NAME

ares_set_socket_functions, ares_set_socket_functions_ex − Set socket io callbacks

SYNOPSIS

```
#include <ares.h>

typedef enum {
  ARES_SOCKFUNC_FLAG_NONBLOCKING = 1 << 0
} ares_sockfunc_flags_t;

typedef enum {
  ARES_SOCKET_OPT_SENDBUF_SIZE,
  ARES_SOCKET_OPT_RECVBUF_SIZE,
  ARES_SOCKET_OPT_BIND_DEVICE,
  ARES_SOCKET_OPT_TCP_FASTOPEN
} ares_socket_opt_t;

typedef enum {
  ARES_SOCKET_CONN_TCP_FASTOPEN = 1 << 0
} ares_socket_connect_flags_t;

typedef enum {
  ARES_SOCKET_BIND_TCP = 1 << 0,
  ARES_SOCKET_BIND_CLIENT = 1 << 1
} ares_socket_bind_flags_t;

struct ares_socket_functions_ex {
  unsigned int version; /* ABI Version: must be "1" */
  unsigned int flags;

  ares_socket_t (*asocket)(int domain, int type, int protocol, void *user_data);
  int (*aclose)(ares_socket_t sock, void *user_data);
  int (*asetsockopt)(ares_socket_t sock, ares_socket_opt_t opt, const void *val,
                     ares_socklen_t val_size, void *user_data);
  int (*aconnect)(ares_socket_t sock, const struct sockaddr *address,
                  ares_socklen_t address_len, unsigned int flags,
                  void *user_data);
  ares_ssize_t (*arecvfrom)(ares_socket_t sock, void *buffer, size_t length,
                            int flags, struct sockaddr *address,
                            ares_socklen_t *address_len, void *user_data);
  ares_ssize_t (*asendto)(ares_socket_t sock, const void *buffer, size_t length,
                          int flags, const struct sockaddr *address,
                          ares_socklen_t address_len, void *user_data);
  int (*agetsockname)(ares_socket_t sock, struct sockaddr *address,
                      ares_socklen_t *address_len, void *user_data);
  int (*abind)(ares_socket_t sock, unsigned int flags,
               const struct sockaddr *address, socklen_t address_len,
               void *user_data);
  unsigned int (*aif_nametoindex)(const char *ifname, void *user_data);
  const char *(*aif_indextoname)(unsigned int ifindex, char *ifname_buf,
                                 size_t ifname_buf_len, void *user_data);
};

ares_status_t ares_set_socket_functions_ex(ares_channel_t *channel,
  const struct ares_socket_functions_ex *funcs, void *user_data);
```

```
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_t *channel,
                               const struct ares_socket_functions * functions,
                               void *user_data);
```

DESCRIPTION

ares_set_socket_functions_ex(3) sets a set of callback *functions* in the given ares channel handle. Cannot be used when **ARES_OPT_EVENT_THREAD** is passed to *ares_init_options(3)*. This function replaces the now deprecated **ares_set_socket_functions(3)** call.

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.

Some callbacks may be optional and are documented as such below, but failing to implement such callbacks will disable certain features within c-ares. It is strongly recommended to implement all callbacks.

All callback functions are expected to operate like their system equivalents, and to set **errno(2)** or **WSASetLastError(2)** to an appropriate error code on failure. It is strongly recommended that io callbacks are implemented to be asynchronous and indicated as such in the *flags* member. The io callbacks can return error codes of **EAGAIN**, **EWOULDBLOCK**, or **WSAEWOULDBLOCK** when they would otherwise block.

The *user_data* value is provided to each callback function invocation to serve as context.

The **ares_set_socket_functions_ex(3)** must provide the following structure members and callbacks (which are different from the **ares_set_socket_functions(3)** members and callbacks):

unsigned int *version*
: ABI Version of structure. Must be set to a value of "1".

unsigned int *flags*
: Flags available are specified in *ares_sockfunc_flags_t*.

ares_socket_t (*asocket***)(int** *domain***, int** *type***, int** *protocol***, void *** *user_data***)**
: *REQUIRED*. Creates an endpoint for communication and returns a descriptor. *domain*, *type*, and *protocol* each correspond to the parameters of **socket(2)**. Returns a handle to the newly created socket, or **ARES_SOCKET_BAD** on error.

int (*aclose***)(ares_socket_t** *fd***, void *** *user_data***)**
: *REQUIRED*. Closes the socket endpoint indicated by *fd*. See **close(2)**.

int (*asetsockopt***)(ares_socket_t** *fd***, ares_socket_opt_t** *opt***, const void *** *val***, ares_socklen_t** *val_size***, void *** *user_data***)**
: *REQUIRED*. Set socket option. This shares a similar syntax to the BSD *setsockopt(2)* call,

however c-ares uses different options for portability. The value is a pointer to the desired value, and each option has its own data type listed in the options below defined in *ares_socket_opt_t*.

int (**aconnect***)(ares_socket_t** *fd***, const struct sockaddr *** *addr***, ares_socklen_t** *addr_len***, unsigned int** *flags***, void *** *user_data***)**
REQUIRED. Initiate a connection to the address indicated by *addr* on a socket. Additional flags controlling behavior are in *ares_socket_connect_flags_t*. See **connect(2)**.

ares_ssize_t (**arecvfrom***)(ares_socket_t** *fd***, void *** *buffer***, size_t** *buf_size***, int** *flags***, struct sockaddr *** *addr***, ares_socklen_t *** *addr_len***, void *** *user_data***)**
REQUIRED. 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. The *flags* parameter is currently unused. See **recvfrom(2)**.

