**c-ares API 手册**

库版本：1.15.0

目录

[**ares\_cancel** 4](#_Toc13764544)

[**ares\_create\_query** 5](#_Toc13764545)

[**ares\_destroy** 7](#_Toc13764546)

[**ares\_destroy\_options** 8](#_Toc13764547)

[**ares\_dup** 9](#_Toc13764548)

[**ares\_expand\_name** 10](#_Toc13764549)

[**ares\_expand\_string** 12](#_Toc13764550)

[**ares\_fds** 14](#_Toc13764551)

[**ares\_free\_data** 16](#_Toc13764552)

[**ares\_free\_hostent** 18](#_Toc13764553)

[**ares\_free\_string** 19](#_Toc13764554)

[**ares\_get\_servers** 20](#_Toc13764555)

[**ares\_gethostbyaddr** 22](#_Toc13764556)

[**ares\_gethostbyname** 24](#_Toc13764557)

[**ares\_gethostbyname\_file** 26](#_Toc13764558)

[**ares\_getnameinfo** 28](#_Toc13764559)

[**ares\_getsock** 31](#_Toc13764560)

[**ares\_inet\_ntop** 33](#_Toc13764561)

[**ares\_inet\_pton** 35](#_Toc13764562)

[**ares\_init** 36](#_Toc13764563)

[**ares\_init\_options** 38](#_Toc13764564)

[**ares\_library\_cleanup** 43](#_Toc13764565)

[**ares\_library\_initialized** 45](#_Toc13764566)

[**ares\_mkquery** 46](#_Toc13764567)

[**ares\_parse\_a\_reply** 48](#_Toc13764568)

[**ares\_parse\_aaaa\_reply** 50](#_Toc13764569)

[**ares\_parse\_mx\_reply** 52](#_Toc13764570)

[**ares\_parse\_naptr\_reply** 54](#_Toc13764571)

[**ares\_parse\_ns\_reply** 56](#_Toc13764572)

[**ares\_parse\_ptr\_reply** 58](#_Toc13764573)

[**ares\_parse\_soa\_reply** 60](#_Toc13764574)

[**ares\_parse\_srv\_reply** 62](#_Toc13764575)

[**ares\_parse\_txt\_reply** 64](#_Toc13764576)

[**ares\_process** 66](#_Toc13764577)

[**ares\_process\_fd** 66](#_Toc13764578)

[**ares\_query** 68](#_Toc13764579)

[**ares\_save\_options** 71](#_Toc13764580)

[**ares\_search** 73](#_Toc13764581)

[**ares\_send** 76](#_Toc13764582)

[**ares\_set\_local\_dev** 78](#_Toc13764583)

[**ares\_set\_servers** 79](#_Toc13764584)

[**ares\_set\_servers\_csv** 81](#_Toc13764585)

[**ares\_set\_servers\_ports** 83](#_Toc13764586)

[**ares\_set\_servers\_ports\_csv** 85](#_Toc13764587)

[**ares\_set\_socket\_callback** 87](#_Toc13764588)

[**ares\_set\_socket\_configure\_callback** 88](#_Toc13764589)

[**ares\_set\_socket\_functions** 90](#_Toc13764590)

[**ares\_set\_sortlist** 93](#_Toc13764591)

[**ares\_strerror** 95](#_Toc13764592)

[**ares\_timeout** 96](#_Toc13764593)

[**ares\_version** 97](#_Toc13764594)

**ares\_cancel**

NAME

ares\_cancel - Cancel a resolve

SYNOPSIS

#include <ares.h>

void ares\_cancel(ares\_channel channel)

DESCRIPTION

The [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html) function cancels all lookups/requests made on the the name service channel identified by channel. [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html) invokes the callbacks for each pending query on the channel, passing a status of ARES\_ECANCELLED .These calls give the callbacks a chance to clean up any state which might have been stored in their arguments. If such a callback invocation adds a new request to the channel, that request will not be cancelled by the current invocation of [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html).

SEE ALSO

[ares\_init,](https://c-ares.haxx.se/ares_init.html) [ares\_destroy](https://c-ares.haxx.se/ares_destroy.html)

NOTES

This function was added in c-ares 1.2.0

c-ares 1.6.0 and earlier pass a status of , ARES\_ETIMEOUTinstead of ARES\_ECANCELLED .

AUTHOR

Dirk Manske

**ares\_create\_query**

NAME

ares\_create\_query - Compose a single-question DNS query buffer

SYNOPSIS

#include <ares.h>

int ares\_create\_query(const char \*name,

int dnsclass,

int type,

unsigned short id,

int rd,

unsigned char \*\*buf,

int \*buflen,

int max\_udp\_size)

DESCRIPTION

The [ares\_create\_query](https://c-ares.haxx.se/ares_create_query.html) function composes a DNS query with a single question. The parameter name gives the query name as a NUL-terminated C string of period-separated labels optionally ending with a period; periods and backslashes within a label must be escaped with a backlash.

The parameters dnsclass and type give the class and type of the query using the values defined in <arpa/nameser.h>.

The parameter id gives a 16-bit identifier for the query.

The parameter rd should be nonzero if recursion is desired, zero if not.

The query will be placed in an allocated buffer, a pointer to which will be stored in the variable pointed to by buf, and the length of which will be stored in the variable pointed to by buflen.

It is the caller's responsibility to free this buffer using [ares\_free\_string](https://c-ares.haxx.se/ares_free_string.html) when it is no longer needed. The parameter max\_udp\_size should be nonzero to activate EDNS. Usage of [ares\_create\_query](https://c-ares.haxx.se/ares_create_query.html) with max\_udp\_size set to zero is equivalent to using [ares\_mkquery](https://c-ares.haxx.se/ares_mkquery.html).

RETURN VALUES

ares\_create\_query can return any of the following values:

ARES\_SUCCESS Construction of the DNS query succeeded.

[ARES\_ENOTFOUND The query name name refers to a .onion domain name. See RFC 7686.](http://www.ietf.org/rfc/rfc7686.txt)

ARES\_EBADNAME The query name name could not be encoded as a domain name, either because it contained a zero-length label or because it contained a label of more than 63 characters.

ARES\_ENOMEM Memory was exhausted.

AVAILABILITY

Added in c-ares 1.10.0

SEE ALSO

[ares\_expand\_name,](https://c-ares.haxx.se/ares_expand_name.html) [ares\_free\_string,](https://c-ares.haxx.se/ares_free_string.html) [ares\_mkquery](https://c-ares.haxx.se/ares_mkquery.html)

AUTHOR

**ares\_destroy**

NAME

ares\_destroy - Destroy a resolver channel

SYNOPSIS

#include <ares.h>

void ares\_destroy(ares\_channel channel)

DESCRIPTION

The [ares\_destroy](https://c-ares.haxx.se/ares_destroy.html) function destroys the name service channel identified by channel, freeing all memory and closing all sockets used by the channel.

[ares\_destroy](https://c-ares.haxx.se/ares_destroy.html) invokes the callbacks for each pending query on the channel, passing a status of ARES\_EDESTRUCTION. These calls give the callbacks a chance to clean up any state which might have been stored in their arguments. A callback must not add new requests to a channel being destroyed.

SEE ALSO

[ares\_init,](https://c-ares.haxx.se/ares_init.html) [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html)

AUTHOR

Greg Hudson, MIT Information Systems

Copyright 1998 by the Massachusetts Institute of Technology.

**ares\_destroy\_options**

NAME

ares\_destroy\_options - Destroy options initialized with ares\_save\_options

SYNOPSIS

#include <ares.h> void ares\_destroy\_options(struct ares\_options \*options)

DESCRIPTION

The [ares\_destroy\_options](https://c-ares.haxx.se/ares_destroy_options.html) function destroys the options struct identified by Ioptions, freeing all memory allocated by [ares\_save\_options.](https://c-ares.haxx.se/ares_save_options.html)

SEE ALSO

[ares\_save\_options,](https://c-ares.haxx.se/ares_save_options.html) [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html)

AUTHOR

Brad House

Copyright 1998 by the Massachusetts Institute of Technology.

**ares\_dup**

NAME

ares\_dup - Duplicate a resolver channel

SYNOPSIS

#include <ares.h> int ares\_dup(ares\_channel \*dest, ares\_channel source)

DESCRIPTION

The [ares\_dup](https://c-ares.haxx.se/ares_dup.html) function duplicates an existing communications channel for name service lookups. If it returns successfully, [ares\_dup](https://c-ares.haxx.se/ares_dup.html) will set the variable pointed to by dest to a handle used to identify the name service channel. The caller should invoke [ares\_destroy](https://c-ares.haxx.se/ares_destroy.html) on the handle when the channel is no longer needed.

SEE ALSO

[ares\_destroy](https://c-ares.haxx.se/ares_destroy.html), [ares\_init,](https://c-ares.haxx.se/ares_init.html) [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html)

AVAILABILITY

[ares\_dup](https://c-ares.haxx.se/ares_dup.html) was added in c-ares 1.6.0

AUTHOR

Daniel Stenberg

**ares\_expand\_name**

NAME

ares\_expand\_name - Expand a DNS-encoded domain name

SYNOPSIS

#include <ares.h>

int ares\_expand\_name(const unsigned char \*encoded, const unsigned char \*abuf, int alen, char \*\*s, long \*enclen)

DESCRIPTION

The ares\_expand\_name function converts a DNS-encoded domain name to a dotseparated C string. The argument encoded gives the beginning of the encoded domain name, and the arguments abuf and alen give the containing message buffer (necessary for the processing of indirection pointers within the encoded domain name). The result is placed in a NUL-terminated allocated buffer, a pointer to which is stored in the variable pointed to by s . The length of the encoded name is stored in the variable pointed to by enclen so that the caller can advance past the encoded domain name to read further data in the message.

Use [ares\_free\_string(3)](https://c-ares.haxx.se/ares_free_string.html) to free the allocated hostname.

RETURN VALUES

ares\_expand\_name can return any of the following values:

ARES\_SUCCESS Expansion of the encoded name succeeded.

ARES\_EBADNAME The encoded domain name was malformed and could not be expanded.

ARES\_ENOMEM Memory was exhausted.

SEE ALSO

[ares\_mkquery (3)](https://c-ares.haxx.se/ares_mkquery.html) [ares\_free\_string (3)](https://c-ares.haxx.se/ares_free_string.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_expand\_string**

NAME

ares\_expand\_string - Expand a length encoded string

SYNOPSIS

#include <ares.h>

int ares\_expand\_string(const unsigned char \*encoded, const unsigned char \*abuf, int alen, unsigned char \*\*s, long \*enclen)

DESCRIPTION

The ares\_expand\_string function converts a length encoded string to a NUL-terminated C string. The argument encoded gives the beginning of the encoded string, and the arguments abuf and alen give the containing message buffer (necessary for the processing of indirection pointers within the encoded domain name). The result is placed in a NUL-terminated allocated buffer, a pointer to which is stored in the variable pointed to by s . The length of the encoded string is stored in the variable pointed to by enclen so that the caller can advance past the encoded string to read further data in the message.

RETURN VALUES

ares\_expand\_string can return any of the following values:

ARES\_SUCCESS Expansion of the encoded string succeeded.

ARES\_EBADSTR The encoded string was malformed and could not be expanded.

ARES\_ENOMEM Memory was exhausted.

