Infrastructure

Infrastructure runbooks: daemons, email, backups, monitoring

- Email Agent Runbook
- Runbook: LightRAG ingest LaunchAgent fix (MC #10286)
- Runbook: LightRAG ingest LaunchAgent fix (MC #10286)
- Runbook: LightRAG ingest LaunchAgent fix (MC #10286)

Email Agent Runbook

Email Agent Runbook

Service: Email Agent Daemon
Location: `~/system/daemons/email-agent.js`
LaunchAgent: `com.john.email-agent`
Interval: Every 5 minutes (300s)
Last Updated: 2026-04-15

---

1. Architecture

What It Does

The Email Agent is a 24/7 daemon that:

- Fetches unseen emails from **6 IMAP accounts** every 5 minutes
- Classifies emails using VIP bypass → quick filter → Ollama (llama3.1:8b, $0 cost)
- Creates Mission Control tasks for ACTION-worthy emails
- Auto-archives INFO and SPAM emails
- Downloads attachments for CEO-forwarded emails
- Logs all activity to HiveMind and JSONL results

Accounts Monitored

Account Key	Email Address	Bitwarden Vault Name
`john`	john@basicconsulting.no	Email - john@basicconsulting.no
`info`	info@basicconsulting.no	Email - info@basicconsulting.no
`alai`	john@alai.no	Email - john@alai.no
`alem`	alem@alai.no	Email - alem@alai.no
`dev`	dev@alai.no	Email - dev@alai.no
`gmail`	alembasic@gmail.com	Email - alembasic@gmail.com

Classification Pipeline

1. **VIP Bypass:** Emails from CEO/family → forced to `ACTION/high`, label: `CEO FORWARD`
2. **Quick Filter:** Pattern-based detection for OWN emails and known SPAM
3. **Ollama Classification:** Remaining emails sent to local llama3.1:8b model
4. **Circuit Breaker:** Falls back to pattern heuristics if Ollama is down (3 failure threshold)

VIP Senders (CEO Bypass List)

Emails from these addresses **bypass all filters** and are always classified as `ACTION/high` with label `CEO FORWARD`:

- alem@alai.no
- alem@basicconsulting.no
- alem.basic@gmail.com
- alembasic@gmail.com
- sibilabasic@gmail.com (CEO's wife)
- riadbasic007@gmail.com (CEO's brother)

Transport: Himalaya Adapter

The daemon uses `~/system/tools/himalaya-adapter.js`, which wraps the Rust-based `himalaya` CLI (`/opt/homebrew/bin/himalaya`).

Config: `~/.config/himalaya/config.toml` — all 6 accounts configured.

---

2. Credentials

Bitwarden Storage

All email accounts are stored in Bitwarden with vault item names following the pattern: `Email - <address>`.

Gmail Account (Special Configuration)

The Gmail account (`alembasic@gmail.com`) uses **App Password authentication** (not the regular Google account password).

Bitwarden Item: `Email - alembasic@gmail.com`
Custom Fields in Vault:

- `imap_host` = `imap.gmail.com`
- `imap_port` = `993`
- `password` = **App Password** (16-character token from Google)

Himalaya Config

File: `~/.config/himalaya/config.toml`

Contains 6 account blocks with IMAP/SMTP settings. Credentials are loaded from Bitwarden at runtime via `mail-native.js`.

---

3. How to Verify

Is the Daemon Running?

```
launchctl list | grep email-agent
# Expected output: PID + exit status 0
# Example: 12345  0  com.john.email-agent
```

Last Heartbeat (Should Be < 10 Minutes Ago)

```
cat ~/system/logs/email-agent-heartbeat.txt
# Shows timestamp of last successful run
```

Recent Activity Log

```
tail -20 ~/system/logs/email-agent-launchd.log
# Should show recent classification activity like:
# {"timestamp":"2026-04-15T13:49:06.450Z","service":"email-
agent","level":"info","message":"Classifying via Ollama: ..."}
```

Pending Emails (Email Inbox Tool)

```
node ~/system/tools/email-inbox.js pending
# Lists emails waiting for classification or action
```

Daemon Status (Full Details)

```
launchctl print gui/$(id -u)/com.john.email-agent
# Shows full launchd status, last run time, exit codes
```

---

4. Troubleshooting

Problem: Daemon Dead (MODULE_NOT_FOUND Error)

Symptom:

```
tail -20 ~/system/logs/email-agent-launchd-error.log
# Shows: Error: Cannot find module '~/system/tools/himalaya-adapter'
```

Root Cause: The `himalaya-adapter.js` file was accidentally archived or deleted.