ares_ssize_t (**asendto***)(ares_socket_t** *fd***, const void *** *buffer***, size_t** *length***, int** *flags***, const struct sockaddr *** *address***, ares_socklen_t** *address_len***, void *** *user_data***)**
REQUIRED. Send data, as provided by the *buffer*, to the socket endpoint. The *flags* member may be used on systems that have **MSG_NOSIGNAL** defined but is otherwise unused. An *address* is provided primarily to support TCP FastOpen scenarios, which will be NULL in other circumstances. See **sendto(2)**.

int (**agetsockname***)(ares_socket_t** *fd***, struct sockaddr *** *address***, ares_socklen_t *** *address_len***, void *** *user_data***)**
Optional. Retrieve the local address of a socket and store it into the provided *address* buffer. May impact DNS Cookies if not provided. See **getsockname(2)**.

int (**abind***)(ares_socket_t** *fd***, unsigned int** *flags***, const struct sockaddr *** *address***, ares_socklen_t** *address_len***, void *** *user_data***)**
Optional. Bind the socket to an address. This can be used for client connections to bind the source address for packets before connect, or for server connections to bind to an address and port before listening. Currently c-ares only supports client connections. *fla gs* from *ares_socket_bind_flags_t* can be specified. See **bind(2)**.

unsigned int (**aif_nametoindex***)(const char *** *ifname***, void *** *user_data***)**
Optional. Convert an interface name into the interface index. If this callback is not specified, then IPv6 Link-Local DNS servers cannot be used. See **if_nametoindex(2)**.

const char * (**aif_indextoname***)(unsigned int** *ifindex***, char *** *ifname_buf***, size_t** *ifname_buf_len***, void *** *user_data***)**
Optional. Convert an interface index into the interface name. If this callback is not specified, then IPv6 Link-Local DNS servers cannot be used. *ifname_b uf* must be at least **IF_NAMESIZE** or **IFNAMSIZ** in size. See **if_indextoname(2)**.

ares_sockfunc_flags_t values:

ARES_SOCKFUNC_FLAG_NONBLOCKING
Used to indicate the implementation of the io functions are asynchronous.

ares_socket_opt_t values:

ARES_SOCKET_OPT_SENDBUF_SIZE
: Set the Send Buffer size. Value is a pointer to an int. (SO_SNDBUF).

ARES_SOCKET_OPT_RECVBUF_SIZE
: Set the Receive Buffer size. Value is a pointer to an int. (SO_RCVBUF).

ARES_SOCKET_OPT_BIND_DEVICE
: Set the network interface to use as the source for communication. Value is a C string. (SO_BINDTODEVICE)

ARES_SOCKET_OPT_TCP_FASTOPEN
: Enable TCP Fast Open. Value is a pointer to an *ares_bool_t*. On some systems this could be a no-op if it is known it is on by default and return success. Other systems may be a no-op if known the system does not support the feature and returns failure with errno set to **ENOSYS** or **WSASetLastError(WSAEOPNOTSUPP);**.

ares_socket_connect_flags_t values:

ARES_SOCKET_CONN_TCP_FASTOPEN
: Connect using TCP Fast Open.

ares_socket_bind_flags_t values:

ARES_SOCKET_BIND_TCP
: Bind is for a TCP connection.

ARES_SOCKET_BIND_CLIENT
: Bind is for a client connection, not server.

ares_set_socket_functions(3) sets a set of callback *functions* in the given ares channel handle. Cannot be used when **ARES_OPT_EVENT_THREAD** is passed to *ares_init_options(3)*. This function is deprecated as of c-ares 1.34.0 in favor of *ares_set_socket_functions_ex(3)*.

ares_set_socket_functions(3) allows you to choose to only implement some of the socket functions, and provide NULL to any others and c-ares will use its built-in system functions in that case.

All callback functions are expected to operate like their system equivalents, and to set **errno(2)** or **WSASetLastError(2)** to an appropriate error code on failure. It is strongly recommended all io functions behave asynchronously and return error codes of **EAGAIN**, **EWOULDBLOCK**, or **WSAEWOULDBLOCK** when they would otherwise block.

The *user_data* value is provided to each callback function invocation to serve as context.

The **ares_set_socket_functions(3)** must provide the following callbacks (which are different from the **ares_set_socket_functions_ex(3)** callbacks):

ares_socket_t (*asocket***)(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 ARES_SOCKET_BAD on error.

int (**aclose***)(ares_socket_t** *fd***, void *** *user_data***)**
Closes the socket endpoint indicated by *fd*. See **close(2)**.

int (**aconnect***)(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)**

ares_ssize_t (**arecvfrom***)(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)**

ares_ssize_t (**asendv***)(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_set_socket_functions(3)** struct provided is not copied but directly referenced, and must thus remain valid through out the channels and any created socket's lifetime. However, the **ares_set_socket_functions_ex(3)** struct is duplicated and does not need to survive past the call to the function.

AVAILABILITY

ares_socket_functions added in c-ares 1.13.0, ares_socket_functions_ex added in c-ares 1.34.0

SEE ALSO

ares_init_options(3), **socket**(2), **close**(2), **connect**(2), **recvfrom**(2), **sendto**(2), **bind**(2), **getsockname**(2), **setsockopt**(2), **writev**(2)