SEE ALSO

[ares\_free\_string (3)](https://c-ares.haxx.se/ares_free_string.html)

AUTHOR

Dominick Meglio

**ares\_fds**

NAME

ares\_fds - return file descriptors to select on

SYNOPSIS

#include <ares.h>

int ares\_fds(ares\_channel channel, fd\_set \*read\_fds, fd\_set \*write\_fds)

DESCRIPTION

The [ares\_fds](https://c-ares.haxx.se/ares_fds.html) function retrieves the set of file descriptors which the calling application should select on for reading and writing for the processing of name service queries pending on the name service channel identified by channel.

File descriptors will be set in the file descriptor sets pointed to by read\_fds and write\_fds as appropriate. File descriptors already set in read\_fds and write\_fds will remain set; initialization of the file descriptor sets (using FD\_ZERO) is the responsibility of the caller.

RETURN VALUES

[ares\_fds](https://c-ares.haxx.se/ares_fds.html) returns a value that is one greater than the number of the highest socket set in either read\_fds or write\_fds. If no queries are active, [ares\_fds](https://c-ares.haxx.se/ares_fds.html) returns 0.

SEE ALSO

[ares\_timeout,](https://c-ares.haxx.se/ares_timeout.html) [ares\_process](https://c-ares.haxx.se/ares_process.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_free\_data**

NAME

ares\_free\_data - Free data allocated by several c-ares functions

SYNOPSIS

#include <ares.h>

void ares\_free\_data(void \*dataptr) cc file.c -lcares

DESCRIPTION

The [ares\_free\_data](https://c-ares.haxx.se/ares_free_data.html) function frees one or more data structures allocated and returned by several c-ares functions. Specifically the data returned by the following list of functions must be deallocated using this function.

[ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html) When used to free the data returned by ares\_get\_servers(3) this will free the whole linked list of ares\_addr\_node structures returned by ares\_get\_servers(3).

[ares\_parse\_srv\_reply](https://c-ares.haxx.se/ares_parse_srv_reply.html) When used to free the data returned by ares\_parse\_srv\_reply(3) this will free the whole linked list of ares\_srv\_reply structures returned by ares\_parse\_srv\_reply(3), along with any additional storage associated with those structures.

[ares\_parse\_mx\_reply](https://c-ares.haxx.se/ares_parse_mx_reply.html) When used to free the data returned by ares\_parse\_mx\_reply(3) this

will free the whole linked list of ares\_mx\_reply structures returned by ares\_parse\_mx\_reply(3), along with any additional storage associated with those structures.

[ares\_parse\_txt\_reply](https://c-ares.haxx.se/ares_parse_txt_reply.html) When used to free the data returned by ares\_parse\_txt\_reply(3) this will free the whole linked list of ares\_txt\_reply structures returned by ares\_parse\_txt\_reply(3), along with any additional storage associated with those structures.

[ares\_parse\_soa\_reply](https://c-ares.haxx.se/ares_parse_soa_reply.html) When used to free the data returned by ares\_parse\_soa\_reply(3) this will free the ares\_soa\_reply structure, along with any additional storage associated with those structure.

RETURN VALUE

The ares\_free\_data() function does not return a value.

AVAILABILITY

This function was first introduced in c-ares version 1.7.0.

SEE ALSO

[ares\_get\_servers,](https://c-ares.haxx.se/ares_get_servers.html) [ares\_parse\_srv\_reply,](https://c-ares.haxx.se/ares_parse_srv_reply.html) [ares\_parse\_mx\_reply](https://c-ares.haxx.se/ares_parse_mx_reply.html), [ares\_parse\_txt\_reply,](https://c-ares.haxx.se/ares_parse_txt_reply.html) [ares\_parse\_soa\_reply](https://c-ares.haxx.se/ares_parse_soa_reply.html)

AUTHOR

Yang Tse

**ares\_free\_hostent**

NAME

ares\_free\_hostent - Free host structure allocated by ares functions

SYNOPSIS

#include <ares.h>

void ares\_free\_hostent(struct hostent \*host)

DESCRIPTION

The ares\_free\_hostent function frees a struct hostent allocated by one of the functions [ares\_parse\_a\_reply(3),](https://c-ares.haxx.se/ares_parse_a_reply.html) [ares\_parse\_aaaa\_reply(3)](https://c-ares.haxx.se/ares_parse_aaaa_reply.html), or [ares\_parse\_ptr\_reply(3)](https://c-ares.haxx.se/ares_parse_ptr_reply.html).

NOTES

It is not necessary (and is not correct) to free the host structure passed to the callback functions for [ares\_gethostbyname(3)](https://c-ares.haxx.se/ares_gethostbyname.html) or [ares\_gethostbyaddr(3)](https://c-ares.haxx.se/ares_gethostbyaddr.html). c-ares will automatically free such host structures when the callback returns.

SEE ALSO

[ares\_parse\_a\_reply (3)](https://c-ares.haxx.se/ares_parse_a_reply.html) [ares\_parse\_aaaa\_reply (3)](https://c-ares.haxx.se/ares_parse_aaaa_reply.html) [ares\_parse\_ptr\_reply (3)](https://c-ares.haxx.se/ares_parse_ptr_reply.html) [ares\_parse\_ns\_reply (3)](https://c-ares.haxx.se/ares_parse_ns_reply.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_free\_string**

NAME

ares\_free\_string - Free strings allocated by ares functions

SYNOPSIS

#include <ares.h> void ares\_free\_string(void \*str)

DESCRIPTION

The [ares\_free\_string](https://c-ares.haxx.se/ares_free_string.html) function frees a string allocated by an ares function.

SEE ALSO

[ares\_mkquery](https://c-ares.haxx.se/ares_mkquery.html), [ares\_expand\_string](https://c-ares.haxx.se/ares_expand_string.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_get\_servers**

NAME

ares\_get\_servers, ares\_get\_servers\_ports - Retrieve name servers from an initialized ares\_channel

SYNOPSIS

#include <ares.h>

int ares\_get\_servers(ares\_channel channel, struct ares\_addr\_node \*\*servers) int ares\_get\_servers\_ports(ares\_channel channel, struct ares\_addr\_port\_node \*\*servers)

DESCRIPTION

The [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html) function retrieves name servers configuration from the channel data identified by channel , as a linked list of ares\_addr\_node structs storing a pointer to the first node at the address specified by servers .

The [ares\_get\_servers\_ports](https://c-ares.haxx.se/ares_get_servers_ports.html) function also retrieves any per-server port information that may have been previously configured, returning a linked list of ares\_addr\_port structures.

Function caller may traverse the returned name server linked list, or may use it directly as suitable input for the [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) / [ares\_set\_servers\_ports](https://c-ares.haxx.se/ares_set_servers_ports.html) functions, but shall not shrink or extend the list on its own.

Each node of the name server linked list is stored in memory dynamically allocated and managed by c-ares. It is the caller's responsibility to free the resulting linked list, using [ares\_free\_data](https://c-ares.haxx.se/ares_free_data.html) , once the caller does not need it any longer.

This function is capable of handling IPv4 and IPv6 name server addresses simultaneously, rendering [ares\_save\_options](https://c-ares.haxx.se/ares_save_options.html) with optmask ARES\_OPT\_SERVERS functionally obsolete except for IPv4-only name server usage.

RETURN VALUES

This function may return any of the following values:

ARES\_SUCCESS The name servers configuration was successfully retrieved

ARES\_ENOMEM The memory was exhausted

ARES\_ENODATA The channel data identified by channel was invalid.

SEE ALSO

[ares\_set\_servers,](https://c-ares.haxx.se/ares_set_servers.html) [ares\_init\_options,](https://c-ares.haxx.se/ares_init_options.html) [ares\_save\_options](https://c-ares.haxx.se/ares_save_options.html)

AVAILABILITY

[ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html) was added in c-ares 1.7.1; [ares\_get\_servers\_ports](https://c-ares.haxx.se/ares_get_servers_ports.html) was added in c-ares 1.11.0.

AUTHOR

Implementation of this function and associated library internals are based on code, comments and feedback provided in November and December of 2008 by Daniel Stenberg, Gregor Jasny, Phil Blundell and Yang Tse, December 2009 by Cedric Bail, February 2010 by Jakub Hrozek. On March 2010 Yang Tse shuffled all the bits and this function popped out.

Copyright 1998 by the Massachusetts Institute of Technology.

**ares\_gethostbyaddr**

NAME

ares\_gethostbyaddr - Initiate a host query by address

SYNOPSIS

#include <ares.h>

typedef void (\*ares\_host\_callback)(void \*arg, int status, int timeouts, struct hostent

\*hostent)

void ares\_gethostbyaddr(ares\_channel channel, const void \*addr, int addrlen, int family, ares\_host\_callback callback, void \*arg)

DESCRIPTION

The ares\_gethostbyaddr function initiates a host query by address on the name service channel identified by channel . The parameters addr and addrlen give the address as a series of bytes, and family gives the type of address. When the query is complete or has failed, the ares library will invoke callback. Completion or failure of the query may happen immediately, or may happen during a later call to [ares\_process(3),](https://c-ares.haxx.se/ares_process.html) [ares\_destroy(3)](https://c-ares.haxx.se/ares_destroy.html) or [ares\_cancel(3).](https://c-ares.haxx.se/ares_cancel.html)

The callback argument arg is copied from the ares\_gethostbyaddr argument arg . The callback argument status indicates whether the query succeeded and, if not, how it failed. It may have any of the following values:

ARES\_SUCCESS The host lookup completed successfully.

ARES\_ENOTIMP The ares library does not know how to look up addresses of type family

.

ARES\_ENOTFOUND The address addr was not found.

ARES\_ENOMEM Memory was exhausted.

ARES\_ECANCELLED The query was cancelled.

ARES\_EDESTRUCTION The name service channel channel is being destroyed; the query will not be completed.

The callback argument timeouts reports how many times a query timed out during the execution of the given request.

On successful completion of the query, the callback argument hostent points to a struct hostent containing the name of the host returned by the query. The callback need not

and should not attempt to free the memory pointed to by hostent ; the ares library will free it when the callback returns. If the query did not complete successfully, hostent will be NULL .

SEE ALSO

[ares\_process (3)](https://c-ares.haxx.se/ares_process.html) [ares\_gethostbyname (3)](https://c-ares.haxx.se/ares_gethostbyname.html)

AUTHOR

Greg Hudson, MIT Information Systems

Copyright 1998 by the Massachusetts Institute of Technology.

**ares\_gethostbyname**

NAME

ares\_gethostbyname - Initiate a host query by name

SYNOPSIS

#include <ares.h>

typedef void (\*ares\_host\_callback)(void \*arg, int status, int timeouts, struct hostent \*hostent)

void ares\_gethostbyname(ares\_channel channel, const char \*name, int family, ares\_host\_callback callback, void \*arg)

DESCRIPTION

The ares\_gethostbyname function initiates a host query by name on the name service channel identified by channel . The parameter name gives the hostname as a NULterminated C string, and family gives the desired type of address for the resulting host entry. When the query is complete or has failed, the ares library will invoke callback. Completion or failure of the query may happen immediately, or may happen during a later call to [ares\_process,](https://c-ares.haxx.se/ares_process.html) [ares\_destroy](https://c-ares.haxx.se/ares_destroy.html) or [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html).

