NAME

ares_process_fds, ares_process_fd, ares_process − Process events for name resolution

SYNOPSIS

```
#include <ares.h>

/*! Events used by ares_fd_events_t */
typedef enum {
  ARES_FD_EVENT_NONE  = 0,      /*!< No events */
  ARES_FD_EVENT_READ  = 1 << 0, /*!< Read event (including disconnect/error) */
  ARES_FD_EVENT_WRITE = 1 << 1  /*!< Write event */
} ares_fd_eventflag_t;

/*! Type holding a file descriptor and mask of events, used by
 *  ares_process_fds() */
typedef struct {
  ares_socket_t fd;     /*!< File descriptor */
  unsigned int  events; /*!< Mask of ares_fd_event_t */
} ares_fd_events_t;

typedef enum {
  ARES_PROCESS_FLAG_NONE       = 0,
  ARES_PROCESS_FLAG_SKIP_NON_FD = 1 << 0
} ares_process_flag_t;


ares_status_t ares_process_fds(ares_channel_t        *channel,
                               const ares_fd_events_t *events,
                               size_t                 nevents,
                               unsigned int           flags)

void ares_process_fd(ares_channel_t *channel,
                     ares_socket_t read_fd,
                     ares_socket_t write_fd)

void ares_process(ares_channel_t *channel,
                  fd_set *read_fds,
                  fd_set *write_fds)
```

DESCRIPTION

These functions must be used by integrators choosing not to use the EventThread enabled via **ARES_OPT_EVENT_THREAD** passed to **ares_init_options**. This assumes integrators already have their own event loop handling event notifications for various file descriptors and wish to do the same with their integration with c-ares.

The **ares_process_fds(3)** function handles input/output events on file descriptors and timeouts associated with queries pending on the channel identified by *channel*. The file descriptors to be processed are passed in an array of *ares_fd_events_t* data structures in the *fd* member, and events are a bitwise mask of *ares_fd_eventflag_t* in the *event* member. This function can also be used to process timeouts by passing NULL to the *events* member with *nevents* value of 0. Flags may also be specified in the *flags* field and are defined in **ares_process_flag_t**.

ARES_PROCESS_FLAG_SKIP_NON_FD can be specified to specifically skip any processing unrelated to the file descriptor events passed in, examples include timeout processing and cleanup handling. This is useful if an integrator knows they will be sending multiple *ares_process_fds(3)* requests and wants to skip

that extra processing. However, the integrator must send the final request with the flag so that timeout and other processing gets performed before their event loop waits on additional events.

It is allowable to use an *ares_fd_events_t* with *events* member of value *ARES_FD_EVENT_NONE* (0) if there are no events for a given file descriptor if an integrator wishes to simply maintain an array with all possible file descriptors and update readiness via the *event* member.

This function will return *ARES_ENOMEM* in out of memory conditions, otherwise will return *ARES_SUCCESS*.

This function is recommended over **ares_process_fd(3)** since it can handle processing of multiple file descriptors at once, thus skipping repeating additional logic such as timeout processing which would be required if calling **ares_process_fd(3)** for multiple file descriptors notified at the same time.

This function is typically used with the *ARES_OPT_SOCK_STATE_CB* option.

ares_timeout(3) should be used to retrieve the desired timeout, and when the timeout expires, the integrator must call **ares_process_fds(3)** with a NULL *events* array. (or **ares_process_fd(3)** with both sockets set to *ARES_SOCKET_BAD*). There is no need to do this if events are also delivered for any file descriptors as timeout processing will automatically be handled by any call to **ares_process_fds(3)** or **ares_process_fd(3)**.

The **ares_process_fd(3)** function is the same as **ares_process_fds(3)** except can only process a single read and write file descriptor at a time. New integrators should use **ares_process_fds(3)** if possible.

The **ares_process(3)** function works in the same manner, except it works on *fd_sets* as is used by **select(3)** and retrieved by **ares_fds(3)**. This method is deprecated and should not be used in modern applications due to known limitations to the **select(3)** implementation.

AVAILABILITY

ares_process_fds(3) was introduced in c-ares 1.34.0.

SEE ALSO

ares_fds(3), **ares_timeout**(3), **ares_init_options**(3) with *ARES_OPT_EVENT_THREAD* or *ARES_OPT_SOCK_STATE_CB*