:c:type:`uv_tcp_t` --- TCP handle

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

Unknown interpreted text role "c:type".

TCP handles are used to represent both TCP streams and servers.

:c:type:`uv_tcp_t` is a 'subclass' of :c:type:`uv_stream_t`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 9); *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)tcp.rst, line 9); *backlink*

Unknown interpreted text role "c:type".

Data types

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 15)

Unknown directive type "c:type".

```
.. c:type:: uv_tcp_t

    TCP handle type.
```

Public members

N/A

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 25)

Unknown directive type "seealso".

```
.. seealso:: The :c:type:`uv_stream_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)tcp.rst, line 31)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_init(uv_loop_t* loop, uv_tcp_t* handle)

    Initialize the handle. No socket is created as of yet.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 35)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_init_ex(uv_loop_t* loop, uv_tcp_t* handle, unsigned int flags)

    Initialize the handle with the specified flags. At the moment only 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_tcp_init`.

    .. versionadded:: 1.7.0
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 44)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_open(uv_tcp_t* handle, uv_os_sock_t sock)

    Open an existing file descriptor or SOCKET as a TCP handle.
```

```
.. 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 stream socket.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 54)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_nodelay(uv_tcp_t* handle, int enable)

    Enable `TCP_NODELAY`, which disables Nagle's algorithm.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 58)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_keepalive(uv_tcp_t* handle, int enable, unsigned int delay)

    Enable / disable TCP keep-alive. `delay` is the initial delay in seconds,
    ignored when `enable` is zero.

    After `delay` has been reached, 10 successive probes, each spaced 1 second
    from the previous one, will still happen. If the connection is still lost
    at the end of this procedure, then the handle is destroyed with a
    ``UV_ETIMEDOUT`` error passed to the corresponding callback.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 68)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_simultaneous_accepts(uv_tcp_t* handle, int enable)

    Enable / disable simultaneous asynchronous accept requests that are
    queued by the operating system when listening for new TCP connections.

    This setting is used to tune a TCP server for the desired performance.
    Having simultaneous accepts can significantly improve the rate of accepting
    connections (which is why it is enabled by default) but may lead to uneven
    load distribution in multi-process setups.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 78)

Unknown directive type "c:function".

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

    Bind the handle to an address and port. `addr` should point to an
    initialized ``struct sockaddr_in`` or ``struct sockaddr_in6``.

    When the port is already taken, you can expect to see an ``UV_EADDRINUSE``
    error from :c:func:`uv_listen` or :c:func:`uv_tcp_connect`. That is,
    a successful call to this function does not guarantee that the call
    to :c:func:`uv_listen` or :c:func:`uv_tcp_connect` will succeed as well.

    `flags` can contain ``UV_TCP_IPV6ONLY``, in which case dual-stack support
    is disabled and only IPv6 is used.
```

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

Unknown directive type "c:function".

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

    Get the current address to which the handle is bound. `name` must point to
    a valid and big enough chunk of memory, ``struct sockaddr_storage`` is
    recommended for IPv4 and IPv6 support.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 97)

Unknown directive type "c:function".

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

    Get the address of the peer connected to the handle. `name` must point to
    a valid and big enough chunk of memory, ``struct sockaddr_storage`` is
    recommended for IPv4 and IPv6 support.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 103)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_connect(uv_connect_t* req, uv_tcp_t* handle, const struct sockaddr* addr, uv_conn

    Establish an IPv4 or IPv6 TCP connection. Provide an initialized TCP handle
    and an uninitialized :c:type:`uv_connect_t`. `addr` should point to an
    initialized ``struct sockaddr_in`` or ``struct sockaddr_in6``.

    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.

    The callback is made when the connection has been established or when a
    connection error happened.

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 119)

Unknown directive type "seealso".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)tcp.rst, line 121)

Unknown directive type "c:function".

```
.. c:function:: int uv_tcp_close_reset(uv_tcp_t* handle, uv_close_cb close_cb)

    Resets a TCP connection by sending a RST packet. This is accomplished by
    setting the `SO_LINGER` socket option with a linger interval of zero and
    then calling :c:func:`uv_close`.
    Due to some platform inconsistencies, mixing of :c:func:`uv_shutdown` and
    :c:func:`uv_tcp_close_reset` calls is not allowed.

    .. 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)tcp.rst, line 131)

Unknown directive type "c:function".

```
.. c:function:: int uv_socketpair(int type, int protocol, uv_os_sock_t socket_vector[2], int flags0, int fla

    Create a pair of connected sockets with the specified properties.
    The resulting handles can be passed to `uv_tcp_open`, used with `uv_spawn`,
    or for any other purpose.

    Valid values for `flags0` and `flags1` are:

      - UV_NONBLOCK_PIPE: Opens the specified socket handle for `OVERLAPPED`
        or `FIONBIO`/`O_NONBLOCK` I/O usage.
        This is recommended for handles that will be used by libuv,
        and not usually recommended otherwise.

    Equivalent to :man:`socketpair(2)` with a domain of AF_UNIX.

    .. versionadded:: 1.41.0
```