:c:type:`uv_handle_t` --- Base handle

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

Unknown interpreted text role "c:type".

uv_handle_t is the base type for all libuv handle types.

Structures are aligned so that any libuv handle can be cast to *uv_handle_t*. All API functions defined here work with any handle type.

Libuv handles are not movable. Pointers to handle structures passed to functions must remain valid for the duration of the requested operation. Take care when using stack allocated handles.

Data types

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 19)

Unknown directive type "c:type".

```
.. c:type:: uv_handle_t

    The base libuv handle type.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 23)

Unknown directive type "c:enum".

```
.. c:enum:: uv_handle_type

    The kind of the libuv handle.

    ::

        typedef enum {
          UV_UNKNOWN_HANDLE = 0,
          UV_ASYNC,
          UV_CHECK,
          UV_FS_EVENT,
          UV_FS_POLL,
          UV_HANDLE,
          UV_IDLE,
          UV_NAMED_PIPE,
          UV_POLL,
          UV_PREPARE,
          UV_PROCESS,
          UV_STREAM,
          UV_TCP,
          UV_TIMER,
          UV_TTY,
          UV_UDP,
          UV_SIGNAL,
          UV_FILE,
          UV_HANDLE_TYPE_MAX
        } uv_handle_type;
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 51)

Unknown directive type "c:type".

```
.. c:type:: uv_any_handle

    Union of all handle types.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 55)

Unknown directive type "c:type".

```
.. c:type:: void (*uv_alloc_cb)(uv_handle_t* handle, size_t suggested_size, uv_buf_t* buf)

    Type definition for callback passed to :c:func:`uv_read_start` and
    :c:func:`uv_udp_recv_start`. The user must allocate memory and fill the supplied
    :c:type:`uv_buf_t` structure. If NULL is assigned as the buffer's base or 0 as its length,
    a ``UV_ENOBUFS`` error will be triggered in the :c:type:`uv_udp_recv_cb` or the
    :c:type:`uv_read_cb` callback.

    Each buffer is used only once and the user is responsible for freeing it in the
    :c:type:`uv_udp_recv_cb` or the :c:type:`uv_read_cb` callback.

    A suggested size (65536 at the moment in most cases) is provided, but it's just an indication,
    not related in any way to the pending data to be read. The user is free to allocate the amount
    of memory they decide.

    As an example, applications with custom allocation schemes such as using freelists, allocation
    pools or slab based allocators may decide to use a different size which matches the memory
    chunks they already have.

    Example:

    ::

        static void my_alloc_cb(uv_handle_t* handle, size_t suggested_size, uv_buf_t* buf) {
          buf->base = malloc(suggested_size);
          buf->len = suggested_size;
        }
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 83)

Unknown directive type "c:type".

```
.. c:type:: void (*uv_close_cb)(uv_handle_t* handle)

    Type definition for callback passed to :c:func:`uv_close`.
```

Public members

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 91)

Unknown directive type "c:member".

```
.. c:member:: uv_loop_t* uv_handle_t.loop

    Pointer to the :c:type:`uv_loop_t` the handle is running on. Readonly.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 95)

Unknown directive type "c:member".

```
.. c:member:: uv_handle_type uv_handle_t.type

    The :c:type:`uv_handle_type`, indicating the type of the underlying handle. Readonly.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 99)

Unknown directive type "c:member".

```
.. c:member:: void* uv_handle_t.data

    Space for user-defined arbitrary data. libuv does not use this field.
```

API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 107)

Unknown directive type "c:macro".

```
.. c:macro:: UV_HANDLE_TYPE_MAP(iter_macro)

    Macro that expands to a series of invocations of `iter_macro` for
    each of the handle types. `iter_macro` is invoked with two
    arguments: the name of the `uv_handle_type` element without the
    `UV_` prefix, and the name of the corresponding structure type
    without the `uv_` prefix and `_t` suffix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 115)

Unknown directive type "c:function".

```
.. c:function:: int uv_is_active(const uv_handle_t* handle)

    Returns non-zero if the handle is active, zero if it's inactive. What
    "active" means depends on the type of handle:

    - A uv_async_t handle is always active and cannot be deactivated, except
      by closing it with uv_close().

    - A uv_pipe_t, uv_tcp_t, uv_udp_t, etc. handle - basically any handle that
      deals with i/o - is active when it is doing something that involves i/o,
      like reading, writing, connecting, accepting new connections, etc.

    - A uv_check_t, uv_idle_t, uv_timer_t, etc. handle is active when it has
      been started with a call to uv_check_start(), uv_idle_start(), etc.

    Rule of thumb: if a handle of type `uv_foo_t` has a `uv_foo_start()`
    function, then it's active from the moment that function is called.
    Likewise, `uv_foo_stop()` deactivates the handle again.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 134)

Unknown directive type "c:function".

```
.. c:function:: int uv_is_closing(const uv_handle_t* handle)

    Returns non-zero if the handle is closing or closed, zero otherwise.

    .. note::
        This function should only be used between the initialization of the handle and the
        arrival of the close callback.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 142)

Unknown directive type "c:function".

