NAME

ares_init_options, ares_init − Initialize a resolver channel

SYNOPSIS

```
#include <ares.h>

struct ares_server_failover_options {
  unsigned short retry_chance;
  size_t retry_delay;
};

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;
  char *hosts_path;
  int udp_max_queries;
  int maxtimeout; /* in milliseconds */
  unsigned int qcache_max_ttl; /* in seconds */
  ares_evsys_t evsys;
  struct ares_server_failover_options server_failover_opts;
};

int ares_init_options(ares_channel_t **channelptr,
                      const struct ares_options *options,
                      int optmask);

int ares_init(ares_channel_t **channelptr);
```

DESCRIPTION

The **ares_init(3)** function is equivalent to calling **ares_init_options(channelptr, NULL, 0)**. It is recommended to use **ares_init_options(3)** instead and to set or make configurable the appropriate options for your application.

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.

It is recommended for an application to have at most one ares channel and use this for all DNS queries for the life of the application. When system configuration changes, *ares_reinit(3)* can be called to reload the configuration if necessary. The recommended concurrent query limit is about 32k queries, but remembering that when specifying AF_UNSPEC for **ares_getaddrinfo(3)** or **ares_gethostbyname(3)**, they may spawn 2 queries internally. The reason for the limit is c-ares does not allow duplicate DNS query ids (which have a maximum of 64k) to be oustanding at a given time, and it must randomly search for an available id thus 32k will limit the number of searches. This limitation should not be a concern for most implementations and c-ares may implement queuing in future releases to lift this limitation.

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:

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. As of v1.22, this is on by default if flags are otherwise not set.

ARES_FLAG_NO_DFLT_SVR

Do not attempt to add a default local named server if there are no other servers available. Instead, fail initialization with *ARES_ENOSERVER*.

ARES_FLAG_DNS0x20 Enable support for DNS 0x20 as per https://datatracker.ietf.org/doc/html/draft-vixie-dnsext-dns0x20-00 which adds additional entropy to the request by randomizing the case of the query name. Integrators need to ensure they treat DNS name responses as case-insensitive. In rare circumstances this may cause the inability to lookup certain

domains if the upstream server or the authoritative server for the domain is non-compliant.

ARES_OPT_TIMEOUT

int *timeout***;**
The number of seconds each name server is given to respond to a query on the first try. See*ARES_OPT_TIMEOUTMS* which this v alue is converted into.

ARES_OPT_TIMEOUTMS

int *timeout***;**
The number of milliseconds each name server is given to respond to a query on the first try of any given server. The default is two seconds, however any value below 250ms will automatically be set to 250ms (roughly the RTT half-way around the world). 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.

As of c-ares 1.32.0, this option is only honored on the first successful query to any given server, after that the timeout is automatically calculated based on prior query history.

ARES_OPT_TRIES

int *tries***;**
The number of tries the resolver will try contacting each name server before giving up. The default is three 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. Valid range is 0-15.

ARES_OPT_MAXTIMEOUTMS

int *maxtimeout***;**
The upper bound for timeout between sequential retry attempts. When retrying queries, the timeout is increased from the requested timeout parameter, this caps the value.

ARES_OPT_UDP_PORT

unsigned short *udp_port***;**
The port to use for queries over UDP, in host byte order. The default value is 53, the standard name service port.

ARES_OPT_TCP_PORT

unsigned short *tcp_port***;**
The port to use for queries over TCP, in host byte order. The default value is 53, 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 **0 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. *soc ket_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. The channel lock is held during this callback, if in a multithreaded application, care must be taken to ensure lock order is correct to be able to handle this and avoid deadlocks.

Cannot be used with **ARES_OPT_EVENT_THREAD**.

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 advertised in EDNS; only takes effect if the **ARES_FLAG_EDNS** flag is set. Defaults to 1232, the recommended size.

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**

ARES_OPT_HOSTS_FILE

char ****hosts_path*;**

The path to use for reading the hosts file. The *hosts_path* should be set to a path string, and will be honoured on *nix like systems. The default is **/etc/hosts**

ARES_OPT_UDP_MAX_QUERIES

int *udp_max_queries***;**

The maximum number of udp queries that can be sent on a single ephemeral port to a given DNS server before a new ephemeral port is assigned. Any value of 0 or less will be considered unlimited, and is the default.

ARES_OPT_QUERY_CACHE

unsigned int *qcache_max_ttl***;**
As of c-ares 1.31.0, the query cache is enabled by default with a TTL of 1hr. To disable the query cache, specify this option with a TTL of 0. The query cache is based on the returned TTL in the DNS message. Only fully successful and NXDOMAIN query results will be cached. Fill in the *qcache_max_ttl* with the maximum number of seconds a query result may be cached which will override a larger TTL in the response message. This must be a non-zero value otherwise the cache will be disabled. Choose a reasonable value for your application such as 300 (5 minutes) or 3600 (1 hour). The query cache is automatically flushed if a server configuration change is made.

ARES_OPT_EVENT_THREAD

ares_evsys_t *evsys***;**
Enable the built-in event thread (Recommended). Introduced in c-ares 1.26.0. Set the *evsys* parameter to **ARES_EVSYS_DEFAULT** (0). Other values are reserved for testing and should not be used by integrators.

This option cannot be used with the **ARES_OPT_SOCK_STATE_CB** option, nor the *ares_set_socket_functions(3)* or *ares_set_socket_configure_callback(3)* functions.

When enabled, the integrator is no longer responsible for notifying c-ares of any events on the file descriptors, so *ares_process(3)* nor *ares_process_fd(3)* should ever be called when this option is enabled.

As of c-ares 1.29.0, when enabled, it will also automatically re-load the system configuration when changes are detected.

Use *ares_threadsafety(3)* to determine if this option is available to be used.

Returns **ARES_ENOTIMP** if this option is passed but not available, and **ARES_ESERVFAIL** if there is a critical failure during initialization of the event thread.

ARES_OPT_SERVER_FAILOVER

struct ares_server_failover_options *server_failover_opts***;**
Configure server failover retry behavior. When a DNS server fails to respond to a query, c-ares will deprioritize the server. On subsequent queries, servers with fewer consecutive failures will be selected in preference. However, in order to detect when such a server has recovered, c-ares will occasionally retry failed servers by probing with a copy of the query, without affecting the latency of the actual requested query. The *ares_server_failover_options* structure contains options to control this behavior. The *retry_chance* field gives the probability (1/N) of retrying a failed server on any given query. Setting to a value of 0 disables retries. The *retry_delay* field gives the minimum delay in milliseconds that c-ares will wait before retrying a specific failed server. If this option is not specificed then c-ares will use a probability of 10% and a minimum delay of 5 seconds.

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 default is not to rotate servers, however the

system configuration can specify the desire to rotate and this configuration value can negate such a system configuration.

RETURN VALUES

ares_init_options(3) and **ares_init(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.

ARES_ENOSERVER
No DNS servers were available to use.

NOTES

When initializing from **/etc/resolv.conf,** (or, alternatively when specified by the *resolvconf_path* path location) **ares_init_options(3)** and **ares_init(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 than 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_reinit(3), **ares_destroy**(3), **ares_dup**(3), **ares_library_init**(3), **ares_save_options**(3), **ares_set_servers**(3), **ares_set_sortlist**(3), **ares_threadsafety**(3)