NAME

ares_getaddrinfo − Initiate a host query by name and service

SYNOPSIS

#include <ares.h>

typedef void (*ares_addrinfo_callback)(void **arg*, int *status*,
int *timeouts*,
struct ares_addrinfo **result*)

void ares_getaddrinfo(ares_channel_t **channel*, const char **name*,
const char* *service*,
const struct ares_addrinfo_hints **hints*,
ares_addrinfo_callback *callback*, void **arg*)

DESCRIPTION

The **ares_getaddrinfo(3)** function initiates a host query by name on the name service channel identified by *channel*. The*name* and *service* parameters gi ve the hostname and service as NULL-terminated C strings. The *hints* parameter is an **ares_addrinfo_hints** structure:

```
struct ares_addrinfo_hints {
  int ai_flags;
  int ai_family;
  int ai_socktype;
  int ai_protocol;
};
```

ai_family
Specifies desired address family. AF_UNSPEC means return both AF_INET and AF_INET6.

ai_socktype
Specifies desired socket type, for example SOCK_STREAM or SOCK_DGRAM. Setting this to 0 means any type.

ai_protocol
Setting this to 0 means any protocol.

ai_flags
Specifies additional options, see below.

ARES_AI_NUMERICSERV
If this option is set *service* field will be treated as a numeric value.

ARES_AI_CANONNAME
The ares_addrinfo structure will return a canonical names list.

ARES_AI_NOSORT Result addresses will not be sorted and no connections to resolved addresses will be attempted.

ARES_AI_ENVHOSTS
Read hosts file path from the environment variable *CARES_HOSTS* .

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)*, *ares_destroy(3)* or *ares_cancel(3)*.

When the associated callback is called, it is called with a channel lock so care must be taken to ensure any processing is minimal to prevent DNS channel stalls.

The callback may be triggered from a different thread than the one which called *ares_getaddrinfo(3)*.

For integrators running their own event loops and not using **ARES_OPT_EVENT_THREAD**, care needs

to be taken to ensure any file descriptor lists are updated immediately within the eventloop when notified.

The callback argument *arg* is copied from the **ares_getaddrinfo(3)** 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_ENOTFOUND
The *name* was not found.

ARES_ENOMEM Memory was exhausted.

ARES_ESERVICE The textual service name provided could not be dereferenced into a port.

ARES_ECANCELLED
The query was cancelled.

ARES_EDESTRUCTION
The name service channel *channel* is being destroyed; the query will not be completed.

On successful completion of the query, the callback argument *result* points to a **struct ares_addrinfo** which contains two linked lists, one with resolved addresses and another with canonical names. Also included is the official name of the host (analogous to gethostbyname() h_name).

```
struct ares_addrinfo {
  struct ares_addrinfo_cname *cnames;
  struct ares_addrinfo_node  *nodes;
  char *name;
};
```

ares_addrinfo_node structure is similar to RFC3493 addrinfo, but without canonname and with extra ttl field.

```
struct ares_addrinfo_node {
  int                      ai_ttl;
  int                      ai_flags;
  int                      ai_family;
  int                      ai_socktype;
  int                      ai_protocol;
  ares_socklen_t           ai_addrlen;
  struct sockaddr          *ai_addr;
  struct ares_addrinfo_node *ai_next;
};
```

ares_addrinfo_cname structure is a linked list of CNAME records where *ttl* is a time to live *alias* is a label of the resource record and *name* is a value (canonical name) of the resource record. See RFC2181 10.1.1. CNAME terminology.

```
struct ares_addrinfo_cname {
  int                       ttl;
  char                     *alias;
  char                     *name;
  struct ares_addrinfo_cname *next;
};
```

The reserved memory has to be deleted by **ares_freeaddrinfo(3)**.

The result is sorted according to RFC6724 except:
- Rule 3 (Avoid deprecated addresses)

- Rule 4 (Prefer home addresses)
- Rule 7 (Prefer native transport)

Please note that the function will attempt a connection on each of the resolved addresses as per RFC6724.

AVAILABILITY

This function was added in c-ares 1.16.0, released in March 2020.

SEE ALSO

ares_freeaddrinfo(3)