Fix:

1. Verify the file exists: `ls -lh ~/system/tools/himalaya-adapter.js`
2. If missing, restore from `~/system/tools/archive/` or Git history
3. Restart the daemon:

   ```
   launchctl unload ~/Library/LaunchAgents/com.john.email-agent.plist
   launchctl load ~/Library/LaunchAgents/com.john.email-agent.plist
   ```

4. Verify restart: `launchctl list | grep email-agent`

Problem: Gmail "Unknown Account" Error

Symptom:

```
Error: Unknown account: gmail. Available: john, info, alai, alem, dev
```

Root Cause: The `gmail` key is missing from the `VAULT_NAMES` object in `~/system/tools/mail-native.js`.

Fix:

1. Open `~/system/tools/mail-native.js`
2. Locate the `VAULT_NAMES` object (around line 20)
3. Add the gmail entry:

```
const VAULT_NAMES = {
  john: 'Email - john@basicconsulting.no',
  info: 'Email - info@basicconsulting.no',
  alai: 'Email - john@alai.no',
  alem: 'Email - alem@alai.no',
  dev: 'Email - dev@alai.no',
  gmail: 'Email - alembasic@gmail.com'  // Add this line
};
```

4. Save and reload daemon

Problem: Gmail Hanging Daemon (High CPU/Memory)

Symptom:

- Multiple overlapping `email-agent` processes running
- 400%+ CPU usage (seen in `top`)
- Email agent not completing runs

Root Cause: Gmail IMAP fetch is hanging indefinitely, causing overlapping daemon instances.

Fix:

1. Identify stuck process:

```
ps aux | grep email-agent
```

2. Kill the stuck process gracefully:

```
kill -QUIT <PID>
# Or if unresponsive:
kill -9 <PID>
```

3. Unload and reload daemon:

```
launchctl unload ~/Library/LaunchAgents/com.john.email-agent.plist
launchctl load ~/Library/LaunchAgents/com.john.email-agent.plist
```

Problem: Vault Credentials Unavailable (Circuit Breaker Triggered)

Symptom:

```
Error: Bitwarden session not available
# Or: Circuit breaker OPEN for account: john
```

Root Cause: Bitwarden CLI session expired or `/tmp/bw-session` is empty.

Fix:

1. Check session file:
   ```
   cat /tmp/bw-session
   # Should contain a session token string
   ```
2. If empty, unlock Bitwarden and regenerate session:
   ```
   bw unlock --raw > /tmp/bw-session
   # Enter master password when prompted
   ```
3. Verify session works:
   ```
   bw get item "Email - john@basicconsulting.no" --session $(cat /tmp/bw-session)
   ```
4. Circuit breaker will reset automatically on next successful run (backoff resets after threshold period)

Problem: Alem's Emails Not Showing as ACTION

Symptom: Emails from CEO are classified as INFO or SPAM instead of ACTION/high.

Root Cause: VIP_SENDERS list is incomplete or outdated.

Fix:

1. Open `~/system/daemons/email-agent.js`
2. Locate the `VIP_SENDERS` array (around line 92)
3. Ensure all Alem's addresses are present:

```
const VIP_SENDERS = [
  'alem@alai.no',
  'alem@basicconsulting.no',
  'alem.basic@gmail.com',
  'alembasic@gmail.com',
  'sibilabasic@gmail.com',
  'riadbasic007@gmail.com'
];
```

4. Save and reload daemon

Problem: Ollama Circuit Breaker Open (Fallback Mode)

Symptom:

```
WARN: Ollama circuit breaker OPEN — using pattern heuristic
```

Root Cause: Ollama service is down or unresponsive (3+ consecutive failures).

