:c:type:`uv_udp_t` --- UDP handle

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 4); *backlink*

Unknown interpreted text role "c:type".

UDP handles encapsulate UDP communication for both clients and servers.

Data types

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 13)

Unknown directive type "c:type".

```
.. c:type:: uv_udp_t

    UDP handle type.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 17)

Unknown directive type "c:type".

```
.. c:type:: uv_udp_send_t

    UDP send request type.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 21)

Unknown directive type "c:type".

```
.. c:type:: uv_udp_flags

    Flags used in :c:func:`uv_udp_bind` and :c:type:`uv_udp_recv_cb`..

    ::

        enum uv_udp_flags {
            /* Disables dual stack mode. */
            UV_UDP_IPV6ONLY = 1,
            /*
            * Indicates message was truncated because read buffer was too small. The
            * remainder was discarded by the OS. Used in uv_udp_recv_cb.
            */
            UV_UDP_PARTIAL = 2,
            /*
            * Indicates if SO_REUSEADDR will be set when binding the handle in
            * uv_udp_bind.
            * This sets the SO_REUSEPORT socket flag on the BSDs and OS X. On other
            * Unix platforms, it sets the SO_REUSEADDR flag. What that means is that
            * multiple threads or processes can bind to the same address without error
            * (provided they all set the flag) but only the last one to bind will receive
            * any traffic, in effect "stealing" the port from the previous listener.
            */
            UV_UDP_REUSEADDR = 4,
            /*
             * Indicates that the message was received by recvmmsg, so the buffer provided
             * must not be freed by the recv_cb callback.
             */
            UV_UDP_MMSG_CHUNK = 8,
            /*
             * Indicates that the buffer provided has been fully utilized by recvmmsg and
             * that it should now be freed by the recv_cb callback. When this flag is set
             * in uv_udp_recv_cb, nread will always be 0 and addr will always be NULL.
             */
            UV_UDP_MMSG_FREE = 16,
            /*
             * Indicates if IP_RECVERR/IPV6_RECVERR will be set when binding the handle.
             * This sets IP_RECVERR for IPv4 and IPV6_RECVERR for IPv6 UDP sockets on
             * Linux. This stops the Linux kernel from supressing some ICMP error messages
             * and enables full ICMP error reporting for faster failover.
             * This flag is no-op on platforms other than Linux.
             */
            UV_UDP_LINUX_RECVERR = 32,
            /*
            * Indicates that recvmmsg should be used, if available.
            */
            UV_UDP_RECVMMSG = 256
        };
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 70)

Unknown directive type "c:type".

```
.. c:type:: void (*uv_udp_send_cb)(uv_udp_send_t* req, int status)

    Type definition for callback passed to :c:func:`uv_udp_send`, which is
    called after the data was sent.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 75)

Unknown directive type "c:type".

```
.. c:type:: void (*uv_udp_recv_cb)(uv_udp_t* handle, ssize_t nread, const uv_buf_t* buf, const struct sockaddr* addr, unsigned fla

    Type definition for callback passed to :c:func:`uv_udp_recv_start`, which
    is called when the endpoint receives data.

    * `handle`: UDP handle
    * `nread`:  Number of bytes that have been received.
      0 if there is no more data to read. Note that 0 may also mean that an
      empty datagram was received (in this case `addr` is not NULL). < 0 if
      a transmission error was detected; if using :man:`recvmmsg(2)` no more
      chunks will be received and the buffer can be freed safely.
    * `buf`: :c:type:`uv_buf_t` with the received data.
    * `addr`: ``struct sockaddr*`` containing the address of the sender.
      Can be NULL. Valid for the duration of the callback only.
    * `flags`: One or more or'ed UV_UDP_* constants.

    The callee is responsible for freeing the buffer, libuv does not reuse it.
    The buffer may be a null buffer (where `buf->base` == NULL and `buf->len` == 0)
    on error.

    When using :man:`recvmmsg(2)`, chunks will have the `UV_UDP_MMSG_CHUNK` flag set,
    those must not be freed. If no errors occur, there will be a final callback with
    `nread` set to 0, `addr` set to NULL and the buffer pointing at the initially
    allocated data with the `UV_UDP_MMSG_CHUNK` flag cleared and the `UV_UDP_MMSG_FREE`
    flag set. If a UDP socket error occurs, `nread` will be < 0. In either scenario,
    the callee can now safely free the provided buffer.

    .. versionchanged:: 1.40.0 added the `UV_UDP_MMSG_FREE` flag.

    .. note::
        The receive callback will be called with `nread` == 0 and `addr` == NULL when there is
        nothing to read, and with `nread` == 0 and `addr` != NULL when an empty UDP packet is
        received.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 109)

Unknown directive type "c:enum".