The callback argument arg is copied from the ares\_gethostbyname argument arg . The callback argument status indicates whether the query succeeded and, if not, how it failed. It may have any of the following values:

ARES\_SUCCESS The host lookup completed successfully.

ARES\_ENOTIMP The ares library does not know how to find addresses of type family .

ARES\_EBADNAME The hostname name is composed entirely of numbers and periods, but is not a valid representation of an Internet address.

ARES\_ENODATA There was no data returned to extract a result from.

ARES\_ENOTFOUND The name name was not found.

ARES\_ENOMEM Memory was exhausted.

ARES\_ECANCELLED The query was cancelled.

ARES\_EDESTRUCTION The name service channel channel is being destroyed; the query will not be completed.

The callback argument timeouts reports how many times a query timed out during the execution of the given request.

On successful completion of the query, the callback argument hostent points to a struct hostent containing the name of the host returned by the query. The callback need not and should not attempt to free the memory pointed to by hostent ; the ares library will free it when the callback returns. If the query did not complete successfully, hostent will be NULL .

SEE ALSO

[ares\_process](https://c-ares.haxx.se/ares_process.html), [ares\_gethostbyaddr](https://c-ares.haxx.se/ares_gethostbyaddr.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_gethostbyname\_file**

NAME

ares\_gethostbyname\_file - Lookup a name in the system's hosts file

SYNOPSIS

#include <ares.h>

int ares\_gethostbyname\_file(ares\_channel channel, const char \*name, int family, struct hostent \*\*host)

DESCRIPTION

The ares\_gethostbyname\_file function performs a host lookup by name against the system's hosts file (or equivalent local hostname database). The channel parameter is required, but no asynchronous queries are performed. Instead, the lookup is done via the same mechanism used to perform 'f' lookups (see the lookups options field in [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html)). The parameter name gives the hostname as a NUL-terminated C string, and family gives the desired type of address for the resulting host entry.

The return value indicates whether the query succeeded and, if not, how it failed. It may have any of the following values:

ARES\_SUCCESS The host lookup completed successfully and host now points to the result (and must be freed with [ares\_free\_hostent)](https://c-ares.haxx.se/ares_free_hostent.html).

ARES\_ENOTFOUND The hostname name was not found.

ARES\_EFILE There was a file I/O error while performing the lookup.

ARES\_ENOMEM Memory was exhausted.

On successful completion of the query, the pointer pointed to by host points to a struct hostent containing the address of the host returned by the lookup. The user must free the memory pointed to by host when finished with it by calling [ares\_free\_hostent.](https://c-ares.haxx.se/ares_free_hostent.html) If the lookup did not complete successfully, host will be NULL .

AVAILABILITY

Added in c-ares 1.5.4

SEE ALSO

[ares\_gethostbyname,](https://c-ares.haxx.se/ares_gethostbyname.html) [ares\_free\_hostent](https://c-ares.haxx.se/ares_free_hostent.html), [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html)

AUTHOR

Brad Spencer

**ares\_getnameinfo**

NAME

ares\_getnameinfo - Address-to-nodename translation in protocol-independent manner

SYNOPSIS

#include <ares.h>

typedef void (\*ares\_nameinfo\_callback)(void \*arg, int status, int timeouts, char \*node, char \*service) void ares\_getnameinfo(ares\_channel channel, const struct sockaddr \*sa, ares\_socklen\_t salen, int flags, ares\_nameinfo\_callback callback, void \*arg)

DESCRIPTION

The ares\_getnameinfo function is defined for protocol-independent address translation. The function is a combination of [ares\_gethostbyaddr](https://c-ares.haxx.se/ares_gethostbyaddr.html) and getservbyport(3). The function will translate the address either by executing a host query on the name service channel identified by channel or it will attempt to resolve it locally if possible. The parameters sa and len give the address as a sockaddr structure, and flags gives the options that the function will use. Valid flags are listed below:

ARES\_NI\_NOFQDN Only the nodename portion of the FQDN is returned for local hosts.

ARES\_NI\_NUMERICHOST The numeric form of the hostname is returned rather than the name.

ARES\_NI\_NAMEREQD An error is returned if the hostname cannot be found in the DNS.

ARES\_NI\_NUMERICSERV The numeric form of the service is returned rather than the name.

ARES\_NI\_TCP The service name is to be looked up for the TCP protocol.

ARES\_NI\_UDP The service name is to be looked up for the UDP protocol.

ARES\_NI\_SCTP The service name is to be looked up for the SCTP protocol.

ARES\_NI\_DCCP The service name is to be looked up for the DCCP protocol.

ARES\_NI\_NUMERICSCOPE The numeric form of the scope ID is returned rather than the name.

ARES\_NI\_LOOKUPHOST A hostname lookup is being requested.

ARES\_NI\_LOOKUPSERVICE A service name lookup is being requested.

When the query is complete or has failed, the ares library will invoke callback. Completion or failure of the query may happen immediately, or may happen during a later call to [ares\_process,](https://c-ares.haxx.se/ares_process.html) [ares\_destroy](https://c-ares.haxx.se/ares_destroy.html) or [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html).

The callback argument arg is copied from the ares\_getnameinfo argument arg . The callback argument status indicates whether the query succeeded and, if not, how it failed. It may have any of the following values:

ARES\_SUCCESS The host lookup completed successfully.

ARES\_ENOTIMP The ares library does not know how to look up addresses of type family

.

ARES\_ENOTFOUND The address addr was not found.

ARES\_ENOMEM Memory was exhausted.

ARES\_ECANCELLED The query was cancelled.

ARES\_EDESTRUCTION The name service channel channel is being destroyed; the query will not be completed.

ARES\_EBADFLAGS The flags parameter contains an illegal value.

The callback argument timeouts reports how many times a query timed out during the execution of the given request.

On successful completion of the query, the callback argument node contains a string representing the hostname (assuming ARES\_NI\_LOOKUPHOST was specified).

Additionally, service contains a string representing the service name (assuming

ARES\_NI\_LOOKUPSERVICE was specified). If the query did not complete successfully, or one of the values was not requested, node or service will be NULL .

SEE ALSO

[ares\_process](https://c-ares.haxx.se/ares_process.html),

AUTHOR

Dominick Meglio

**ares\_getsock**

NAME

ares\_getsock - get socket descriptors to wait on

SYNOPSIS

#include <ares.h>

int ares\_getsock(ares\_channel channel, ares\_socket\_t \*socks, int numsocks);

DESCRIPTION

The ares\_getsock function retrieves the set of socket descriptors which the calling application should wait on for reading and/or writing for the processing of name service queries pending on the name service channel identified by channel . Socket descriptors will be set in the socket descriptor array pointed to by socks. numsocks is the size of the given array in number of ints.

This function can only return information about up to 16 sockets. If more are in use (however unlikely that is), they are simply not reported back.

RETURN VALUES

ares\_getsock returns a bitmask for what actions to wait for on the different sockets. The ares.h header file provides these convenience macros to extract the information appropriately:

#define ARES\_GETSOCK\_MAXNUM 16 /\* ares\_getsock() can return info about this many sockets \*/ #define ARES\_GETSOCK\_READABLE(bits,num) (bits & (1<< (num))) #define

ARES\_GETSOCK\_WRITABLE(bits,num) (bits & (1 << ((num) + \ ARES\_GETSOCK\_MAXNUM)))

NOTES

This function was added in c-ares 1.3.1

SEE ALSO

[ares\_timeout (3)](https://c-ares.haxx.se/ares_timeout.html) [ares\_fds (3)](https://c-ares.haxx.se/ares_fds.html) [ares\_process (3)](https://c-ares.haxx.se/ares_process.html)

**ares\_inet\_ntop**

NAME

ares\_inet\_ntop - convert a network format address to presentation format

SYNOPSIS

#include <ares.h>

const char \* ares\_inet\_ntop(int af, const void \*src, char \*dst, ares\_socklen\_t size);

DESCRIPTION

This is a portable version with the identical functionality of the commonly available inet\_ntop.

The ares\_inet\_ntop() function converts a numeric address into a text string suitable for presentation. The af argument shall specify the family of the address. This can be AF\_INET or AF\_INET6. The src argument points to a buffer holding an IPv4 address if the af argument is AF\_INET, or an IPv6 address if the af argument is AF\_INET6; the address must be in network byte order. The dst argument points to a buffer where the function stores the resulting text string; it shall not be NULL. The size argument specifies the size of this buffer, which shall be large enough to hold the text string (INET\_ADDRSTRLEN (16) characters for IPv4, INET6\_ADDRSTRLEN (46) characters for IPv6).

SEE ALSO

[ares\_init(3)](https://c-ares.haxx.se/ares_init.html) [ares\_inet\_pton(3)](https://c-ares.haxx.se/ares_inet_pton.html)

AVAILABILITY

made properly publicly available in c-ares for real in version 1.10.0

AUTHOR

Daniel Stenberg

**ares\_inet\_pton**

NAME

ares\_inet\_pton - convert an IPv4 or IPv6 address from text to binary form

SYNOPSIS

#include <ares.h>

const char \*ares\_inet\_pton(int af, const char \*src, void \*dst);

DESCRIPTION

This is a portable version with the identical functionality of the commonly available inet\_pton.

The ares\_inet\_pton() function converts the address in its standard text presentation form into its numeric binary form. The af argument shall specify the family of the address. The AF\_INET and AF\_INET6 address families shall be supported. The src argument points to the string being passed in. The dst argument points to a buffer into which the function stores the numeric address; this shall be large enough to hold the numeric address (32 bits for AF\_INET, 128 bits for AF\_INET6).

SEE ALSO

[ares\_init(3)](https://c-ares.haxx.se/ares_init.html) [ares\_inet\_ntop(3)](https://c-ares.haxx.se/ares_inet_ntop.html)

AVAILABILITY

made properly publicly available in c-ares for real in version 1.10.0

AUTHOR

Daniel Stenberg

**ares\_init**

NAME

ares\_init - Initialize a resolver channel

SYNOPSIS

#include <ares.h>

int ares\_init(ares\_channel \*channelptr)

DESCRIPTION

The [ares\_init](https://c-ares.haxx.se/ares_init.html) function initializes a communications channel for name service lookups. If it returns successfully, [ares\_init](https://c-ares.haxx.se/ares_init.html) will set the variable pointed to by channelptr to a handle used to identify the name service channel. The caller should invoke [ares\_destroy](https://c-ares.haxx.se/ares_destroy.html) on the handle when the channel is no longer needed.

The [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) function is provide to offer more init alternatives.

RETURN VALUES

[ares\_init](https://c-ares.haxx.se/ares_init.html) can return any of the following values:

ARES\_SUCCESS Initialization succeeded.

ARES\_EFILE A configuration file could not be read.

ARES\_ENOMEM The process's available memory was exhausted.

ARES\_ENOTINITIALIZED c-ares library initialization not yet performed.

NOTES

When initializing from /etc/resolv.conf, [ares\_init](https://c-ares.haxx.se/ares_init.html) reads the domain and search directives to allow lookups of short names relative to the domains specified. The domain and search directives override one another. If more that one instance of either domain or search directives is specified, the last occurrence wins. For more information, please see the resolv.conf (5)manual page.

