:c:type:`uv_process_t` --- Process handle

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

Unknown interpreted text role "c:type".

Process handles will spawn a new process and allow the user to control it and establish communication channels with it using streams.

Data types

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 14)

Unknown directive type "c:type".

```
.. c:type:: uv_process_t

    Process handle type.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 18)

Unknown directive type "c:type".

```
.. c:type:: uv_process_options_t

    Options for spawning the process (passed to :c:func:`uv_spawn`.

    ::

        typedef struct uv_process_options_s {
            uv_exit_cb exit_cb;
            const char* file;
            char** args;
            char** env;
            const char* cwd;
            unsigned int flags;
            int stdio_count;
            uv_stdio_container_t* stdio;
            uv_uid_t uid;
            uv_gid_t gid;
        } uv_process_options_t;
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 37)

Unknown directive type "c:type".

```
.. c:type:: void (*uv_exit_cb)(uv_process_t*, int64_t exit_status, int term_signal)

    Type definition for callback passed in :c:type:`uv_process_options_t` which
    will indicate the exit status and the signal that caused the process to
    terminate, if any.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 43)

Unknown directive type "c:type".

```
.. c:type:: uv_process_flags

    Flags to be set on the flags field of :c:type:`uv_process_options_t`.

    ::

        enum uv_process_flags {
            /*
            * Set the child process' user id.
            */
            UV_PROCESS_SETUID = (1 << 0),
            /*
            * Set the child process' group id.
            */
```

```
            UV_PROCESS_SETGID = (1 << 1),
            /*
            * Do not wrap any arguments in quotes, or perform any other escaping, when
            * converting the argument list into a command line string. This option is
            * only meaningful on Windows systems. On Unix it is silently ignored.
            */
            UV_PROCESS_WINDOWS_VERBATIM_ARGUMENTS = (1 << 2),
            /*
            * Spawn the child process in a detached state - this will make it a process
            * group leader, and will effectively enable the child to keep running after
            * the parent exits. Note that the child process will still keep the
            * parent's event loop alive unless the parent process calls uv_unref() on
            * the child's process handle.
            */
            UV_PROCESS_DETACHED = (1 << 3),
            /*
            * Hide the subprocess window that would normally be created. This option is
            * only meaningful on Windows systems. On Unix it is silently ignored.
            */
            UV_PROCESS_WINDOWS_HIDE = (1 << 4),
            /*
            * Hide the subprocess console window that would normally be created. This
            * option is only meaningful on Windows systems. On Unix it is silently
            * ignored.
            */
            UV_PROCESS_WINDOWS_HIDE_CONSOLE = (1 << 5),
            /*
            * Hide the subprocess GUI window that would normally be created. This
            * option is only meaningful on Windows systems. On Unix it is silently
            * ignored.
            */
            UV_PROCESS_WINDOWS_HIDE_GUI = (1 << 6)
        };
```

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

Unknown directive type "c:type".

```
.. c:type:: uv_stdio_container_t

    Container for each stdio handle or fd passed to a child process.

    ::

        typedef struct uv_stdio_container_s {
            uv_stdio_flags flags;
            union {
                uv_stream_t* stream;
                int fd;
            } data;
        } uv_stdio_container_t;
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 105)

Unknown directive type "c:enum".

```
.. c:enum:: uv_stdio_flags

    Flags specifying how a stdio should be transmitted to the child process.

    ::

        typedef enum {
            UV_IGNORE = 0x00,
            UV_CREATE_PIPE = 0x01,
            UV_INHERIT_FD = 0x02,
            UV_INHERIT_STREAM = 0x04,
            /*
            * When UV_CREATE_PIPE is specified, UV_READABLE_PIPE and UV_WRITABLE_PIPE
            * determine the direction of flow, from the child process' perspective. Both
            * flags may be specified to create a duplex data stream.
            */
            UV_READABLE_PIPE = 0x10,
            UV_WRITABLE_PIPE = 0x20,
            /*
            * When UV_CREATE_PIPE is specified, specifying UV_NONBLOCK_PIPE opens the
            * handle in non-blocking mode in the child. This may cause loss of data,
            * if the child is not designed to handle to encounter this mode,
            * but can also be significantly more efficient.
            */
            UV_NONBLOCK_PIPE = 0x40
```

```
} uv_stdio_flags;
```

Public members

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 136)

Unknown directive type "c:member".

```
.. c:member:: int uv_process_t.pid

    The PID of the spawned process. It's set after calling :c:func:`uv_spawn`.
```

Note

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 141); *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)process.rst, line 143)

Unknown directive type "c:member".

```
.. c:member:: uv_exit_cb uv_process_options_t.exit_cb

    Callback called after the process exits.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 147)

Unknown directive type "c:member".