```
.. c:enum:: uv_membership

    Membership type for a multicast address.

    ::

        typedef enum {
            UV_LEAVE_GROUP = 0,
            UV_JOIN_GROUP
        } uv_membership;
```

Public members

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 124)

Unknown directive type "c:member".

```
.. c:member:: size_t uv_udp_t.send_queue_size

    Number of bytes queued for sending. This field strictly shows how much
    information is currently queued.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 129)

Unknown directive type "c:member".

```
.. c:member:: size_t uv_udp_t.send_queue_count

    Number of send requests currently in the queue awaiting to be processed.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 133)

Unknown directive type "c:member".

```
.. c:member:: uv_udp_t* uv_udp_send_t.handle

    UDP handle where this send request is taking place.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 137)

Unknown directive type "seealso".

```
.. seealso:: The :c:type:`uv_handle_t` members also apply.
```

API

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 143)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_init(uv_loop_t* loop, uv_udp_t* handle)

    Initialize a new UDP handle. The actual socket is created lazily.
    Returns 0 on success.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 148)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_init_ex(uv_loop_t* loop, uv_udp_t* handle, unsigned int flags)

    Initialize the handle with the specified flags. The lower 8 bits of the `flags`
    parameter are used as the socket domain. A socket will be created for the given domain.
    If the specified domain is ``AF_UNSPEC`` no socket is created, just like :c:func:`uv_udp_init`.

    The remaining bits can be used to set one of these flags:

    * `UV_UDP_RECVMMSG`: if set, and the platform supports it, :man:`recvmmsg(2)` will
      be used.

    .. versionadded:: 1.7.0
    .. versionchanged:: 1.37.0 added the `UV_UDP_RECVMMSG` flag.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, line 162)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_open(uv_udp_t* handle, uv_os_sock_t sock)

    Opens an existing file descriptor or Windows SOCKET as a UDP handle.

    Unix only:
    The only requirement of the `sock` argument is that it follows the datagram
```

```
        contract (works in unconnected mode, supports sendmsg()/recvmsg(), etc).
        In other words, other datagram-type sockets like raw sockets or netlink
        sockets can also be passed to this function.

        .. versionchanged:: 1.2.1 the file descriptor is set to non-blocking mode.

        .. note::
            The passed file descriptor or SOCKET is not checked for its type, but
            it's required that it represents a valid datagram socket.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 178)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_bind(uv_udp_t* handle, const struct sockaddr* addr, unsigned int flags)

    Bind the UDP handle to an IP address and port.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param addr: `struct sockaddr_in` or `struct sockaddr_in6`
        with the address and port to bind to.

    :param flags: Indicate how the socket will be bound,
        ``UV_UDP_IPV6ONLY``, ``UV_UDP_REUSEADDR``, and ``UV_UDP_RECVERR``
        are supported.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 194)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_connect(uv_udp_t* handle, const struct sockaddr* addr)

    Associate the UDP handle to a remote address and port, so every
    message sent by this handle is automatically sent to that destination.
    Calling this function with a `NULL` `addr` disconnects the handle.
    Trying to call `uv_udp_connect()` on an already connected handle will result
    in an `UV_EISCONN` error. Trying to disconnect a handle that is not
    connected will return an `UV_ENOTCONN` error.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param addr: `struct sockaddr_in` or `struct sockaddr_in6`
        with the address and port to associate to.

    :returns: 0 on success, or an error code < 0 on failure.

