Policies

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 1)

Unknown directive type "currentmodule".

```
.. currentmodule:: asyncio
```

An event loop policy is a global per-process object that controls the management of the event loop. Each event loop has a default policy, which can be changed and customized using the policy API.

A policy defines the notion of *context* and manages a separate event loop per context. The default policy defines *context* to be the current thread.

By using a custom event loop policy, the behavior of :func:`get_event_loop`, :func:`set_event_loop`, and :func:`new_event_loop` functions can be customized.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 18); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 18); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 18); *backlink*

Unknown interpreted text role "func".

Policy objects should implement the APIs defined in the :class:`AbstractEventLoopPolicy` abstract base class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 22); *backlink*

Unknown interpreted text role "class".

Getting and Setting the Policy

The following functions can be used to get and set the policy for the current process:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 32)

Unknown directive type "function".

```
.. function:: get_event_loop_policy()

   Return the current process-wide policy.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 36)

Unknown directive type "function".

```
.. function:: set_event_loop_policy(policy)

   Set the current process-wide policy to *policy*.

   If *policy* is set to ``None``, the default policy is restored.
```

Policy Objects

The abstract event loop policy base class is defined as follows:

An abstract base class for asyncio policies.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 52)

Unknown directive type "method".

```
.. method:: get_event_loop()

   Get the event loop for the current context.

   Return an event loop object implementing the
   :class:`AbstractEventLoop` interface.

   This method should never return ``None``.

   .. versionchanged:: 3.6
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 63)

Unknown directive type "method".

```
.. method:: set_event_loop(loop)

   Set the event loop for the current context to *loop*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 67)

Unknown directive type "method".

```
.. method:: new_event_loop()

   Create and return a new event loop object.

   This method should never return ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 73)

Unknown directive type "method".

```
.. method:: get_child_watcher()

   Get a child process watcher object.

   Return a watcher object implementing the
   :class:`AbstractChildWatcher` interface.

   This function is Unix specific.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 82)

Unknown directive type "method".

```
.. method:: set_child_watcher(watcher)

   Set the current child process watcher to *watcher*.

   This function is Unix specific.
```

asyncio ships with the following built-in policies:

The default asyncio policy. Uses :class:`SelectorEventLoop` on Unix and :class:`ProactorEventLoop` on Windows.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 94); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 94); *backlink*

Unknown interpreted text role "class".

There is no need to install the default policy manually. asyncio is configured to use the default policy automatically.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 100)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8

   On Windows, :class:`ProactorEventLoop` is now used by default.
```

An alternative event loop policy that uses the :class:`SelectorEventLoop` event loop implementation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 107); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 110)

Unknown directive type "availability".

```
.. availability:: Windows.
```

An alternative event loop policy that uses the :class:`ProactorEventLoop` event loop implementation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 115); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 118)

Unknown directive type "availability".

```
.. availability:: Windows.
```

Process Watchers

A process watcher allows customization of how an event loop monitors child processes on Unix. Specifically, the event loop needs to know when a child process has exited.

In asyncio, child processes are created with :func:`create_subprocess_exec` and :meth:`loop.subprocess_exec` functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 129); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 129); *backlink*

Unknown interpreted text role "meth".

asyncio defines the :class:`AbstractChildWatcher` abstract base class, which child watchers should implement, and has four different implementations: :class:`ThreadedChildWatcher` (configured to be used by default), :class:`MultiLoopChildWatcher`, :class:`SafeChildWatcher`, and :class:`FastChildWatcher`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 133); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 133); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 133); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 133); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 133); *backlink*

Unknown interpreted text role "class".

See also the :ref:`Subprocess and Threads <asyncio-subprocess-threads>` section.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 139); *backlink*

Unknown interpreted text role "ref".

The following two functions can be used to customize the child process watcher implementation used by the asyncio event loop:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 145)

Unknown directive type "function".

```
.. function:: get_child_watcher()

   Return the current child watcher for the current policy.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 149)

Unknown directive type "function".

```
.. function:: set_child_watcher(watcher)

   Set the current child watcher to *watcher* for the current
   policy.  *watcher* must implement methods defined in the
   :class:`AbstractChildWatcher` base class.
```

Note

Third-party event loops implementations might not support custom child watchers. For such event loops, using :func:`set_child_watcher` might be prohibited or have no effect.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-policy.rst`, line 156); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 162)

Unknown directive type "method".

```
.. method:: add_child_handler(pid, callback, *args)

   Register a new child handler.