SEE ALSO

[ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html), [ares\_destroy,](https://c-ares.haxx.se/ares_destroy.html) [ares\_dup](https://c-ares.haxx.se/ares_dup.html), [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html), [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_init\_options**

NAME

ares\_init\_options - Initialize a resolver channel

SYNOPSIS

#include <ares.h>

struct ares\_options {

int flags;

int timeout; /\* in seconds or milliseconds, depending on options \*/ int tries; int ndots;

unsigned short udp\_port; unsigned short tcp\_port;

int socket\_send\_buffer\_size;

int socket\_receive\_buffer\_size;

struct in\_addr \*servers;

int nservers;

char \*\*domains;

int ndomains;

char \*lookups;

ares\_sock\_state\_cb sock\_state\_cb;

void \*sock\_state\_cb\_data;

struct apattern \*sortlist;

int nsort;

int ednspsz;

char \*resolvconf\_path;

};

int ares\_init\_options(ares\_channel \*channelptr, struct ares\_options \*options,

int optmask)

DESCRIPTION

The [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) function initializes a communications channel for name service lookups. If it returns successfully, [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) will set the variable pointed to by channelptr to a handle used to identify the name service channel. The caller should invoke [ares\_destroy](https://c-ares.haxx.se/ares_destroy.html) on the handle when the channel is no longer needed.

The optmask parameter generally specifies which fields in the structure pointed to by options are set, as follows:

ARES\_OPT\_FLAGS int flags;

Flags controlling the behavior of the resolver. See below for a description of possible flag values.

ARES\_OPT\_TIMEOUT int timeout;

The number of seconds each name server is given to respond to a query on the first try.

(After the first try, the timeout algorithm becomes more complicated, but scales linearly

with the value of timeout.) The default is five seconds. This option is being deprecated by ARES\_OPT\_TIMEOUTMS starting in c-ares 1.5.2.

ARES\_OPT\_TIMEOUTMS int timeout;

The number of milliseconds each name server is given to respond to a query on the first try. (After the first try, the timeout algorithm becomes more complicated, but scales linearly with the value of timeout.) The default is five seconds. Note that this option is specified with the same struct field as the former ARES\_OPT\_TIMEOUT, it is but the option bits that tell c-ares how to interpret the number. This option was added in c-ares 1.5.2.

ARES\_OPT\_TRIES int tries;

The number of tries the resolver will try contacting each name server before giving up. The default is four tries.

ARES\_OPT\_NDOTS int ndots;

The number of dots which must be present in a domain name for it to be queried for "as is" prior to querying for it with the default domain extensions appended. The default value is 1 unless set otherwise by resolv.conf or the RES\_OPTIONS environment variable.

ARES\_OPT\_UDP\_PORT unsigned short udp\_port;

The port to use for queries over UDP, in network byte order. The default value is 53 (in network byte order), the standard name service port.

ARES\_OPT\_TCP\_PORT unsigned short tcp\_port;

The port to use for queries over TCP, in network byte order. The default value is 53 (in network byte order), the standard name service port.

ARES\_OPT\_SERVERS struct in\_addr \*servers; int nservers;

The list of IPv4 servers to contact, instead of the servers specified in resolv.conf or the local named. In order to allow specification of either IPv4 or IPv6 name servers, the [ares\_set\_serversf](https://c-ares.haxx.se/ares_set_servers.html)unction must be used instead.

ARES\_OPT\_DOMAINS char \*\*domains; int ndomains;

The domains to search, instead of the domains specified in resolv.conf or the domain derived from the kernel hostname variable.

ARES\_OPT\_LOOKUPS char \*lookups;

The lookups to perform for host queries. lookups should be set to a string of the characters "b" or "f", where "b" indicates a DNS lookup and "f" indicates a lookup in the hosts file.

ARES\_OPT\_SOCK\_STATE\_CB void (\*sock\_state\_cb)(void \*data, ares\_socket\_t socket\_fd, int readable, int writable); void \*sock\_state\_cb\_data;

A callback function to be invoked when a socket changes state. socket\_fd will be passed the socket whose state has changed; readable will be set to true if the socket should listen for read events, and writable will be set to true if the socket should listen for write events. The value of sock\_state\_cb\_data will be passed as the data argument.

ARES\_OPT\_SORTLIST struct apattern \*sortlist; int nsort;

A list of IP address ranges that specifies the order of preference that results from ares\_gethostbyname should be returned in. Note that this can only be used with a sortlist retrieved via [ares\_save\_options](https://c-ares.haxx.se/ares_save_options.html) (because struct apattern is opaque); to set a fresh sort list, use [ares\_set\_sortlist.](https://c-ares.haxx.se/ares_set_sortlist.html)

ARES\_OPT\_SOCK\_SNDBUF int socket\_send\_buffer\_size;

The send buffer size to set for the socket.

ARES\_OPT\_SOCK\_RCVBUF int socket\_receive\_buffer\_size; The receive buffer size to set for the socket.

ARES\_OPT\_EDNSPSZ int ednspsz;

ARES\_OPT\_RESOLVCONF char \*resolvconf\_path; The path to use for reading the resolv.conf file. The resolvconf\_path should be set to a path string, and will be honoured on \*nix like systems. The default is /etc/resolv.conf

The message size to be advertized in EDNS; only takes effect if the ARES\_FLAG\_EDNS flag is set.

The optmask parameter also includes options without a corresponding field in the ares\_options structure, as follows:

ARES\_OPT\_ROTATE Perform round-robin selection of the nameservers configured for the channel for each resolution.

ARES\_OPT\_NOROTATE Do not perform round-robin nameserver selection; always use the list of nameservers in the same order.

The flags field should be the bitwise or of some subset of the following values:

ARES\_FLAG\_USEVC Always use TCP queries (the "virtual circuit") instead of UDP queries. Normally, TCP is only used if a UDP query yields a truncated result.

ARES\_FLAG\_PRIMARY Only query the first server in the list of servers to query.

ARES\_FLAG\_IGNTC If a truncated response to a UDP query is received, do not fall back to TCP; simply continue on with the truncated response.

ARES\_FLAG\_NORECURSE Do not set the "recursion desired" bit on outgoing queries, so that the name server being contacted will not try to fetch the answer from other servers if it doesn't know the answer locally. Be aware that ares will not do the recursion for you. Recursion must be handled by the application calling ares if ARES\_FLAG\_NORECURSE is set.

ARES\_FLAG\_STAYOPEN Do not close communications sockets when the number of active queries drops to zero.

ARES\_FLAG\_NOSEARCH Do not use the default search domains; only query hostnames as-is or as aliases.

ARES\_FLAG\_NOALIASES Do not honor the HOSTALIASES environment variable, which normally specifies a file of hostname translations.

ARES\_FLAG\_NOCHECKRESP Do not discard responses with the SERVFAIL, NOTIMP, or REFUSED response code or responses whose questions don't match the questions in the request. Primarily useful for writing clients which might be used to test or debug name servers.

ARES\_FLAG\_EDNS Include an EDNS pseudo-resource record (RFC 2671) in generated requests.

RETURN VALUES

[ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) can return any of the following values:

ARES\_SUCCESS Initialization succeeded.

ARES\_EFILE A configuration file could not be read.

ARES\_ENOMEM The process's available memory was exhausted.

ARES\_ENOTINITIALIZED c-ares library initialization not yet performed.

NOTES

When initializing from /etc/resolv.conf, (or, alternatively when specified by the resolvconf\_path path location) [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) reads the domain and search directives to allow lookups of short names relative to the domains specified. The domain and search directives override one another. If more that one instance of either domain or search directives is specified, the last occurrence wins. For more information, please see the resolv.conf (5)manual page.

SEE ALSO

[ares\_init,](https://c-ares.haxx.se/ares_init.html) [ares\_destroy,](https://c-ares.haxx.se/ares_destroy.html) [ares\_dup](https://c-ares.haxx.se/ares_dup.html), [ares\_library\_init,](https://c-ares.haxx.se/ares_library_init.html) [ares\_save\_options](https://c-ares.haxx.se/ares_save_options.html), [ares\_set\_servers,](https://c-ares.haxx.se/ares_set_servers.html) [ares\_set\_sortlist](https://c-ares.haxx.se/ares_set_sortlist.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_library\_cleanup**

NAME

ares\_library\_cleanup - c-ares library deinitialization

SYNOPSIS

#include <ares.h>

void ares\_library\_cleanup(void)

DESCRIPTION

The ares\_library\_cleanup function uninitializes the c-ares library, freeing all resources previously acquired by [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html) when the library was initialized, provided there was only one single previous call to [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html). If there was more than one previous call to [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html), this function uninitializes the c-ares library only if it is the call matching the call to [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html) which initialized the library (usually the very first call to [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html)). Other calls to [ares\_library\_cleanup](https://c-ares.haxx.se/ares_library_cleanup.html) have no effect other than decrementing an internal counter.

This function must be called when the program using c-ares will no longer need any cares function. Once the program has called [ares\_library\_cleanup](https://c-ares.haxx.se/ares_library_cleanup.html) sufficiently often such that the library is uninitialised, it shall not make any further call to any c-ares function.

This function does not cancel any pending c-ares lookups or requests previously done. Program must use [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html) for this purpose.

This function is not thread safe. You have to call it once the program is about to terminate, but this call must be done once the program has terminated every single thread that it could have initiated. This is required to avoid potential race conditions in library deinitialization, and also due to the fact that [ares\_library\_cleanup](https://c-ares.haxx.se/ares_library_cleanup.html) might call functions from other libraries that are thread unsafe, and could conflict with any other thread that is already using these other libraries.

Win32/64 application DLLs shall not call [ares\_library\_cleanup](https://c-ares.haxx.se/ares_library_cleanup.html) from the DllMain function. Doing so will produce deadlocks and other problems.

AVAILABILITY

This function was first introduced in c-ares version 1.7.0 along with the definition of preprocessor symbol CARES\_HAVE\_ARES\_LIBRARY\_CLEANUP as an indication of the

availability of this function. Reference counting in ares\_library\_init() and

ares\_library\_cleanup(), which requires calls to the former function to match calls to the latter, is present since c-ares version 1.10.0. Earlier versions would deinitialize the library on the first call to ares\_library\_cleanup().

Since the introduction of this function, it is absolutely mandatory to call it for any Win32/64 program using c-ares.

Non-Win32/64 systems can still use c-ares version 1.7.0 without calling

[ares\_library\_cleanup](https://c-ares.haxx.se/ares_library_cleanup.html) due to the fact that currently it is nearly a do-nothing function on non-Win32/64 platforms.

SEE ALSO

[ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html), [ares\_cancel](https://c-ares.haxx.se/ares_cancel.html)

AUTHOR

Yang Tse

**ares\_library\_initialized**

NAME

ares\_library\_initialized - get the initialization state

SYNOPSIS

#include <ares.h>

int ares\_library\_initialized(void)

DESCRIPTION

Returns information if c-ares needs to get initialized.

RETURN VALUE

ARES\_ENOTINITIALIZED if not initialized and ARES\_SUCCESS if no initialization is needed.

AVAILABILITY

This function was first introduced in c-ares version 1.11.0

SEE ALSO

[ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html), [ares\_library\_cleanup](https://c-ares.haxx.se/ares_library_cleanup.html)

**ares\_mkquery**

NAME

ares\_mkquery - Compose a single-question DNS query buffer

SYNOPSIS

#include <ares.h>

int ares\_mkquery(const char \*name, int dnsclass, int type, unsigned short id, int rd, unsigned char \*\*buf, int \*buflen)

DESCRIPTION

Deprecated function. See [ares\_create\_query](https://c-ares.haxx.se/ares_create_query.html) instead!