    .. versionadded:: 1.27.0
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 213)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_getpeername(const uv_udp_t* handle, struct sockaddr* name, int* namelen)

    Get the remote IP and port of the UDP handle on connected UDP handles.
    On unconnected handles, it returns `UV_ENOTCONN`.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init` and bound.

    :param name: Pointer to the structure to be filled with the address data.
        In order to support IPv4 and IPv6 `struct sockaddr_storage` should be
        used.

    :param namelen: On input it indicates the data of the `name` field. On
        output it indicates how much of it was filled.

    :returns: 0 on success, or an error code < 0 on failure

    .. versionadded:: 1.27.0
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 232)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_getsockname(const uv_udp_t* handle, struct sockaddr* name, int* namelen)

    Get the local IP and port of the UDP handle.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init` and bound.

    :param name: Pointer to the structure to be filled with the address data.
        In order to support IPv4 and IPv6 `struct sockaddr_storage` should be
        used.

    :param namelen: On input it indicates the data of the `name` field. On
        output it indicates how much of it was filled.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 248)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_set_membership(uv_udp_t* handle, const char* multicast_addr, const char* interface_addr, uv_membership

    Set membership for a multicast address

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param multicast_addr: Multicast address to set membership for.

    :param interface_addr: Interface address.

    :param membership: Should be ``UV_JOIN_GROUP`` or ``UV_LEAVE_GROUP``.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 263)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_set_source_membership(uv_udp_t* handle, const char* multicast_addr, const char* interface_addr, const cl

    Set membership for a source-specific multicast group.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param multicast_addr: Multicast address to set membership for.

    :param interface_addr: Interface address.

    :param source_addr: Source address.

    :param membership: Should be ``UV_JOIN_GROUP`` or ``UV_LEAVE_GROUP``.

    :returns: 0 on success, or an error code < 0 on failure.

    .. versionadded:: 1.32.0
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 282)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_set_multicast_loop(uv_udp_t* handle, int on)

    Set IP multicast loop flag. Makes multicast packets loop back to
    local sockets.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param on: 1 for on, 0 for off.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 294)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_set_multicast_ttl(uv_udp_t* handle, int ttl)

    Set the multicast ttl.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param ttl: 1 through 255.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 305)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_set_multicast_interface(uv_udp_t* handle, const char* interface_addr)

    Set the multicast interface to send or receive data on.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param interface_addr: interface address.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 316)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_set_broadcast(uv_udp_t* handle, int on)

    Set broadcast on or off.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param on: 1 for on, 0 for off.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 327)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_set_ttl(uv_udp_t* handle, int ttl)

    Set the time to live.

    :param handle: UDP handle. Should have been initialized with
        :c:func:`uv_udp_init`.

    :param ttl: 1 through 255.

    :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst, line 338)

Unknown directive type "c:function".

```
.. c:function:: int uv_udp_send(uv_udp_send_t* req, uv_udp_t* handle, const uv_buf_t bufs[], unsigned int nbufs, const struct socka

    Send data over the UDP socket. If the socket has not previously been bound
    with :c:func:`uv_udp_bind` it will be bound to 0.0.0.0
    (the "all interfaces" IPv4 address) and a random port number.

    On Windows if the `addr` is initialized to point to an unspecified address
    (``0.0.0.0`` or ``::``) it will be changed to point to ``localhost``.
```

```
        This is done to match the behavior of Linux systems.

        For connected UDP handles, `addr` must be set to `NULL`, otherwise it will
        return `UV_EISCONN` error.

        For connectionless UDP handles, `addr` cannot be `NULL`, otherwise it will
        return `UV_EDESTADDRREQ` error.

        :param req: UDP request handle. Need not be initialized.

        :param handle: UDP handle. Should have been initialized with
            :c:func:`uv_udp_init`.

        :param bufs: List of buffers to send.

        :param nbufs: Number of buffers in `bufs`.

        :param addr: `struct sockaddr_in` or `struct sockaddr_in6` with the
            address and port of the remote peer.

        :param send_cb: Callback to invoke when the data has been sent out.

        :returns: 0 on success, or an error code < 0 on failure.

        .. versionchanged:: 1.19.0 added ``0.0.0.0`` and ``::`` to ``localhost``
            mapping

        .. versionchanged:: 1.27.0 added support for connected sockets
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst`, line 375)

Unknown directive type "c:function".