   Arrange for ``callback(pid, returncode, *args)`` to be called
   when a process with PID equal to *pid* terminates.  Specifying
   another callback for the same process replaces the previous
   handler.

   The *callback* callable must be thread-safe.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 173)

Unknown directive type "method".

```
.. method:: remove_child_handler(pid)

   Removes the handler for process with PID equal to *pid*.

   The function returns ``True`` if the handler was successfully
   removed, ``False`` if there was nothing to remove.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 180)

Unknown directive type "method".

```
.. method:: attach_loop(loop)

   Attach the watcher to an event loop.

   If the watcher was previously attached to an event loop, then
   it is first detached before attaching to the new loop.

   Note: loop may be ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 189)

Unknown directive type "method".

```
.. method:: is_active()

   Return ``True`` if the watcher is ready to use.

   Spawning a subprocess with *inactive* current child watcher raises
   :exc:`RuntimeError`.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 198)

Unknown directive type "method".

```
.. method:: close()

   Close the watcher.

   This method has to be called to ensure that underlying
   resources are cleaned-up.
```

This implementation starts a new waiting thread for every subprocess spawn.

It works reliably even when the asyncio event loop is run in a non-main OS thread.

There is no noticeable overhead when handling a big number of children (*O(1)* each time a child terminates), but starting a thread per process requires extra memory.

This watcher is used by default.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 216)

Unknown directive type "versionadded".

```
.. versionadded:: 3.8
```

This implementation registers a :py:data:`SIGCHLD` signal handler on instantiation. That can break third-party code that installs a custom handler for :py:data:`SIGCHLD` signal.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 220); *backlink*

Unknown interpreted text role "py:data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 220); *backlink*

Unknown interpreted text role "py:data".

The watcher avoids disrupting other code spawning processes by polling every process explicitly on a :py:data:`SIGCHLD` signal.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 224); *backlink*

Unknown interpreted text role "py:data".

There is no limitation for running subprocesses from different threads once the watcher is installed.

The solution is safe but it has a significant overhead when handling a big number of processes (*O(n)* each time a :py:data:`SIGCHLD` is received).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 230); *backlink*

Unknown interpreted text role "py:data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 234)

Unknown directive type "versionadded".

```
.. versionadded:: 3.8
```

This implementation uses active event loop from the main thread to handle :py:data:`SIGCHLD` signal. If the main thread has no running event loop another thread cannot spawn a subprocess (:exc:`RuntimeError` is raised).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 238); *backlink*

Unknown interpreted text role "py:data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 238); *backlink*

Unknown interpreted text role "exc".

The watcher avoids disrupting other code spawning processes by polling every process explicitly on a :py:data:`SIGCHLD` signal.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 242); *backlink*

Unknown interpreted text role "py:data".

This solution is as safe as :class:`MultiLoopChildWatcher` and has the same *O(N)* complexity but requires a running event loop in the main thread to work.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 245); *backlink*

Unknown interpreted text role "class".

This implementation reaps every terminated processes by calling `os.waitpid(-1)` directly, possibly breaking other code spawning processes and waiting for their termination.

There is no noticeable overhead when handling a big number of children (*O(1)* each time a child terminates).

This solution requires a running event loop in the main thread to work, as :class:`SafeChildWatcher`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 257); *backlink*

Unknown interpreted text role "class".

This implementation polls process file descriptors (pidfds) to await child process termination. In some respects, :class:`PidfdChildWatcher` is a "Goldilocks" child watcher implementation. It doesn't require signals or threads, doesn't interfere with any processes launched outside the event loop, and scales linearly with the number of subprocesses launched by the event loop. The main disadvantage is that pidfds are specific to Linux, and only work on recent (5.3+) kernels.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 262); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 270)

Unknown directive type "versionadded".

```
.. versionadded:: 3.9
```

Custom Policies

To implement a new event loop policy, it is recommended to subclass :class:`DefaultEventLoopPolicy` and override the methods for which custom behavior is wanted, e.g.:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-policy.rst, line 276); *backlink*

Unknown interpreted text role "class".

```
class MyEventLoopPolicy(asyncio.DefaultEventLoopPolicy):

    def get_event_loop(self):
        """Get the event loop.

        This may be None or an instance of EventLoop.
        """
        loop = super().get_event_loop()
        # Do something with loop ...
        return loop

asyncio.set_event_loop_policy(MyEventLoopPolicy())
```