```
.. c:function:: void uv_close(uv_handle_t* handle, uv_close_cb close_cb)

    Request handle to be closed. `close_cb` will be called asynchronously after
    this call. This MUST be called on each handle before memory is released.
    Moreover, the memory can only be released in `close_cb` or after it has
    returned.

    Handles that wrap file descriptors are closed immediately but
    `close_cb` will still be deferred to the next iteration of the event loop.
    It gives you a chance to free up any resources associated with the handle.

    In-progress requests, like uv_connect_t or uv_write_t, are cancelled and
    have their callbacks called asynchronously with status=UV_ECANCELED.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 156)

Unknown directive type "c:function".

```
.. c:function:: void uv_ref(uv_handle_t* handle)

    Reference the given handle. References are idempotent, that is, if a handle
    is already referenced calling this function again will have no effect.

    See :ref:`refcount`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 163)

Unknown directive type "c:function".

```
.. c:function:: void uv_unref(uv_handle_t* handle)

    Un-reference the given handle. References are idempotent, that is, if a handle
    is not referenced calling this function again will have no effect.

    See :ref:`refcount`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 170)

Unknown directive type "c:function".

```
.. c:function:: int uv_has_ref(const uv_handle_t* handle)

    Returns non-zero if the handle referenced, zero otherwise.

    See :ref:`refcount`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 176)

Unknown directive type "c:function".

```
.. c:function:: size_t uv_handle_size(uv_handle_type type)

    Returns the size of the given handle type. Useful for FFI binding writers
    who don't want to know the structure layout.
```

Miscellaneous API functions

The following API functions take a :c:type:`uv_handle_t` argument but they work just for some handle types.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 185); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 188)

Unknown directive type "c:function".

```
.. c:function:: int uv_send_buffer_size(uv_handle_t* handle, int* value)

    Gets or sets the size of the send buffer that the operating
    system uses for the socket.

    If `*value` == 0, then it will set `*value` to the current send buffer size.
    If `*value` > 0 then it will use `*value` to set the new send buffer size.

    On success, zero is returned. On error, a negative result is
    returned.

    This function works for TCP, pipe and UDP handles on Unix and for TCP and
    UDP handles on Windows.

    .. note::
        Linux will set double the size and return double the size of the original set value.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 205)

Unknown directive type "c:function".

```
.. c:function:: int uv_recv_buffer_size(uv_handle_t* handle, int* value)

    Gets or sets the size of the receive buffer that the operating
    system uses for the socket.

    If `*value` == 0, then it will set `*value` to the current receive buffer size.
    If `*value` > 0 then it will use `*value` to set the new receive buffer size.

    On success, zero is returned. On error, a negative result is
    returned.

    This function works for TCP, pipe and UDP handles on Unix and for TCP and
    UDP handles on Windows.

    .. note::
        Linux will set double the size and return double the size of the original set value.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 222)

Unknown directive type "c:function".

```
.. c:function:: int uv_fileno(const uv_handle_t* handle, uv_os_fd_t* fd)

    Gets the platform dependent file descriptor equivalent.

    The following handles are supported: TCP, pipes, TTY, UDP and poll. Passing
    any other handle type will fail with `UV_EINVAL`.

    If a handle doesn't have an attached file descriptor yet or the handle
    itself has been closed, this function will return `UV_EBADF`.

    .. warning::
        Be very careful when using this function. libuv assumes it's in control of the file
        descriptor so any change to it may lead to malfunction.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 236)

Unknown directive type "c:function".

```
.. c:function:: uv_loop_t* uv_handle_get_loop(const uv_handle_t* handle)

    Returns `handle->loop`.

    .. 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)handle.rst, line 242)

Unknown directive type "c:function".

```
.. c:function:: void* uv_handle_get_data(const uv_handle_t* handle)

    Returns `handle->data`.

    .. 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)handle.rst, line 248)

Unknown directive type "c:function".

```
.. c:function:: void* uv_handle_set_data(uv_handle_t* handle, void* data)

    Sets `handle->data` to `data`.

    .. 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)handle.rst, line 254)

Unknown directive type "c:function".

```
.. c:function:: uv_handle_type uv_handle_get_type(const uv_handle_t* handle)

    Returns `handle->type`.

    .. 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)handle.rst, line 260)

Unknown directive type "c:function".

```
.. c:function:: const char* uv_handle_type_name(uv_handle_type type)

    Returns the name for the equivalent struct for a given handle type,
    e.g. `"pipe"` (as in :c:type:`uv_pipe_t`) for `UV_NAMED_PIPE`.

    If no such handle type exists, this returns `NULL`.

    .. versionadded:: 1.19.0
```

Reference counting

The libuv event loop (if run in the default mode) will run until there are no active *and* referenced handles left. The user can force the loop to exit early by unreferencing handles which are active, for example by calling :c:func:`uv_unref` after calling :c:func:`uv_timer_start`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 274); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)handle.rst, line 274); *backlink*

Unknown interpreted text role "c:func".

A handle can be referenced or unreferenced, the refcounting scheme doesn't use a counter, so both operations are idempotent.

All handles are referenced when active by default, see :c:func:`uv_is_active` for a more detailed explanation on what being *active* involves.

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

Unknown interpreted text role "c:func".