

Migrating off python3-eventlib

Status: proposal. Nothing in this document has been implemented yet.

1. Why eventlib exists

python3-eventlib is AG Projects' fork of eventlet 0.8.x. Denis Bilenko started it in January 2009 as python-eventlet-0.8. It was renamed eventlib in 0.1.0 and ported to Python 3 in 0.3.0. Upstream eventlet merged the Twisted integration in 0.9.0 and later dropped it, so the fork is the only place that integration still lives.

sipsimple depends on these parts of it:

Feature	Where used	Upstream eventlet
hubs/twistedr.py BaseTwistedHub: greenlets scheduled by the Twisted reactor	application.py via twistedutil.join_reactor	removed
twistedutil.callInGreenThread	threading/green.py (run_in_green_thread, call_in_green_thread, Worker)	removed
twistedutil.protocol.GreenTransportBase	msrplib transports	removed
proc (spawn, ProcExit, links, wrap_errors, RunningProcSet)	session, streams, account, lookup, audio	removed
coros.event, coros.queue, coros.Semaphore	threading/green.py Command, most state machines	reshaped/removed
Explicit eventlib.green.*, no monkey- patching	lookup.py (dnspython socket/select), xcap, bonjour	monkey-patching is the norm

Fork-local fixes since then: non-blocking getaddrinfo, GnuTLS-based SSL, Windows fixes, GreenSocket context manager for dnspython >= 2.7, and the hub-stall watchdog (0.3.2).

Runtime model today:

```
main / UI thread (Cocoa, Qt, SylkServer main)
    | reactor.callFromThread / run_in_green_thread
    v
"Reactor Thread" -- twisted reactor.run()
    |           BaseTwistedHub: greenlets switch on reactor
    |           readers/writers/callLater
    v
PJSIP worker threads -> Engine callbacks -> callFromThread
```

Blocking calls are not marked. Any function a green thread calls can block five frames down (proc.wait, coros.event.wait, a green socket read, api.sleep), and that is the property every migration has to deal with.

Inventory: sipsimple has 16 modules importing eventlib; the most common import is `from eventlib import api, coros, proc` (6 sites). msrplib, xcplib, Blink (Cocoa and Qt) and SylkServer also use `proc` and `coros` directly and must be migrated too.

2. Target

Use `asyncio`, written as `async def / await`.

Twisted's `inlineCallbacks` was considered and rejected:

- It keeps us on Twisted only. Twisted itself now recommends `async def` with `Deferred.fromCoroutine / ensureDeferred`.
- `Deferred.cancel()` does not reliably interrupt a generator partway through. sipsimple depends on `proc.kill` interrupting a green thread at its current blocking point. `asyncio` raises `CancelledError` at the `await` point, which has the same semantics.
- `asyncio` primitives map directly onto ours: `coros.event` → `asyncio.Event/Future`, `coros.queue` → `asyncio.Queue`, `proc.spawn` → `asyncio.create_task / TaskGroup`, `api.timeout` → `asyncio.timeout()`, `proc.kill` → `Task.cancel()`.
- Better library support (dnspython `async`, aiohttp, websockets; autobahn in SylkServer supports `asyncio` as well), and better tracebacks.

Either choice requires marking every blocking path as `async`, so this rewrite is the cost no matter which target we pick. The plan below avoids doing it in one go.

3. Migration path

Greenlets stay until the end. First the hub moves under `asyncio`, and then modules move to `async def` one at a time, bridged in both directions.

Phase 1: Twisted on `asyncioreactor`, `eventlib` unchanged

Install `twisted.internet.asyncioreactor` before anything imports the reactor, and run it in the existing Reactor Thread with its own event loop:

```
# application.py, _run_reactor(), before join_reactor
import asyncio
from twisted.internet import asyncio as asyncio_reactor
loop = asyncio.new_event_loop()
asyncio.set_event_loop(loop)
asyncioreactor.install(loop)
```

The reactor must be installed in that thread before the first `from twisted.internet import reactor` anywhere in the process. Several modules import the reactor at module level (`application.py`, `session.py`, `lookup.py`, `util`), so in practice the install must happen at sipsimple import time, or the clients (Blink, SylkServer) must do it at startup. Decide which before starting.

Blocker in eventlib: `hubs/twistedr.py:callLater()` does not call `reactor.callLater`. It builds a `DelayedCall` itself and appends it to `reactor._newTimedCalls`, using `_cancelCallLater / _moveCallLaterSooner`. Those are `ReactorBase` internals. `AsyncioSelectorReactor` overrides

`callLater` and schedules through `loop.call_later`, so it never processes `_newTimedCalls`. Every `schedule_call_local/global` timer (meaning `api.sleep`, timeouts, `coros` with timeouts) would silently never fire.