The ares\_mkquery function composes a DNS query with a single question. The parameter name gives the query name as a NUL-terminated C string of period-separated labels optionally ending with a period; periods and backslashes within a label must be escaped with a backlash. The parameters dnsclass and type give the class and type of the query using the values defined in <arpa/nameser.h> .The parameter id gives a 16-bit identifier for the query. The parameter rd should be nonzero if recursion is desired, zero if not. The query will be placed in an allocated buffer, a pointer to which will be stored in the variable pointed to by buf , and the length of which will be stored in the variable pointed to by buflen . It is the caller's responsibility to free this buffer using [ares\_free\_string](https://c-ares.haxx.se/ares_free_string.html) when it is no longer needed.

Usage of [ares\_mkquery](https://c-ares.haxx.se/ares_mkquery.html) is deprecated, whereas the function is equivalent to [ares\_create\_query](https://c-ares.haxx.se/ares_create_query.html) with max\_udp\_size set to 0.

RETURN VALUES

ares\_mkquery can return any of the following values:

ARES\_SUCCESS Construction of the DNS query succeeded.

[ARES\_ENOTFOUND The query name name refers to a .onion domain name. See RFC 7686.](http://www.ietf.org/rfc/rfc7686.txt)

ARES\_EBADNAME The query name name could not be encoded as a domain name, either because it contained a zero-length label or because it contained a label of more than 63 characters.

ARES\_ENOMEM Memory was exhausted.

SEE ALSO

[ares\_expand\_name,](https://c-ares.haxx.se/ares_expand_name.html) [ares\_free\_string](https://c-ares.haxx.se/ares_free_string.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_parse\_a\_reply**

NAME

ares\_parse\_a\_reply - Parse a reply to a DNS query of type A

SYNOPSIS

#include <ares.h>

int ares\_parse\_a\_reply(const unsigned char \*abuf, int alen, struct hostent \*\*host, struct ares\_addrttl \*addrttls, int \*naddrttls);

DESCRIPTION

The ares\_parse\_a\_reply function parses the response to a query of type A into a struct hostent and/or an array of struct ares\_addrttls . The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by host , if host is nonnull. It is the caller's

responsibility to free the resulting host structure using [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html) when it is no longer needed.

If addrttls and naddrttls are both nonnull, then up to \*naddrttls struct ares\_addrttl records are stored in the array pointed to by addrttls, and then \*naddrttls is set to the number of records so stored. Note that the memory for these records is supplied by the caller.

RETURN VALUES

ares\_parse\_a\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

SEE ALSO

[ares\_gethostbyname (3)](https://c-ares.haxx.se/ares_gethostbyname.html) [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_parse\_aaaa\_reply**

NAME

ares\_parse\_aaaa\_reply - Parse a reply to a DNS query of type AAAA

SYNOPSIS

#include <ares.h>

int ares\_parse\_aaaa\_reply(const unsigned char \*abuf, int alen, struct hostent \*\*host, struct ares\_addr6ttl \*addrttls, int \*naddrttls);

DESCRIPTION

The ares\_parse\_aaaa\_reply function parses the response to a query of type AAAA into a struct hostent and/or an array of struct ares\_addr6ttl . The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by host , if host is nonnull. It is the caller's responsibility to free the resulting host structure using [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html) when it is no longer needed.

If addrttls and naddrttls are both nonnull, then up to \*naddrttls struct ares\_addr6ttl records are stored in the array pointed to by addrttls, and then \*naddrttls is set to the number of records so stored. Note that the memory for these records is supplied by the caller.

RETURN VALUES

ares\_parse\_aaaa\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

SEE ALSO

[ares\_gethostbyname (3)](https://c-ares.haxx.se/ares_gethostbyname.html) [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html)

AUTHOR

Dominick Meglio

**ares\_parse\_mx\_reply**

NAME

ares\_parse\_mx\_reply - Parse a reply to a DNS query of type MX

SYNOPSIS

#include <ares.h>

int ares\_parse\_mx\_reply(const unsigned char\* abuf, int alen, struct ares\_mx\_reply\*\* mx\_out);

DESCRIPTION

The ares\_parse\_mx\_reply function parses the response to a query of type MX into a linked list of struct ares\_mx\_reply The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by mx\_out . It is the caller's responsibility to free the resulting mx\_out structure when it is no longer needed using the function ares\_free\_data The structure ares\_mx\_reply contains the following fields:

struct ares\_mx\_reply {

struct ares\_mx\_reply \*next;

char \*host;

unsigned short priority;

};

RETURN VALUES

ares\_parse\_mx\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

AVAILABILITY

This function was first introduced in c-ares version 1.7.2.

SEE ALSO

[ares\_query](https://c-ares.haxx.se/ares_query.html), [ares\_free\_data](https://c-ares.haxx.se/ares_free_data.html)

AUTHOR

Written by Jeremy Lal <kapouer@melix.org>

**ares\_parse\_naptr\_reply**

NAME

ares\_parse\_naptr\_reply - Parse a reply to a DNS query of type NAPTR

SYNOPSIS

#include <ares.h>

int ares\_parse\_naptr\_reply(const unsigned char\* abuf, int alen, struct ares\_naptr\_reply\*\* naptr\_out);

DESCRIPTION

The ares\_parse\_naptr\_reply function parses the response to a query of type NAPTR into a linked list of struct ares\_naptr\_reply The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by naptr\_out . It is the caller's responsibility to free the resulting naptr\_out structure when it is no longer needed using the function ares\_free\_data The structure ares\_naptr\_reply contains the following fields:

struct ares\_naptr\_reply {

struct ares\_naptr\_reply \*next;

unsigned char \*flags;

unsigned char \*service;

unsigned char \*regexp;

char \*replacement;

unsigned short order;

unsigned short preference;

};

RETURN VALUES

ares\_parse\_naptr\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

AVAILABILITY

This function was first introduced in c-ares version 1.7.6.

SEE ALSO

[ares\_query](https://c-ares.haxx.se/ares_query.html), [ares\_free\_data](https://c-ares.haxx.se/ares_free_data.html)

AUTHOR

Written by Jakub Hrozek <jhrozek@redhat.com>, on behalf of Red Hat, Inc [http://www.redhat.com](http://www.redhat.com/)

**ares\_parse\_ns\_reply**

NAME

ares\_parse\_ns\_reply - Parse a reply to a DNS query of type NS into a hostent

SYNOPSIS

#include <ares.h>

int ares\_parse\_ns\_reply(const unsigned char \*abuf, int alen, struct hostent \*\*host);

DESCRIPTION

The ares\_parse\_ns\_reply function parses the response to a query of type NS into a struct hostent . The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by host . The nameservers are stored into the aliases field of the host structure. It is the caller's responsibility to free the resulting host structure using [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html) when it is no longer needed.

RETURN VALUES

ares\_parse\_ns\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

SEE ALSO

[ares\_query (3)](https://c-ares.haxx.se/ares_query.html) [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html)

AUTHOR

Written by Vlad Dinulescu <vlad.dinulescu@avira.com>, on behalf of AVIRA Gmbh [http://www.avira.com](http://www.avira.com/)

**ares\_parse\_ptr\_reply**

NAME

ares\_parse\_ptr\_reply - Parse a reply to a DNS query of type PTR into a hostent

SYNOPSIS

#include <ares.h>

int ares\_parse\_ptr\_reply(const unsigned char \*abuf, int alen, const void \*addr, int addrlen, int family, struct hostent \*\*host);

DESCRIPTION

The ares\_parse\_ptr\_reply function parses the response to a query of type PTR into a struct hostent . The parameters abuf and alen give the contents of the response. The parameters addr , addrlen , and family specify which address was queried for; they are not used to verify the response, merely used to fill in the address of the struct hostent . The resulting struct hostent is stored in allocated memory and a pointer to it stored into the variable pointed to by host . It is the caller's responsibility to free the resulting host structure using [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html) when it is no longer needed.

RETURN VALUES

ares\_parse\_ptr\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

SEE ALSO

[ares\_gethostbyaddr (3)](https://c-ares.haxx.se/ares_gethostbyaddr.html) [ares\_free\_hostent (3)](https://c-ares.haxx.se/ares_free_hostent.html)

AUTHOR

Greg Hudson, MIT Information Systems

Copyright 1998 by the Massachusetts Institute of Technology.

**ares\_parse\_soa\_reply**

NAME

ares\_parse\_soa\_reply - Parse a reply to a DNS query of type SOA

SYNOPSIS

#include <ares.h>

int ares\_parse\_soa\_reply(const unsigned char\* abuf, int alen, struct ares\_soa\_reply\*\* soa\_out);

DESCRIPTION

The ares\_parse\_soa\_reply function parses the response to a query of type SOA into a struct\ ares\_soa\_reply . The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by soa\_out . It is the caller's responsibility to free the resulting soa\_out structure when it is no longer needed using the function ares\_free\_data

The structure ares\_soa\_reply contains the following fields:

struct ares\_soa\_reply { char \*nsname; char \*hostmaster; unsigned int serial; unsigned int refresh; unsigned int retry; unsigned int expire; unsigned int minttl; };

RETURN VALUES

ares\_parse\_soa\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

AVAILABILITY

This function was first introduced in c-ares version 1.9.0.

SEE ALSO

[ares\_query (3)](https://c-ares.haxx.se/ares_query.html) [ares\_free\_data (3)](https://c-ares.haxx.se/ares_free_data.html)

**ares\_parse\_srv\_reply**

NAME

ares\_parse\_srv\_reply - Parse a reply to a DNS query of type SRV

SYNOPSIS

#include <ares.h>

int ares\_parse\_srv\_reply(const unsigned char\* abuf, int alen, struct ares\_srv\_reply\*\* srv\_out);

DESCRIPTION

The ares\_parse\_srv\_reply function parses the response to a query of type SRV into a linked list of struct ares\_srv\_reply The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by srv\_out . It is the caller's responsibility to free the resulting srv\_out structure when it is no longer needed using the function ares\_free\_data

The structure ares\_srv\_reply contains the following fields:

struct ares\_srv\_reply { struct ares\_srv\_reply \*next; unsigned short weight; unsigned short priority; unsigned short port; char \*host; };

RETURN VALUES

ares\_parse\_srv\_reply can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

AVAILABILITY

This function was first introduced in c-ares version 1.7.0.

