Adare_net Manual

version 0.0.109

Preparing a party

- Create a network address and port
  - many
  - just one
- Create a presence in network (socket)

The server part of the party

- I'm at port (bind)
- I'm listening you! Please connect!
- I accepted you! I waited you forever! thanks for connecting!
- I accepted you! But I'm too Busy! Thanks for connecting or Bye!

The client part of the party

- I'm connecting to you at address and port server!

Party Start!

- receive
- send
- receive_from
- send_to
- plain raw data, vulgo stream_element_array
- buffered data, vulgo socket_buffer
- plain raw data ou buffered data ?

Apendixes

- Full Client and Server TCP/IP example
- Full Client and Server UDP/IP example
- Hints for developers and users of others Network Ada Libraries
  - Anet
  - Gnat-sockets
  - A minimum gnat project to work with.

Preparing a party

Create a network address and port

- Many (actually until 10 addresses)

```
declare
    many_addresses :  addresses_list_access := null;
begin
    procedure init_addresses
      (ip_or_host   =>  "duckduckgo.com",
       port         =>  "25000", -- ignored without bind(). Use "0" to choose automatically
       ai_socktype  =>  tcp, -- or udp
       ai_family    =>  any, -- or v4 or v6
       addr         =>  many_addresses
      );

    if many_addresses.all'Length < 1 then
      TEXT_IO.Put_Line (' none address discovered ');
      return;
    end if

    utils.show_address_and_port (many_addresses);
end;
```

- Just one

```
declare
    mi_address :  addresses_access := null;
begin
    procedure init_addresses
      (ip_or_host   =>  "duckduckgo.com",
       port         =>  "25000", -- ignored without bind(). Use "0" to choose automatically
       ai_socktype  =>  tcp, -- or udp
       ai_family    =>  any, -- or v4 or else v6
       addr         =>  mi_address
      );

    if mi_address.all'Length < 1 then
      TEXT_IO.Put_Line (' none address discovered ');
      return;
    end if

    utils.show_address_and_port (many_addresses);
end;
```

Create a presence in network (socket)

```
declare
    mi_presence :  socket_access := null;
begin
    if init_socket (mi_presence, many_addresses) then
      TEXT_IO.Put_Line (' Worked! ');
      return;
    end if
end;
```

- or

```
declare
    mi_presence :  socket_access := null;
begin
    if init_socket (mi_presence, mi_address) then
      TEXT_IO.Put_Line (' Worked! ');
      return;
    end if
end;
```