Fix: reimplement `callLater` in `twistedr.py` on top of the public `reactor.callLater`, keeping the `LocalDelayedCall` behaviour (skip if the owning greenlet is dead) as a wrapper around the returned `IDelayedCall`. The rest of `BaseTwistedHub` uses only public reactor API (`addReader`, `addWriter`, `removeReader`, `getReaders`, `stop`, `crash`, `running`) and should work unchanged.

Exit criteria: the full Blink and SylkServer test cycle passes on `asyncioreactor` (register, presence, audio/video call, MSRP chat and file transfer, XCAP sync, DNS failover, sleep/wake). The default reactor stays switchable via an env var until this is proven.

Phase 2: asyncio-native hub in eventlib

Add `eventlib/hubs/asyncio.py` that drives greenlets from the `asyncio` loop directly (`loop.add_reader/add_writer`, `loop.call_later`). Twisted stays installed on the same loop, so `msrplib` and other Twisted code keep working. Upstream eventlet has done exactly this (`EVENTLET_HUB=asyncio`; see its migration guide) and its implementation can be used as a reference.

Add the bridge APIs to `eventlib`:

- `eventlib.asyncio.spawn_for_awaitable(coro)` returns a `proc` that waits on a coroutine from green code.
- `await proc_instance` / `await coros.event` lets `async` code wait on green code.
- Cancellation propagates both ways: `Task.cancel()` $\leftrightarrow$ `proc.kill()`.

Known limitation (also true upstream): `eventlib` locks, queues and greenlet-locals do not work inside `async def`, and `asyncio` locks do not work inside greenlets. Each shared primitive has to belong to one side.

Phase 3: port leaves to async

Do the leaf modules first, because they have few callers and clear interfaces:

1. `lookup.py`: `dnspython` has a native `asyncio` backend, so the `_green_wait_for` / `dns.query.socket` patching goes away. (`lookup-gevent.py` was an earlier experiment and can be deleted.)
2. XCAP client / `xcaplib`: switch to `aiohttp`.
3. `account/bonjour`.
4. `audio.py`, `util` timers.

Each ported module exposes `async def` APIs and keeps a thin green wrapper (`spawn_for_awaitable(...).wait()`) for callers that are not ported yet.

Phase 4: core state machines

Port `account/registration.py`, `subscription.py` and `publication.py`, then `session.py` and `streams/`. Replace `threading/green.py`:

- Command uses `asyncio.Future` instead of `coros.event`.

- `run_in_green_thread` / `call_in_green_thread` become `asyncio.run_coroutine_threadsafe(coro, loop)` from foreign threads and `loop.create_task` in the loop thread.
- PJSIP callbacks use `loop.call_soon_threadsafe` instead of `reactor.callFromThread`. The two are equivalent once the reactor is on `asyncio`.

Phase 5: msrplib and Twisted removal

Port `msrplib` from `GreenTransportBase` to `asyncio Protocol/streams`. After that Twisted is only used for `reactor.callLater` and a few utilities, and those can go. Then drop `eventlib`, `greenlet` and Twisted as dependencies.

Clients

The public API changes from blocking to coroutines at phase 4. Blink Cocoa, Blink Qt and SylkServer need a matching change for each sipssimple release from that point. Until then the green wrappers keep the old signatures working. Blink Qt can use `qasync` if it ever needs the loop in the UI thread. Today it doesn't, because the loop stays in the Reactor Thread.

4. Risks

- Hidden blocking: code that blocks the loop today blocks it in exactly the same way afterwards. The 0.3.2 watchdog should stay in place, or be replaced by `asyncio debug mode` (`loop.slow_callback_duration`).
- Reactor install ordering (phase 1) across all clients.
- `proc` link semantics (`link_exception`, `RunningProcSet.killall`) have no exact `asyncio` equivalent. Audit each use and map it to `TaskGroup` or `done-callbacks`.
- Thread affinity: `threadable.isInIOThread()` checks must be replaced by a check against the loop's thread.

References

- eventlet migration guide: <https://eventlet.readthedocs.io/en/latest/asyncio/migration.html>
- eventlet asyncio compatibility notes: <https://eventlet.readthedocs.io/en/latest/asyncio/compatibility.html>
- Twisted `asyncreactor`: <https://docs.twisted.org/en/stable/api/twisted.internet.asyncreactor.html>