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;
};

int ares_init_options(ares_channel *channelptr,
                      struct ares_options *options,
                      int optmask)
```

DESCRIPTION

The **ares_init_options(3)** function initializes a communications channel for name service lookups. If it returns successfully, **ares_init_options(3)** will set the variable pointed to by *channelptr* to a handle used to identify the name service channel. The caller should invoke *ares_destroy(3)* 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_servers(3)** function 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, int s, int read, int write);**
void ***sock_state_cb_data*;**
A callback function to be invoked when a socket changes state. *s* will be passed the socket whose state has changed; *read* will be set to true if the socket should listen for read events, and *write* 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(3)** (because **struct apattern** is opaque); to set a fresh sort list, use **ares_set_sortlist(3)**.

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***;**
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(3) 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_options(3)** 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(3), ares_destroy(3), ares_dup(3), ares_library_init(3), ares_save_options(3), ares_set_servers(3), ares_set_sortlist(3)

AUTHOR

Greg Hudson, MIT Information Systems