Fix:

1. Check Ollama service:
   ```
   curl http://localhost:11434/api/tags
   # Should return JSON list of models
   ```
2. If unresponsive, restart Ollama:
   ```
   brew services restart ollama
   # Or manually:
   ollama serve
   ```
3. Circuit breaker will auto-reset after backoff period (starts at 10s, max 5 minutes)
4. Emails will still be processed using pattern-based heuristics during circuit breaker OPEN state

---

5. Gmail App Password Setup

If the Gmail App Password needs to be regenerated (e.g., after credential rotation or security incident):

1. Go to https://myaccount.google.com/apppasswords (must be logged in as alembasic@gmail.com)
2. Click **Generate**
3. Select app: **Mail**
4. Select device: **Mac** (or custom name like "IMAP Daemon")
5. Copy the 16-character App Password (no spaces)
6. Update Bitwarden:

   ```
   bw get item "Email - alembasic@gmail.com" --session $(cat /tmp/bw-session) | \
     jq '.login.password = "<NEW_APP_PASSWORD>"' | \
     bw encode | \
     bw edit item $(bw get item "Email - alembasic@gmail.com" --session $(cat /tmp/bw-session) | jq -r .id) --session $(cat /tmp/bw-session)
   ```

   Or update manually via Bitwarden web vault.

7. Reload daemon:

   ```
   launchctl unload ~/Library/LaunchAgents/com.john.email-agent.plist
   launchctl load ~/Library/LaunchAgents/com.john.email-agent.plist
   ```

---

6. Key Files and Locations

File	Purpose
`~/system/daemons/email-agent.js`	Main daemon script
`~/system/tools/mail-native.js`	VAULT_NAMES map + credential loader
`~/system/tools/himalaya-adapter.js`	Himalaya CLI wrapper (IMAP/SMTP)
`~/.config/himalaya/config.toml`	Himalaya account configuration
`~/Library/LaunchAgents/com.john.email-agent.plist`	LaunchAgent config (5-minute interval)
`~/system/logs/email-agent-launchd.log`	Daemon stdout log
`~/system/logs/email-agent-launchd-error.log`	Daemon stderr log
`~/system/logs/email-agent-heartbeat.txt`	Last successful run timestamp
`~/system/logs/email-triage-results.jsonl`	JSONL log of all classifications
`/tmp/bw-session`	Bitwarden CLI session token

---

7. Escalation

If the daemon is down for > 30 minutes and troubleshooting steps do not resolve:

1. Check `email-agent-launchd-error.log` for stack traces
2. Capture full logs:

   ```
   tail -100 ~/system/logs/email-agent-launchd.log > /tmp/email-agent-debug.log
   tail -100 ~/system/logs/email-agent-launchd-error.log >> /tmp/email-agent-debug.log
   launchctl print gui/$(id -u)/com.john.email-agent >> /tmp/email-agent-debug.log
   ```

3. Slack alert to `#ops`:

   ```
   node ~/system/tools/slack.js send ops "@john Email Agent daemon DOWN for 30+ minutes.
   Logs: /tmp/email-agent-debug.log"
   ```

4. Fallback: manually check inboxes via webmail until daemon is restored

---

Document Status: □ Production
Owner: John (primary agent)
Last Incident: 2026-02-25 — MODULE_NOT_FOUND (himalaya-adapter archived)
Last Review: 2026-04-15

Runbook: LightRAG ingest LaunchAgent fix (MC #10286)

Overview

This runbook documents the investigation and fix applied to three LightRAG-related LaunchAgents on the ALAI Mac Studio host in MC #10286. The fix was validated by Proveo (Angie Jones) with a PARTIAL verdict: 3 PASS, 1 PARTIAL (AC3), 1 FAIL (AC4 — same-day unverifiable). CF Access root cause is tracked separately in MC #10298.

---

1. Symptom — How to Detect This Failure

These signals indicate the `com.alai.lightrag-outbox-ingest` LaunchAgent is failing silently:

- **Outbox file grows, doc count does not:** `wc -l ~/system/logs/mc-task-outcomes.jsonl` increases after each `mc.js done`, but `curl http://localhost:9621/documents | jq .total` stays flat over days.
- **SQLite checkpoint stops advancing:** `sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT MAX(ingested_at) FROM processed"` returns a timestamp from days ago.
- **Watchdog calendar_err alert:** Daemon-fleet-watchdog fires a `calendar_err_<N>` alert for `com.alai.lightrag-outbox-ingest` or `com.john.lightrag-monitor`.
- **HTTP 302 in error log:** `tail ~/system/logs/lightrag-outbox-ingest.err` shows 302 or redirect errors when posting to `https://lightrag.alai.no/documents/text`.
- **PID column is "-" but daemon is not calendar-scheduled:** `launchctl list | grep lightrag` shows PID="-" with non-zero LastExitStatus for a timer-scheduled daemon (StartInterval) is abnormal; for calendar daemons it is normal between scheduled windows.