SEE ALSO

[ares\_query (3)](https://c-ares.haxx.se/ares_query.html) [ares\_free\_data (3)](https://c-ares.haxx.se/ares_free_data.html)

AUTHOR

Written by Jakub Hrozek <jhrozek@redhat.com>, on behalf of Red Hat, Inc [http://www.redhat.com](http://www.redhat.com/)

**ares\_parse\_txt\_reply**

NAME

ares\_parse\_txt\_reply - Parse a reply to a DNS query of type TXT

SYNOPSIS

#include <ares.h>

int ares\_parse\_txt\_reply(const unsigned char\* abuf, int alen, struct ares\_txt\_reply \*\*txt\_out);

int ares\_parse\_txt\_reply\_ext(const unsigned char\* abuf, int alen, struct ares\_txt\_ext \*\*txt\_out);

DESCRIPTION

The ares\_parse\_txt\_reply ( ares\_parse\_txt\_reply\_ext )function parses the response to a query of type TXT into a linked list (one element per sub-string) of struct ares\_txt\_reply ( struct ares\_txt\_ext ) The parameters abuf and alen give the contents of the response. The result is stored in allocated memory and a pointer to it stored into the variable pointed to by txt\_out . It is the caller's responsibility to free the resulting txt\_out structure when it is no longer needed using the function ares\_free\_data

The structure ares\_txt\_reply contains the following fields:

struct ares\_txt\_reply {

struct ares\_txt\_reply \*next;

unsigned int length;

unsigned char \*txt;

};

The structure ares\_txt\_ext contains the following fields:

struct ares\_txt\_ext {

struct ares\_txt\_ext \*next;

unsigned int length;

unsigned char \*txt;

unsigned char record\_start;

};

The record\_start field in struct ares\_txt\_ext is 1 if this structure is a start of a TXT record, and 0 if the structure is a continuation of a previous record. The linked list of the struct ares\_txt\_ext will have at least one item with record\_start equal to 1, and may have some items with record\_start equal to 0 between them.

These sequences of struct ares\_txt\_ext (starting from the item with record\_start equal to 1, and ending right before the record start item) may be treated as either components of a single TXT record or as a multi-parted TXT record, depending on particular use case.

RETURN VALUES

ares\_parse\_txt\_reply ( ares\_parse\_txt\_reply\_ext )can return any of the following values:

ARES\_SUCCESS The response was successfully parsed.

ARES\_EBADRESP The response was malformatted.

ARES\_ENODATA The response did not contain an answer to the query.

ARES\_ENOMEM Memory was exhausted.

AVAILABILITY

This function was first introduced in c-ares version 1.7.0.

SEE ALSO

[ares\_query](https://c-ares.haxx.se/ares_query.html), [ares\_free\_data](https://c-ares.haxx.se/ares_free_data.html)

AUTHOR

Written by Jakub Hrozek <jhrozek@redhat.com>, on behalf of Red Hat, Inc [http://www.redhat.com](http://www.redhat.com/)

**ares\_process**

**ares\_process\_fd**

NAME

ares\_process - Process events for name resolution

SYNOPSIS

#include <ares.h>

void ares\_process(ares\_channel channel, fd\_set \*read\_fds,

fd\_set \*write\_fds)

void ares\_process\_fd(ares\_channel channel, ares\_socket\_t read\_fd,

ares\_socket\_t write\_fd)

DESCRIPTION

The [ares\_process](https://c-ares.haxx.se/ares_process.html) function handles input/output events and timeouts associated with queries pending on the name service channel identified by channel . The file descriptor sets pointed to by read\_fds and write\_fds should have file descriptors set in them according to whether the file descriptors specified by [ares\_fds](https://c-ares.haxx.se/ares_fds.html) are ready for reading and writing. (The easiest way to determine this information is to invoke select(3) with a timeout no greater than the timeout given by [ares\_timeout)](https://c-ares.haxx.se/ares_timeout.html).

The [ares\_process](https://c-ares.haxx.se/ares_process.html) function will invoke callbacks for pending queries if they complete successfully or fail.

ares\_process\_fd(3) works the same way but acts and operates only on the specific file descriptors (sockets) you pass in to the function. Use ARES\_SOCKET\_BAD for "no action". This function is provided to allow users of c-ares to void select(3) in their applications and within c-ares.

To only process possible timeout conditions without a socket event occurring, one may pass NULL as the values for both read\_fds and write\_fds for [ares\_process](https://c-ares.haxx.se/ares_process.html), or ARES\_SOCKET\_BAD for both read\_fd and write\_fd for ares\_process\_fd(3).

EXAMPLE

The following code fragment waits for all pending queries on a channel to complete:

int nfds, count; fd\_set readers, writers; struct timeval tv, \*tvp;

while (1) {

FD\_ZERO(&readers); FD\_ZERO(&writers);

nfds = ares\_fds(channel, &readers, &writers);

if (nfds == 0)

break;

tvp = ares\_timeout(channel, NULL, &tv);

count = select(nfds, &readers, &writers, NULL, tvp);

ares\_process(channel, &readers, &writers); }

SEE ALSO

[ares\_fds,](https://c-ares.haxx.se/ares_fds.html) [ares\_timeout](https://c-ares.haxx.se/ares_timeout.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_query**

NAME

ares\_query - Initiate a single-question DNS query

SYNOPSIS

#include <ares.h>

typedef void (\*ares\_callback)(void \*arg, int status, int timeouts, unsigned char \*abuf, int alen)

void ares\_query(ares\_channel channel, const char \*name, int dnsclass, int type, ares\_callback callback, void \*arg)

DESCRIPTION

The ares\_query function initiates a single-question DNS query on the name service channel identified by channel . The parameter name gives the query name as a NULterminated C string of period-separated labels optionally ending with a period; periods and backslashes within a label must be escaped with a backslash. The parameters dnsclass and type give the class and type of the query using the values defined in <arpa/nameser.h> . When the query is complete or has failed, the ares library will invoke callback . Completion or failure of the query may happen immediately, or may happen during a later call to [ares\_process (3)](https://c-ares.haxx.se/ares_process.html) or ares\_destroy (3).

The callback argument arg is copied from the ares\_query argument arg . The callback argument status indicates whether the query succeeded and, if not, how it failed. It may have any of the following values:

ARES\_SUCCESS The query completed successfully.

ARES\_ENODATA The query completed but contains no answers.

ARES\_EFORMERR The query completed but the server claims that the query was malformatted.

ARES\_ESERVFAIL The query completed but the server claims to have experienced a failure. (This code can only occur if the ARES\_FLAG\_NOCHECKRESP flag was specified at channel initialization time; otherwise, such responses are ignored at the [ares\_send (3)](https://c-ares.haxx.se/ares_send.html) level.)

ARES\_ENOTFOUND The query completed but the queried-for domain name was not found.

ARES\_ENOTIMP The query completed but the server does not implement the operation requested by the query. (This code can only occur if the ARES\_FLAG\_NOCHECKRESP flag was specified at channel initialization time; otherwise, such responses are ignored at the [ares\_send (3)](https://c-ares.haxx.se/ares_send.html) level.)

ARES\_EREFUSED The query completed but the server refused the query. (This code can only occur if the ARES\_FLAG\_NOCHECKRESP flag was specified at channel initialization time; otherwise, such responses are ignored at the [ares\_send (3)](https://c-ares.haxx.se/ares_send.html) level.)

ARES\_EBADNAME The query name name could not be encoded as a domain name, either because it contained a zero-length label or because it contained a label of more than 63 characters.

ARES\_ETIMEOUT No name servers responded within the timeout period.

ARES\_ECONNREFUSED No name servers could be contacted.

ARES\_ENOMEM Memory was exhausted.

ARES\_ECANCELLED The query was cancelled.

ARES\_EDESTRUCTION The name service channel channel is being destroyed; the query will not be completed.

The callback argument timeouts reports how many times a query timed out during the execution of the given request.

If the query completed (even if there was something wrong with it, as indicated by some of the above error codes), the callback argument abuf points to a result buffer of length alen . If the query did not complete, abuf will be NULL and alen will be 0.

SEE ALSO

[ares\_process (3)](https://c-ares.haxx.se/ares_process.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_save\_options**

NAME

ares\_save\_options - Save configuration values obtained from initialized ares\_channel

SYNOPSIS

#include <ares.h> int ares\_save\_options(ares\_channel channel, struct ares\_options \*options, int \*optmask)

DESCRIPTION

The [ares\_save\_options](https://c-ares.haxx.se/ares_save_options.html) function saves the channel data identified by channel , into the options struct identified by options , and saves the mask of options which are set to the integer pointer (passed by reference) identified by optmask .

The resultant options and optmask are then able to be passed directly to

ares\_init\_options. When the options are no longer needed, ares\_destroy\_options should be called to free any associated memory.

RETURN VALUES

[ares\_save\_options](https://c-ares.haxx.se/ares_save_options.html) can return any of the following values:

ARES\_SUCCESS The channel data was successfully stored

ARES\_ENOMEM The memory was exhausted

ARES\_ENODATA The channel data identified by channel were invalid.

NOTE

Since c-ares 1.6.0 the ares\_options struct has been "locked" meaning that it won't be extended to cover new functions. This function will remain functioning, but it can only return config data that can be represented in this config struct, which may no longer be the complete set of config options. [ares\_dup](https://c-ares.haxx.se/ares_dup.html) will not have that restriction.

The ares\_options struct can not handle potential IPv6 name servers the ares\_channel might be configured to use. The [ares\_save\_options](https://c-ares.haxx.se/ares_save_options.html) function will only return IPv4 servers, if any. In order to retrieve all name servers an ares\_channel might be using, the [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html) function must be used instead.

SEE ALSO

[ares\_destroy\_options](https://c-ares.haxx.se/ares_destroy_options.html), [ares\_init\_options,](https://c-ares.haxx.se/ares_init_options.html) [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html), [ares\_dup](https://c-ares.haxx.se/ares_dup.html)

AVAILABILITY

ares\_save\_options(3) was added in c-ares 1.4.0

AUTHOR

Brad House

**ares\_search**

NAME

ares\_search - Initiate a DNS query with domain search

SYNOPSIS

#include <ares.h>

typedef void (\*ares\_callback)(void \*arg, int status, int timeouts, unsigned char \*abuf, int alen)

void ares\_search(ares\_channel channel, const char \*name, int dnsclass, int type, ares\_callback callback, void \*arg)

DESCRIPTION

The ares\_search function initiates a series of single-question DNS queries on the name service channel identified by channel , using the channel's search domains as well as a host alias file given by the HOSTALIAS environment variable. The parameter name gives the alias name or the base of the query name as a NUL-terminated C string of periodseparated labels; if it ends with a period, the channel's search domains will not be used. Periods and backslashes within a label must be escaped with a backslash. The parameters dnsclass and type give the class and type of the query using the values defined in <arpa/nameser.h> . When the query sequence is complete or has failed, the ares library will invoke callback . Completion or failure of the query sequence may happen immediately, or may happen during a later call to [ares\_process (3)](https://c-ares.haxx.se/ares_process.html) or ares\_destroy (3).

The callback argument arg is copied from the ares\_search argument arg . The callback argument status indicates whether the query sequence ended with a successful query and, if not, how the query sequence failed. It may have any of the following values:

ARES\_SUCCESS A query completed successfully.

ARES\_ENODATA No query completed successfully; when the query was tried without a search domain appended, a response was returned with no answers.

ARES\_EFORMERR A query completed but the server claimed that the query was malformatted.

ARES\_ESERVFAIL No query completed successfully; when the query was tried without a

search domain appended, the server claimed to have experienced a failure. (This code can only occur if the ARES\_FLAG\_NOCHECKRESP flag was specified at channel initialization time; otherwise, such responses are ignored at the [ares\_send (3)](https://c-ares.haxx.se/ares_send.html) level.)

ARES\_ENOTFOUND No query completed successfully; when the query was tried without a search domain appended, the server reported that the queried-for domain name was not found.

ARES\_ENOTIMP A query completed but the server does not implement the operation requested by the query. (This code can only occur if the ARES\_FLAG\_NOCHECKRESP flag was specified at channel initialization time; otherwise, such responses are ignored at the [ares\_send (3)](https://c-ares.haxx.se/ares_send.html) level.)

