr2addr works like krb5\_h\_addr2sockaddr with the exception that it operates on a krb5\_address instead of a struct sockaddr.

**Parameters:**

*context* a Keberos context  
*af* address family  
*haddr* host address from struct hostent.  
*addr* returned krb5\_address.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_h\_addr2sockaddr (krb5\_context context, int af, const char \* addr, struct sockaddr \* sa, krb5\_socklen\_t \* sa\_size, int port)**

krb5\_h\_addr2sockaddr initializes a 'struct sockaddr sa' from af and the 'struct hostent' (see gethostbyname(3) ) h\_addr\_list component. The argument sa\_size should initially contain the size of the sa, and after the call, it will contain the actual length of the address.

**Parameters:**

*context* a Keberos context  
*af* addresses  
*addr* address  
*sa* returned struct sockaddr  
*sa\_size* size of sa  
*port* port to set in sa.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_make\_addrport (krb5\_context context, krb5\_address \*\* res, const krb5\_address \* addr, int16\_t port)**

Create an address of type KRB5\_ADDRESS\_ADDRPORT from (addr, port)

**Parameters:**

*context* a Keberos context  
*res* built address from addr/port  
*addr* address to use  
*port* port to use

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION size\_t KRB5\_LIB\_CALL krb5\_max\_sockaddr\_size (void)**

`krb5_max_sockaddr_size` returns the max size of the `.Li` struct `sockaddr` that the Kerberos library will return.

**Returns:**

Return an `size_t` of the maximum struct `sockaddr`.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_parse\_address (krb5\_context context, const char \* string, krb5\_addresses \* addresses)**

`krb5_parse_address` returns the resolved hostname in `string` to the `krb5_addresses` `addresses` .

**Parameters:**

*context* a Keberos context

*string*

*addresses*

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_print\_address (const krb5\_address \* addr, char \* str, size\_t len, size\_t \* ret\_len)**

`krb5_print_address` prints the address in `addr` to the string `string` that have the length `len`. If `ret_len` is not NULL, it will be filled with the length of the string if size were unlimited (not including the final NUL) .

**Parameters:**

*addr* address to be printed

*str* pointer string to print the address into

*len* length that will fit into area pointed to by 'str' .

*ret\_len* return length the str.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_sockaddr2address (krb5\_context context, const struct sockaddr \* sa, krb5\_address \* addr)**

`krb5_sockaddr2address` stores a address a 'struct `sockaddr`' `sa` in the `krb5_address` `addr`.

**Parameters:**

*context* a Keberos context

*sa* a struct `sockaddr` to extract the address from

*addr* an Kerberos 5 address to store the address in.

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_sockaddr2port (krb5\_context context, const struct sockaddr \* sa, int16\_t \* port)**

*krb5\_sockaddr2port* extracts a port (if possible) from a 'struct sockaddr.

**Parameters:**

*context* a Keberos context

*sa* a struct sockaddr to extract the port from

*port* a pointer to an int16\_t store the port in.

**Returns:**

Return an error code or 0. Will return KRB5\_PROG\_ATYPE\_NOSUPP in case address type is not supported.

**KRB5\_LIB\_FUNCTION krb5\_boolean KRB5\_LIB\_CALL krb5\_sockaddr\_uninteresting (const struct sockaddr \* sa)**

*krb5\_sockaddr\_uninteresting* returns TRUE for all .Fa sa that the kerberos library thinks are uninteresting. One example are link local addresses.

**Parameters:**

*sa* pointer to struct sockaddr that might be interesting.

**Returns:**

Return a non zero for uninteresting addresses.