---

2. Root Cause

The primary failure was in `com.alai.lightrag-outbox-ingest`:

- The plist `LIGHTRAG_URL` environment variable was set to `https://lightrag.alai.no` (the public Cloudflare-proxied URL).
- CF Access service token was returning HTTP 302 on `POST /documents/text` requests from the local host, causing all upload attempts to time out or silently fail.
- LightRAG itself was healthy at `http://localhost:9621` — this is the correct direct URL for host-local callers.

Workaround applied: Changed `LIGHTRAG_URL` to `http://localhost:9621` in the plist. The CF Access token 302 root cause (why the local host receives a redirect instead of being authorized) is tracked in **MC #10298** (priority: M).

The other two daemons were not functionally broken:

- `com.alai.lightrag-backup`: Calendar Sunday-only schedule. PID="-" between fires is launchd-normal. LastExitStatus=0. No defect.
- `com.john.lightrag-monitor`: exit 256 = bash exit 1 = warnings-only state (Ollama route 302, SSH not configured). These are pre-existing infrastructure gaps, not failures. The script exits 1 to flag warnings; this is by design.

---

3. Fix Procedure

Preconditions: You have shell access to the Mac Studio host. LightRAG is running locally on port 9621.

Step 1: Verify current plist URL

```
grep -A1 "LIGHTRAG_URL" ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
```

If the value is `https://lightrag.alai.no`, proceed. If already `http://localhost:9621`, skip to Step 4.

Step 2: Edit the plist

```
# Open in editor — change the LIGHTRAG_URL string value:
# FROM: https://lightrag.alai.no
# TO:   http://localhost:9621
nano ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
```

The relevant section in the plist:

```
<key>LIGHTRAG_URL</key><string>http://localhost:9621</string>
```

Step 3: Unload all 3 lightrag plists

```
launchctl unload ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
launchctl unload ~/Library/LaunchAgents/com.alai.lightrag-backup.plist
launchctl unload ~/Library/LaunchAgents/com.john.lightrag-monitor.plist
```

Step 4: Reload all 3 lightrag plists

```
launchctl load -w ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
launchctl load -w ~/Library/LaunchAgents/com.alai.lightrag-backup.plist
launchctl load -w ~/Library/LaunchAgents/com.john.lightrag-monitor.plist
```

Step 5: Drain the outbox manually (if backlog exists)

```
node ~/system/tools/lightrag-outbox-ingest.js
```

The script is idempotent — it uses `outbox-ingest.sqlite` with `correlation_id` as PRIMARY KEY dedup gate. Running it multiple times is safe. Expected output when backlog is cleared: `processed: 0, skipped: N, failed: 0`.

Step 6: Kickstart the ingest daemon to verify immediate fire

```
launchctl kickstart -k gui/$(id -u)/com.alai.lightrag-outbox-ingest
```

Check the log immediately after:

```
tail -20 ~/system/logs/lightrag-outbox-ingest.log
```

Expected: A `[ingest] DONE` line with exit success.

Step 7: Confirm watchdog detects healthy state

```
bash ~/bin/daemon-fleet-watchdog.sh 2>&1 | grep lightrag
```

Expected: All 3 labels in `calendar_ok` or `calendar_ok` state. No `calendar_err_*` or `not_loaded` transitions.

---

4. Verification Commands

```
# 1. All 3 plists loaded with LastExitStatus=0
launchctl list | grep lightrag

# 2. Checkpoint DB row count (should match mc-task-outcomes.jsonl line count)
sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT count(*) FROM processed"

# 3. Most recent ingest timestamp
sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT MAX(ingested_at) FROM processed"

# 4. LightRAG pipeline health
curl http://localhost:9621/documents/pipeline_status

# 5. LightRAG document total count
curl http://localhost:9621/documents | jq .total

# 6. Outbox log last run summary
grep "DONE" ~/system/logs/lightrag-outbox-ingest.log | tail -5

# 7. Watchdog recent transitions for lightrag
grep lightrag ~/system/logs/daemon-fleet-watchdog.log | tail -20
```

---

5. Known Limitations

- **AC4 cannot be verified same-day:** `com.alai.lightrag-outbox-ingest` fires on `StartInterval=21600` (6 hours). Verifying that launchd autonomously fires the next scheduled cycle requires waiting at least 6 hours after the kickstart. Same-day verification only demonstrates manual-kickstart success. Rely on the daemon-fleet-watchdog for ongoing health monitoring.
- **Log timestamps absent:** `lightrag-outbox-ingest.js` does not emit timestamps to its log file. This makes it impossible to distinguish manually-triggered runs from launchd-autonomous fires in the log tail. Consider adding `console.log(new Date().toISOString())` at script start (MC #10298 or a follow-up TD).