ARES\_EREFUSED A query completed but the server refused the query. (This code can only occur returned if the ARES\_FLAG\_NOCHECKRESP flag was specified at channel initialization time; otherwise, such responses are ignored at the [ares\_send (3)](https://c-ares.haxx.se/ares_send.html) level.) ARES\_TIMEOUT No name servers responded to a query within the timeout period.

ARES\_ECONNREFUSED No name servers could be contacted.

ARES\_ENOMEM Memory was exhausted.

ARES\_ECANCELLED The query was cancelled.

ARES\_EDESTRUCTION The name service channel channel is being destroyed; the query will not be completed.

The callback argument timeouts reports how many times a query timed out during the execution of the given request.

If a query completed successfully, the callback argument abuf points to a result buffer of length alen . If the query did not complete successfully, abuf will usually be NULL and alen will usually be 0, but in some cases an unsuccessful query result may be placed in abuf .

SEE ALSO

[ares\_process (3)](https://c-ares.haxx.se/ares_process.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_send**

NAME

ares\_send - Initiate a DNS query

SYNOPSIS

#include <ares.h>

typedef void (\*ares\_callback)(void \*arg, int status, int timeouts, unsigned char \*abuf, int alen)

void ares\_send(ares\_channel channel, const unsigned char \*qbuf, int qlen, ares\_callback callback, void \*arg)

DESCRIPTION

The ares\_send function initiates a DNS query on the name service channel identified by channel . The parameters qbuf and qlen give the DNS query, which should already have been formatted according to the DNS protocol. When the query is complete or has failed, the ares library will invoke callback . Completion or failure of the query may happen immediately, or may happen during a later call to [ares\_process (3)](https://c-ares.haxx.se/ares_process.html) or ares\_destroy (3).

The callback argument arg is copied from the ares\_send argument arg . The callback argument status indicates whether the query succeeded and, if not, how it failed. It may have any of the following values:

ARES\_SUCCESS The query completed.

ARES\_EBADQUERY The query buffer was poorly formed (was not long enough for a DNS header or was too long for TCP transmission).

ARES\_ETIMEOUT No name servers responded within the timeout period.

ARES\_ECONNREFUSED No name servers could be contacted.

ARES\_ENOMEM Memory was exhausted.

ARES\_ECANCELLED The query was cancelled.

ARES\_EDESTRUCTION The name service channel channel is being destroyed; the query will not be completed.

The callback argument timeouts reports how many times a query timed out during the execution of the given request.

If the query completed, the callback argument abuf points to a result buffer of length alen . If the query did not complete, abuf will be NULL and alen will be 0.

Unless the flag ARES\_FLAG\_NOCHECKRESP was set at channel initialization time, ares\_send will normally ignore responses whose questions do not match the questions in qbuf , as well as responses with reply codes of SERVFAIL NOTIMP and REFUSED . Unlike other query functions in the ares library, however, ares\_send does not inspect the header of the reply packet to determine the error status, so a callback status of ARES\_SUCCESS does not reflect as much about the response as for other query functions.

SEE ALSO

[ares\_process (3)](https://c-ares.haxx.se/ares_process.html)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_set\_local\_dev**

NAME

ares\_set\_local\_dev - Bind to a specific network device when creating sockets.

SYNOPSIS

#include <ares.h> void ares\_set\_local\_dev(ares\_channel channel, const char\* local\_dev\_name)

DESCRIPTION

The ares\_set\_local\_dev function causes all future sockets to be bound to this device with SO\_BINDTODEVICE. This forces communications to go over a certain interface, which can be useful on multi-homed machines. This option is only supported on Linux, and root privileges are required for the option to work. If SO\_BINDTODEVICE is not supported or the setsocktop call fails (probably because of permissions), the error is silently ignored.

SEE ALSO

[ares\_set\_local\_ip4,](https://c-ares.haxx.se/ares_set_local_ip4.html) [ares\_set\_local\_ip6](https://c-ares.haxx.se/ares_set_local_ip6.html)

NOTES

This function was added in c-ares 1.7.4

AUTHOR

Ben Greear

**ares\_set\_servers**

NAME

ares\_set\_servers, ares\_set\_servers\_ports - Initialize an ares\_channel name servers configuration

SYNOPSIS

#include <ares.h>

int ares\_set\_servers(ares\_channel channel, struct ares\_addr\_node \*servers)

int ares\_set\_servers\_ports(ares\_channel channel, struct ares\_addr\_port\_node \*servers)

DESCRIPTION

The [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) function initializes name servers configuration for the channel data identified by channel , from a servers pointer to a linked list of ares\_addr\_node structs holding name servers address data.

The name server linked list pointer argument may be the result of a previous call to [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html) or a linked list of ares\_addr\_node structs set up by other means.

The [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) function also allows the specification of UDP and TCP ports to be used for communication on a per-server basis. The provided linked list argument may be the result of a previous call to [ares\_get\_servers\_ports](https://c-ares.haxx.se/ares_get_servers_ports.html) or a linked list of ares\_addr\_port\_node structs set up by other means.

This function replaces any potentially previously configured name servers with the ones given in the linked list. So, in order to configure a channel with more than one name server all the desired ones must be specified in a single list.

The function does not take ownership of the linked list argument. The caller is responsible for freeing the linked list when no longer needed.

This function is capable of handling IPv4 and IPv6 name server addresses simultaneously, rendering [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) with optmask ARES\_OPT\_SERVERS functionally obsolete except for IPv4-only name server usage.

RETURN VALUES

[ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) may return any of the following values:

ARES\_SUCCESS The name servers configuration was successfully initialized.

ARES\_ENOMEM The process's available memory was exhausted.

ARES\_ENODATA The channel data identified by channel was invalid.

ARES\_ENOTINITIALIZED c-ares library initialization not yet performed.

ARES\_ENOTIMP Changing name servers configuration while queries are outstanding is not implemented.

SEE ALSO

[ares\_set\_servers\_csv,](https://c-ares.haxx.se/ares_set_servers_csv.html) [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html), [ares\_init\_options,](https://c-ares.haxx.se/ares_init_options.html) [ares\_dup](https://c-ares.haxx.se/ares_dup.html)

AVAILABILITY

[ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) was added in c-ares 1.7.1; [ares\_set\_servers\_ports](https://c-ares.haxx.se/ares_set_servers_ports.html) was added in c-ares 1.11.0.

AUTHOR

Implementation of this function and associated library internals are based on code, comments and feedback provided in November and December of 2008 by Daniel Stenberg, Gregor Jasny, Phil Blundell and Yang Tse, December 2009 by Cedric Bail, February 2010 by Jakub Hrozek. On March 2010 Yang Tse shuffled all the bits and this function popped out.

**ares\_set\_servers\_csv**

NAME

ares\_set\_servers, ares\_set\_servers\_ports - Initialize an ares\_channel name servers configuration

SYNOPSIS

#include <ares.h>

int ares\_set\_servers(ares\_channel channel, struct ares\_addr\_node \*servers) int ares\_set\_servers\_ports(ares\_channel channel, struct ares\_addr\_port\_node \*servers)

DESCRIPTION

The [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) function initializes name servers configuration for the channel data identified by channel , from a servers pointer to a linked list of ares\_addr\_node structs holding name servers address data.

The name server linked list pointer argument may be the result of a previous call to [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html) or a linked list of ares\_addr\_node structs set up by other means.

The [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) function also allows the specification of UDP and TCP ports to be used for communication on a per-server basis. The provided linked list argument may be the result of a previous call to [ares\_get\_servers\_ports](https://c-ares.haxx.se/ares_get_servers_ports.html) or a linked list of ares\_addr\_port\_node structs set up by other means.

This function replaces any potentially previously configured name servers with the ones given in the linked list. So, in order to configure a channel with more than one name server all the desired ones must be specified in a single list.

The function does not take ownership of the linked list argument. The caller is responsible for freeing the linked list when no longer needed.

This function is capable of handling IPv4 and IPv6 name server addresses simultaneously, rendering [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) with optmask ARES\_OPT\_SERVERS functionally obsolete except for IPv4-only name server usage.

RETURN VALUES

[ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) may return any of the following values:

ARES\_SUCCESS The name servers configuration was successfully initialized.

ARES\_ENOMEM The process's available memory was exhausted.

ARES\_ENODATA The channel data identified by channel was invalid.

ARES\_ENOTINITIALIZED c-ares library initialization not yet performed.

ARES\_ENOTIMP Changing name servers configuration while queries are outstanding is not implemented.

SEE ALSO

[ares\_set\_servers\_csv,](https://c-ares.haxx.se/ares_set_servers_csv.html) [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html), [ares\_init\_options,](https://c-ares.haxx.se/ares_init_options.html) [ares\_dup](https://c-ares.haxx.se/ares_dup.html)

AVAILABILITY

[ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) was added in c-ares 1.7.1; [ares\_set\_servers\_ports](https://c-ares.haxx.se/ares_set_servers_ports.html) was added in c-ares 1.11.0.

AUTHOR

Implementation of this function and associated library internals are based on code, comments and feedback provided in November and December of 2008 by Daniel Stenberg, Gregor Jasny, Phil Blundell and Yang Tse, December 2009 by Cedric Bail, February 2010 by Jakub Hrozek. On March 2010 Yang Tse shuffled all the bits and this function popped out.

Copyright 1998 by the Massachusetts Institute of Technology.

**ares\_set\_servers\_ports**

NAME

ares\_set\_servers, ares\_set\_servers\_ports - Initialize an ares\_channel name servers configuration

SYNOPSIS

#include <ares.h>

int ares\_set\_servers(ares\_channel channel, struct ares\_addr\_node \*servers) int ares\_set\_servers\_ports(ares\_channel channel, struct ares\_addr\_port\_node \*servers)

DESCRIPTION

The [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) function initializes name servers configuration for the channel data identified by channel , from a servers pointer to a linked list of ares\_addr\_node structs holding name servers address data.

The name server linked list pointer argument may be the result of a previous call to [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html) or a linked list of ares\_addr\_node structs set up by other means.

The [ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) function also allows the specification of UDP and TCP ports to be used for communication on a per-server basis. The provided linked list argument may be the result of a previous call to [ares\_get\_servers\_ports](https://c-ares.haxx.se/ares_get_servers_ports.html) or a linked list of ares\_addr\_port\_node structs set up by other means.

This function replaces any potentially previously configured name servers with the ones given in the linked list. So, in order to configure a channel with more than one name server all the desired ones must be specified in a single list.

The function does not take ownership of the linked list argument. The caller is responsible for freeing the linked list when no longer needed.

This function is capable of handling IPv4 and IPv6 name server addresses simultaneously, rendering [ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html) with optmask ARES\_OPT\_SERVERS functionally obsolete except for IPv4-only name server usage.

RETURN VALUES

[ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) may return any of the following values:

ARES\_SUCCESS The name servers configuration was successfully initialized.

ARES\_ENOMEM The process's available memory was exhausted.

ARES\_ENODATA The channel data identified by channel was invalid.

ARES\_ENOTINITIALIZED c-ares library initialization not yet performed.

ARES\_ENOTIMP Changing name servers configuration while queries are outstanding is not implemented.

