EOFError

mechanism	logging	handled	true

multiprocessing/connection.py in _recv at line 387 — *Hide 3 more frames*

```
382            while remaining > 0:
383                chunk = read(handle, remaining)
384                n = len(chunk)
385                if n == 0:
386                    if remaining == size:
387                        raise EOFError
388                    else:
389                        raise OSError("got end of file during message")
390                buf.write(chunk)
391                remaining -= n
392            return buf
```

buf	<_io.BytesIO object at 0x7f6fb57b8ae0>
chunk	b''
handle	17
n	0
read	<built-in function read>
remaining	4
self	<multiprocessing.connection.Connection object at 0x7f6fb5731d10>
size	4

multiprocessing/connection.py in _recv_bytes at line 418

```
413                 # Also note we want to avoid sending a 0-length buffer separately,
414                 # to avoid "broken pipe" errors if the other end closed the pipe.
415                 self._send(header + buf)
416
417     def _recv_bytes(self, maxsize=None):
418         buf = self._recv(4)
419         size, = struct.unpack("!i", buf.getvalue())
420         if size == -1:
421             buf = self._recv(8)
422             size, = struct.unpack("!Q", buf.getvalue())
423         if maxsize is not None and size > maxsize:
```

maxsize	None
self	<multiprocessing.connection.Connection object at 0x7f6fb5731d10>

multiprocessing/connection.py in recv at line 254

```
249
250     def recv(self):
251         """Receive a (picklable) object"""
252         self._check_closed()
253         self._check_readable()
254         buf = self._recv_bytes()
255         return _ForkingPickler.loads(buf.getbuffer())
256
257     def poll(self, timeout=0.0):
258         """Whether there is any input available to be read"""
259         self._check_closed()
```

self	<multiprocessing.connection.Connection object at 0x7f6fb5731d10>

multiprocessing/managers.py in _callmethod at line 822

```
817                         threading.current_thread().name)
818             self._connect()
819             conn = self._tls.connection
820
821         conn.send((self._id, methodname, args, kwds))
822         kind, result = conn.recv()
823
824         if kind == '#RETURN':
825             return result
826         elif kind == '#PROXY':
827             exposed, token = result
```

args	`[ 'Sanic-Server-0-0' ]`
conn	`<multiprocessing.connection.Connection object at 0x7f6fb5731d10>`
kwds	`{}`
methodname	`'__getitem__'`
self	`<DictProxy object, typeid 'dict' at 0x7f6fb5a1ef50>`

<string> in __getitem__ at line 2 — In App

args	`[ 'Sanic-Server-0-0' ]`
kwds	`{}`
self	`<DictProxy object, typeid 'dict' at 0x7f6fb5a1ef50>`

sanic/worker/manager.py in _sync_states at line 427 — *Hide 3 more frames*

```
422             return all(acked) and len(acked) == self.num_server
423
424         def _sync_states(self):
425             for process in self.processes:
426                 try:
427                     state = self.worker_state[process.name].get("state")
428                 except KeyError:
429                     process.set_state(ProcessState.TERMINATED, True)
430                     continue
431                 if not process.is_alive():
432                     state = "FAILED" if process.exitcode else "COMPLETED"
```

process	`<sanic.worker.process.WorkerProcess object at 0x7f6fb5a1d810>`
self	`<sanic.worker.manager.WorkerManager object at 0x7f6fb63ba850>`

sanic/worker/manager.py in monitor at line 294

```
289                     cycle = self._poll_monitor()
290                     if cycle is MonitorCycle.BREAK:
291                         break
292                     elif cycle is MonitorCycle.CONTINUE:
293                         continue
294                     self._sync_states()
295                     self._cleanup_non_tracked_workers()
296                 except InterruptedError:
297                     if not OS_IS_WINDOWS:
298                         raise
299                     break
```

cycle	None
self	`<sanic.worker.manager.WorkerManager object at 0x7f6fb63ba850>`

sanic/worker/manager.py in run at line 189

```
184             del self.transient[worker.ident]
185
186         def run(self):
187             """Run the worker manager."""
188             self.start()
189             self.monitor()
190             self.join()
191             self.terminate()
192             self.cleanup()
193
194         def start(self):
```

self	`<sanic.worker.manager.WorkerManager object at 0x7f6fb63ba850>`

sanic/mixins/startup.py in serve at line 1146

```
1141                 primary._manager = manager
1142
1143                 ready = primary.listeners["main_process_ready"]
1144                 trigger_events(ready, loop, primary)
1145
1146                 manager.run()
1147             except ServerKilled:
1148                 exit_code = 1
1149             except BaseException:
1150                 kwargs = primary_server_info.settings
1151                 error_logger.exception(
```

app	`Sanic(name="[redacted]")`
app_loader	`<sanic.worker.loader.AppLoader object at 0x7f6fb79bed90>`

apps	`[`
	`  Sanic(name="[redacted]")`
	`]`

cls	`<class 'sanic.app.Sanic'>`
exit_code	`0`
factory	`None`

primary	`Sanic(name='[redacted]')`
primary_server_info	`ApplicationServerInfo(settings={'protocol': <class 'sanic.server.protocols.http_protocol.HttpProtocol'>, 'host': None, 'port': None, 'version': <HTTP.VERSION_1: 1>, 'sock': <socket.socket fd=14, family=10, type=1, proto=0, laddr=('[redacted]', 8003, 0, 0)>, 'unix': '', 'signal': <sanic.models.server_types.Signal object at 0x7f6fb56b5510>, 'loop': None, 'register_sys_signals': True, 'backlog': 100, 'run_multiple': True}, stage=<ServerStage.STOPPED: 1>, server=None)`
socks	`[ <socket.socket fd=14, family=10, type=1, proto=0, laddr=('[redacted]', 8003, 0, 0)> ]`
sync_manager	`<multiprocessing.managers.SyncManager object at 0x7f6fb5a1f710>`