```
.. c:member:: const char* uv_process_options_t.file

    Path pointing to the program to be executed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 151)

Unknown directive type "c:member".

```
.. c:member:: char** uv_process_options_t.args

    Command line arguments. args[0] should be the path to the program. On
    Windows this uses `CreateProcess` which concatenates the arguments into a
    string this can cause some strange errors. See the
    ``UV_PROCESS_WINDOWS_VERBATIM_ARGUMENTS`` flag on :c:type:`uv_process_flags`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 158)

Unknown directive type "c:member".

```
.. c:member:: char** uv_process_options_t.env

    Environment for the new process. If NULL the parents environment is used.
```

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

Unknown directive type "c:member".

```
.. c:member:: const char* uv_process_options_t.cwd
```

```
    Current working directory for the subprocess.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 166)

Unknown directive type "c:member".

```
.. c:member:: unsigned int uv_process_options_t.flags

    Various flags that control how :c:func:`uv_spawn` behaves. See
    :c:type:`uv_process_flags`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 171)

Unknown directive type "c:member".

```
.. c:member:: int uv_process_options_t.stdio_count
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 172)

Unknown directive type "c:member".

```
.. c:member:: uv_stdio_container_t* uv_process_options_t.stdio

    The `stdio` field points to an array of :c:type:`uv_stdio_container_t`
    structs that describe the file descriptors that will be made available to
    the child process. The convention is that stdio[0] points to stdin,
    fd 1 is used for stdout, and fd 2 is stderr.

    .. note::
        On Windows file descriptors greater than 2 are available to the child process only if
        the child processes uses the MSVCRT runtime.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 183)

Unknown directive type "c:member".

```
.. c:member:: uv_uid_t uv_process_options_t.uid
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 184)

Unknown directive type "c:member".

```
.. c:member:: uv_gid_t uv_process_options_t.gid

    Libuv can change the child process' user/group id. This happens only when
    the appropriate bits are set in the flags fields.

    .. note::
        This is not supported on Windows, :c:func:`uv_spawn` will fail and set the error
        to ``UV_ENOTSUP``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 193)

Unknown directive type "c:member".

```
.. c:member:: uv_stdio_flags uv_stdio_container_t.flags

    Flags specifying how the stdio container should be passed to the child.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 197)

Unknown directive type "c:member".

```
.. c:member:: union @0 uv_stdio_container_t.data

    Union containing either the `stream` or `fd` to be passed on to the child
```

process.

API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 206)

Unknown directive type "c:function".

```
.. c:function:: void uv_disable_stdio_inheritance(void)

    Disables inheritance for file descriptors / handles that this process
    inherited from its parent. The effect is that child processes spawned by
    this process don't accidentally inherit these handles.

    It is recommended to call this function as early in your program as possible,
    before the inherited file descriptors can be closed or duplicated.

    .. note::
        This function works on a best-effort basis: there is no guarantee that libuv can discover
        all file descriptors that were inherited. In general it does a better job on Windows than
        it does on Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 220)

Unknown directive type "c:function".

```
.. c:function:: int uv_spawn(uv_loop_t* loop, uv_process_t* handle, const uv_process_options_t* optio

    Initializes the process handle and starts the process. If the process is
    successfully spawned, this function will return 0. Otherwise, the
    negative error code corresponding to the reason it couldn't spawn is
    returned.

    Possible reasons for failing to spawn would include (but not be limited to)
    the file to execute not existing, not having permissions to use the setuid or
    setgid specified, or not having enough memory to allocate for the new
    process.

    .. versionchanged:: 1.24.0 Added `UV_PROCESS_WINDOWS_HIDE_CONSOLE` and
                        `UV_PROCESS_WINDOWS_HIDE_GUI` flags.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 235)

Unknown directive type "c:function".

```
.. c:function:: int uv_process_kill(uv_process_t* handle, int signum)

    Sends the specified signal to the given process handle. Check the documentation
    on :c:ref:`signal` for signal support, specially on Windows.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 240)

Unknown directive type "c:function".

```
.. c:function:: int uv_kill(int pid, int signum)

    Sends the specified signal to the given PID. Check the documentation
    on :c:ref:`signal` for signal support, specially on Windows.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\node-master\deps\uv\docs\src\(node-master)(deps)(uv)(docs)(src)process.rst, line 245)

Unknown directive type "c:function".

```
.. c:function:: uv_pid_t uv_process_get_pid(const uv_process_t* handle)

    Returns `handle->pid`.

    .. 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)process.rst, line 251)

Unknown directive type "seealso".

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