- **CF Access 302 root cause unresolved:** The public URL `https://lightrag.alai.no` still returns HTTP 302 for host-local service token requests. The localhost bypass is a workaround. If the CF tunnel configuration changes or localhost:9621 changes port, the plist must be updated again. See MC #10298 for the proper fix.
- **com.john.lightrag-monitor DRAFT comment:** The plist still contains a stale "DRAFT — pending Alem approval" comment referencing MC #8545. The daemon IS installed and running. This comment is cosmetic noise but should be cleaned up.
- **AC3 drain was incremental, not single-session:** The 312-entry outbox was drained incrementally across multiple sessions starting 2026-04-17. Any future outbox drain may similarly require multiple passes if entries arrive between runs.

---

6. Watchdog Coverage

The daemon-fleet-watchdog at `~/bin/daemon-fleet-watchdog.sh` covers all 3 LightRAG plists via its glob at line 39:

```
for plist in "$HOME"/Library/LaunchAgents/com.{alai,john}.*.plist
```

This glob automatically includes any new LightRAG LaunchAgents matching the pattern without code changes. The watchdog runs every 15 minutes via `com.alai.daemon-fleet-watchdog`.

Alert states to watch for:

- `calendar_err_256` — daemon exits with code 1 (warnings/errors)
- `calendar_err_512` — daemon exits with code 2 (script error)
- `not_loaded` — plist unloaded from launchd (critical)

Healthy state: `calendar_ok` (LastExitStatus=0, plist loaded)

---

7. Related MCs

MC	Title	Status	Notes
#10286	Fix LightRAG ingest LaunchAgents — drain 312 outbox + add watchdog	DONE (PARTIAL verify)	This fix. Delivered by Kelsey Hightower. Proveo: 3 PASS, 1 PARTIAL, 1 FAIL.
#10298	CF Access service token 302 root cause investigation	OPEN (priority: M)	Why does https://lightrag.alai.no return 302 for local host? Should resolve the need for the localhost bypass.

8. Evidence Links

- Proveo full report: `/tmp/postflight-10286/proveo-report.md`
- Proveo JSON: `/tmp/proveo-10286-1777555315.json`
- Watchdog glob source: `~/bin/daemon-fleet-watchdog.sh:39`
- Plist (fixed): `~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist` — LIGHTRAG_URL=http://localhost:9621
- Checkpoint DB: `~/system/state/outbox-ingest.sqlite` — 312 rows as of 2026-04-30
- Ingest log: `~/system/logs/lightrag-outbox-ingest.log` — 6286 lines, multi-session history since 2026-04-17
- Watchdog log transitions: `~/system/logs/daemon-fleet-watchdog.log` — 12:33:44Z calendar_ok→not_loaded, 12:44:21Z not_loaded→calendar_ok

Runbook: LightRAG ingest LaunchAgent fix (MC #10286)

Overview

This runbook documents the investigation and fix applied to three LightRAG-related LaunchAgents on the ALAI Mac Studio host in MC #10286. The fix was validated by Proveo (Angie Jones) with a PARTIAL verdict: 3 PASS, 1 PARTIAL (AC3), 1 FAIL (AC4 — same-day unverifiable). CF Access root cause is tracked separately in MC #10298.

---

1. Symptom — How to Detect This Failure

These signals indicate the `com.alai.lightrag-outbox-ingest` LaunchAgent is failing silently:

- **Outbox file grows, doc count does not:** `wc -l ~/system/logs/mc-task-outcomes.jsonl` increases after each `mc.js done`, but `curl http://localhost:9621/documents | jq .total` stays flat over days.
- **SQLite checkpoint stops advancing:** `sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT MAX(ingested_at) FROM processed"` returns a timestamp from days ago.
- **Watchdog calendar_err alert:** Daemon-fleet-watchdog fires a `calendar_err_N` alert for `com.alai.lightrag-outbox-ingest` or `com.john.lightrag-monitor`.
- **HTTP 302 in error log:** `tail ~/system/logs/lightrag-outbox-ingest.err` shows 302 or redirect errors when posting to `https://lightrag.alai.no/documents/text`.
- **PID column is "-" with non-zero LastExitStatus:** `launchctl list | grep lightrag` shows PID="-" with non-zero LastExitStatus for a timer-scheduled daemon (StartInterval) is abnormal; for calendar daemons it is normal between scheduled windows.