```
    .. c:function:: int uv_udp_try_send(uv_udp_t* handle, const uv_buf_t bufs[], unsigned int nbufs, const struct sockaddr* addr)

        Same as :c:func:`uv_udp_send`, but won't queue a send request if it can't
        be completed immediately.

        For connected UDP handles, `addr` must be set to `NULL`, otherwise it will
        return `UV_EISCONN` error.

        For connectionless UDP handles, `addr` cannot be `NULL`, otherwise it will
        return `UV_EDESTADDRREQ` error.

        :returns: >= 0: number of bytes sent (it matches the given buffer size).
            < 0: negative error code (``UV_EAGAIN`` is returned when the message
            can't be sent immediately).

        .. versionchanged:: 1.27.0 added support for connected sockets
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst`, line 392)

Unknown directive type "c:function".

```
    .. c:function:: int uv_udp_recv_start(uv_udp_t* handle, uv_alloc_cb alloc_cb, uv_udp_recv_cb recv_cb)

        Prepare for receiving data. If the socket has not previously been bound
        with :c:func:`uv_udp_bind` it is bound to 0.0.0.0 (the "all interfaces"
        IPv4 address) and a random port number.

        :param handle: UDP handle. Should have been initialized with
            :c:func:`uv_udp_init`.

        :param alloc_cb: Callback to invoke when temporary storage is needed.

        :param recv_cb: Callback to invoke with received data.

        :returns: 0 on success, or an error code < 0 on failure.

        .. note::
            When using :man:`recvmmsg(2)`, the number of messages received at a time is limited
            by the number of max size dgrams that will fit into the buffer allocated in `alloc_cb`, and
            `suggested_size` in `alloc_cb` for udp_recv is always set to the size of 1 max size dgram.

        .. versionchanged:: 1.35.0 added support for :man:`recvmmsg(2)` on supported platforms).
                            The use of this feature requires a buffer larger than
                            2 * 64KB to be passed to `alloc_cb`.
        .. versionchanged:: 1.37.0 :man:`recvmmsg(2)` support is no longer enabled implicitly,
                            it must be explicitly requested by passing the `UV_UDP_RECVMMSG` flag to
                            :c:func:`uv_udp_init_ex`.
        .. versionchanged:: 1.39.0 :c:func:`uv_udp_using_recvmmsg` can be used in `alloc_cb` to
                            determine if a buffer sized for use with :man:`recvmmsg(2)` should be
                            allocated for the current handle/platform.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst`, line 422)

Unknown directive type "c:function".

```
    .. c:function:: int uv_udp_using_recvmmsg(uv_udp_t* handle)

        Returns 1 if the UDP handle was created with the `UV_UDP_RECVMMSG` flag
        and the platform supports :man:`recvmmsg(2)`, 0 otherwise.

        .. versionadded:: 1.39.0
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst`, line 429)

Unknown directive type "c:function".

```
    .. c:function:: int uv_udp_recv_stop(uv_udp_t* handle)

        Stop listening for incoming datagrams.

        :param handle: UDP handle. Should have been initialized with
            :c:func:`uv_udp_init`.

        :returns: 0 on success, or an error code < 0 on failure.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)udp.rst`, line 438)

Unknown directive type "c:function".

```
    .. c:function:: size_t uv_udp_get_send_queue_size(const uv_udp_t* handle)

        Returns `handle->send_queue_size`.
    .. versionadded:: 1.19.0
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, **line 444)**

Unknown directive type "c:function".

```
.. c:function:: size_t uv_udp_get_send_queue_count(const uv_udp_t* handle)

    Returns `handle->send_queue_count`.

    .. versionadded:: 1.19.0
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master) (deps) (uv) (docs) (src)udp.rst`, **line 450)**

Unknown directive type "seealso".

```
.. seealso:: The :c:type:`uv_handle_t` API functions also apply.
```