SEE ALSO

[ares\_set\_servers\_csv,](https://c-ares.haxx.se/ares_set_servers_csv.html) [ares\_get\_servers](https://c-ares.haxx.se/ares_get_servers.html), [ares\_init\_options,](https://c-ares.haxx.se/ares_init_options.html) [ares\_dup](https://c-ares.haxx.se/ares_dup.html)

AVAILABILITY

[ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html) was added in c-ares 1.7.1; [ares\_set\_servers\_ports](https://c-ares.haxx.se/ares_set_servers_ports.html) was added in c-ares 1.11.0.

AUTHOR

Implementation of this function and associated library internals are based on code, comments and feedback provided in November and December of 2008 by Daniel Stenberg, Gregor Jasny, Phil Blundell and Yang Tse, December 2009 by Cedric Bail, February 2010 by Jakub Hrozek. On March 2010 Yang Tse shuffled all the bits and this function popped out.

**ares\_set\_servers\_ports\_csv**

NAME

ares\_set\_servers\_csv, ares\_set\_servers\_ports\_csv - Set list of DNS servers to be used.

SYNOPSIS

#include <ares.h>

int ares\_set\_servers\_csv(ares\_channel channel, const char\* servers) int ares\_set\_servers\_ports\_csv(ares\_channel channel, const char\* servers)

DESCRIPTION

The ares\_set\_servers\_csv and ares\_set\_servers\_ports\_csvfunctions set the list of DNS servers that ARES will query. The format of the servers option is:

host[:port][,host[:port]]...

For example:

192.168.1.100,192.168.1.101,3.4.5.6

The ares\_set\_servers\_csv function will ignore any port values specified in the input string, whereare the ares\_set\_servers\_ports\_csv function will apply any specified port values as the UDP and TCP port to be used for that particular nameserver.

RETURN VALUES

[ares\_set\_servers\_csv](https://c-ares.haxx.se/ares_set_servers_csv.html) This function may return any of the following values:

ARES\_SUCCESS The name servers configuration was successfully initialized.

ARES\_ENOMEM The process's available memory was exhausted.

ARES\_ENODATA The channel data identified by channel was invalid.

ARES\_ENOTINITIALIZED c-ares library initialization not yet performed.

ARES\_ENOTIMP Changing name servers configuration while queries are outstanding is not implemented.

SEE ALSO

[ares\_set\_servers](https://c-ares.haxx.se/ares_set_servers.html)

AVAILABILITY

ares\_set\_servers\_csv was added in c-ares 1.7.2; ares\_set\_servers\_ports\_csv was added in c-ares 1.11.0.

AUTHOR

Ben Greear

**ares\_set\_socket\_callback**

NAME

ares\_set\_socket\_callback - Set a socket creation callback

SYNOPSIS

#include <ares.h>

typedef int (\*ares\_sock\_create\_callback)(ares\_socket\_t socket\_fd,

int type, void \*userdata) void ares\_set\_socket\_callback(ares\_channel channel,

ares\_sock\_create\_callback callback,

void \*userdata) cc file.c -lcares

DESCRIPTION

This function sets a callback in the given ares channel handle. This callback function will be invoked after the socket has been created, and connected to the remote server. The callback must return ARES\_SUCCESS if things are fine, or return -1 to signal an error. A returned error will abort the ares operation.

SEE ALSO

[ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html), [ares\_set\_socket\_configure\_callback](https://c-ares.haxx.se/ares_set_socket_configure_callback.html)

AVAILABILITY

ares\_set\_socket\_callback(3) was added in c-ares 1.6.0

AUTHOR

Gregor Jasny

**ares\_set\_socket\_configure\_callback**

NAME

ares\_set\_socket\_configure\_callback - Set a socket configuration callback

SYNOPSIS

#include <ares.h>

typedef int (\*ares\_sock\_config\_callback)(ares\_socket\_t socket\_fd,

int type, void \*userdata) void ares\_set\_socket\_configure\_callback(ares\_channel channel, ares\_sock\_config\_callback callback,

void \*userdata) cc file.c -lcares

DESCRIPTION

This function sets a callback in the given ares channel handle. This callback function will be invoked after the socket has been created, but before it has been connected to the remote server, which is an ideal time to configure various socket options. The callback must return ARES\_SUCCESS if things are fine, or return -1 to signal an error. A returned error will abort the ares operation.

SEE ALSO

[ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html), [ares\_set\_socket\_callback](https://c-ares.haxx.se/ares_set_socket_callback.html)

AVAILABILITY

ares\_set\_socket\_configure\_callback(3) was added in c-ares 1.11.0

AUTHOR

Andrew Ayer

**ares\_set\_socket\_functions**

NAME

ares\_set\_socket\_functions - Set socket io callbacks

SYNOPSIS

#include <ares.h>

struct ares\_socket\_functions {

ares\_socket\_t(\*asocket)(int, int, int, void \*); int(\*aclose)(ares\_socket\_t, void \*);

int(\*aconnect)(ares\_socket\_t, const struct sockaddr \*, ares\_socklen\_t, void \*);

ares\_ssize\_t(\*arecvfrom)(ares\_socket\_t, void \*, size\_t, int, struct sockaddr \*, ares\_socklen\_t \*, void \*); ares\_ssize\_t(\*asendv)(ares\_socket\_t, const struct iovec \*, int, void \*); };

void ares\_set\_socket\_functions(ares\_channel channel,

const struct ares\_socket\_functions \* functions,

void \*user\_data);

DESCRIPTION

This function sets a set of callback functions in the given ares channel handle. These callback functions will be invoked to create/destroy socket objects and perform io, instead of the normal system calls. A client application can override normal network operation fully through this functionality, and provide its own transport layer.

All callback functions are expected to operate like their system equivalents, and to set errno(3)to an appropriate error code on failure. C-ares also expects all io functions to behave asynchronously, i.e. as if the socket object has been set to non-blocking mode. Thus read/write calls (for TCP connections) are expected to often generate EAGAINor EWOULDBLOCK.

The user\_data value is provided to each callback function invocation to serve as context.

The ares\_socket\_functions must provide the following callbacks:

asocket ares\_socket\_t(\*)(int domain, int type, int protocol, void \* user\_data)

Creates an endpoint for communication and returns a descriptor. domain, type, and protocol each correspond to the parameters of socket(2).Returns ahandle to the newly created socket, or -1 on error.

aclose int(\*)(ares\_socket\_t fd, void \* user\_data) Closes the socket endpoint indicated by fd. See close(2)

aconnect int(\*)(ares\_socket\_t fd, const struct sockaddr \* addr, ares\_socklen\_t addr\_len, void \* user\_data)

Initiate a connection to the address indicated by addr on a socket. See connect(2)

arecvfrom ares\_ssize\_t(\*)(ares\_socket\_t fd, void \* buffer, size\_t buf\_size, int flags, struct sockaddr \* addr, ares\_socklen\_t \* addr\_len, void \* user\_data)

Receives data from remote socket endpoint, if available. If the addr parameter is not NULL and

the connection protocol provides the source address, the callback should fill this in. See recvfrom(2)

asendv ares\_ssize\_t(\*)(ares\_socket\_t fd, const struct iovec \* data, int len, void \* user\_data) Send data, as provided by the iovec array data, to the socket endpoint. See writev(2)

The ares\_socket\_functions struct provided is not copied but directly referenced, and must thus remain valid through out the channels and any created socket's lifetime.

AVAILABILITY

Added in c-ares 1.13.0

SEE ALSO

[ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html), socket(2), close(2), connect(2), recv(2), recvfrom(2), send(2), writev(2)

AUTHOR

Carl Wilund

**ares\_set\_sortlist**

NAME

ares\_set\_sortlist - Initialize an ares\_channel sortlist configuration

SYNOPSIS

#include <ares.h> int ares\_set\_sortlist(ares\_channel channel, const char \*sortstr)

DESCRIPTION

The [ares\_set\_sortlist](https://c-ares.haxx.se/ares_set_sortlist.html) function initializes an address sortlist configuration for the channel data identified by channel , so that addresses returned by [ares\_gethostbyname](https://c-ares.haxx.se/ares_gethostbyname.html) are sorted according to the sortlist. The provided sortstr string that holds a space separated list of IP-address-netmask pairs. The netmask is optional but follows the address after a slash if present. For example, "130.155.160.0/255.255.240.0 130.155.0.0".

This function replaces any potentially previously configured address sortlist with the ones given in the configuration string.

RETURN VALUES

[ares\_set\_sortlist](https://c-ares.haxx.se/ares_set_sortlist.html) may return any of the following values:

ARES\_SUCCESS The sortlist configuration was successfully initialized.

ARES\_ENOMEM The process's available memory was exhausted.

ARES\_ENODATA The channel data identified by channel was invalid.

ARES\_ENOTINITIALIZED c-ares library initialization not yet performed.

SEE ALSO

[ares\_init\_options](https://c-ares.haxx.se/ares_init_options.html), [ares\_dup](https://c-ares.haxx.se/ares_dup.html)

AVAILABILITY

ares\_set\_sortlist(3) was added in c-ares 1.11.0

**ares\_strerror**

NAME

ares\_strerror - Get the description of an ares library error code

SYNOPSIS

#include <ares.h>

const char \*ares\_strerror(int code)

DESCRIPTION

The ares\_strerror function gets the description of the ares library error code code , returning the result as a NUL-terminated C string.

NOTES

This function is not compatible with ares, it takes a different set of arguments.

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_timeout**

NAME

ares\_timeout - return maximum time to wait

SYNOPSIS

#include <ares.h>

struct timeval \*ares\_timeout(ares\_channel channel, struct timeval \*maxtv,

struct timeval \*tv)

DESCRIPTION

The [ares\_timeout](https://c-ares.haxx.se/ares_timeout.html) function determines the maximum time for which the caller should wait before invoking [ares\_process](https://c-ares.haxx.se/ares_process.html) to process timeouts. The parameter maxtv specifies a existing maximum timeout, or NULL if the caller does not wish to apply a maximum timeout. The parameter tv must point to a writable buffer of type struct timeval It is valid for maxtv and tv to have the same value.

If no queries have timeouts pending sooner than the given maximum timeout, [ares\_timeout](https://c-ares.haxx.se/ares_timeout.html) returns the value of maxtv; otherwise [ares\_timeout](https://c-ares.haxx.se/ares_timeout.html) stores the appropriate timeout value into the buffer pointed to by tv and returns the value of tv.

SEE ALSO

[ares\_fds,](https://c-ares.haxx.se/ares_fds.html) [ares\_process](https://c-ares.haxx.se/ares_process.html), ares\_process\_fd (3)

AUTHOR

Greg Hudson, MIT Information Systems

**ares\_version**

NAME

ares\_version - Get the version number of the library

SYNOPSIS

#include <ares.h>

const char \*ares\_version(int \*version)

DESCRIPTION

The [ares\_version](https://c-ares.haxx.se/ares_version.html) function gets the library version as a string and optionally as an integer stored in the version argument. If you pass a NULL, no integer is attempted to be returned.

The integer is built up as 24bit number, with 8 separate bits used for major number, minor number and patch number. This makes a version string such as 1.2.3 will be returned as the hexadecimal number 0x010203 (decimal 66051).

SEE ALSO

[ares\_init,](https://c-ares.haxx.se/ares_init.html) [ares\_library\_init](https://c-ares.haxx.se/ares_library_init.html)