---

2. Root Cause

The primary failure was in `com.alai.lightrag-outbox-ingest`:

- The plist `LIGHTRAG_URL` environment variable was set to `https://lightrag.alai.no` (the public Cloudflare-proxied URL).
- CF Access service token was returning HTTP 302 on `POST /documents/text` requests from the local host, causing all upload attempts to time out or silently fail.
- LightRAG itself was healthy at `http://localhost:9621` — this is the correct direct URL for host-local callers.

Workaround applied: Changed `LIGHTRAG_URL` to `http://localhost:9621` in the plist. The CF Access token 302 root cause (why the local host receives a redirect instead of being authorized) is tracked in **MC #10298** (priority: M).

The other two daemons were not functionally broken:

- `com.alai.lightrag-backup`: Calendar Sunday-only schedule. PID="-" between fires is launchd-normal. LastExitStatus=0. No defect.
- `com.john.lightrag-monitor`: exit 256 = bash exit 1 = warnings-only state (Ollama route 302, SSH not configured). These are pre-existing infrastructure gaps, not failures. The script exits 1 to flag warnings; this is by design.

---

3. Fix Procedure

Preconditions: You have shell access to the Mac Studio host. LightRAG is running locally on port 9621.

Step 1: Verify current plist URL

```
grep -A1 "LIGHTRAG_URL" ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
```

If the value is `https://lightrag.alai.no`, proceed. If already `http://localhost:9621`, skip to Step 4.

Step 2: Edit the plist

```
nano ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
```

Change the LIGHTRAG_URL string value from `https://lightrag.alai.no` to `http://localhost:9621`. The correct plist line:

```
<key>LIGHTRAG_URL</key><string>http://localhost:9621</string>
```

Step 3: Unload all 3 lightrag plists

```
launchctl unload ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
launchctl unload ~/Library/LaunchAgents/com.alai.lightrag-backup.plist
launchctl unload ~/Library/LaunchAgents/com.john.lightrag-monitor.plist
```

Step 4: Reload all 3 lightrag plists

```
launchctl load -w ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
launchctl load -w ~/Library/LaunchAgents/com.alai.lightrag-backup.plist
launchctl load -w ~/Library/LaunchAgents/com.john.lightrag-monitor.plist
```

Step 5: Drain the outbox manually (if backlog exists)

```
node ~/system/tools/lightrag-outbox-ingest.js
```

The script is idempotent — it uses `outbox-ingest.sqlite` with `correlation_id` as PRIMARY KEY dedup gate. Running it multiple times is safe. Expected output when backlog is cleared: `processed: 0, skipped: N, failed: 0`.

Step 6: Kickstart the ingest daemon to verify immediate fire

```
launchctl kickstart -k gui/$(id -u)/com.alai.lightrag-outbox-ingest
```

Check the log immediately after:

```
tail -20 ~/system/logs/lightrag-outbox-ingest.log
```

Expected: A `[ingest] DONE` line with exit success.

Step 7: Confirm watchdog detects healthy state

```
bash ~/bin/daemon-fleet-watchdog.sh 2>&1 | grep lightrag
```

Expected: All 3 labels in `calendar_ok` state. No `calendar_err_*` or `not_loaded` transitions.

---

4. Verification Commands

```
# 1. All 3 plists loaded with LastExitStatus=0
launchctl list | grep lightrag

# 2. Checkpoint DB row count (should match mc-task-outcomes.jsonl line count)
sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT count(*) FROM processed"

# 3. Most recent ingest timestamp
sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT MAX(ingested_at) FROM processed"

# 4. LightRAG pipeline health
curl http://localhost:9621/documents/pipeline_status

# 5. LightRAG document total count
curl http://localhost:9621/documents | jq .total

# 6. Outbox log last run summary
grep "DONE" ~/system/logs/lightrag-outbox-ingest.log | tail -5

# 7. Watchdog recent transitions for lightrag
grep lightrag ~/system/logs/daemon-fleet-watchdog.log | tail -20
```

---

5. Known Limitations

- **AC4 cannot be verified same-day:** `com.alai.lightrag-outbox-ingest` fires on `StartInterval=21600` (6 hours). Verifying that launchd autonomously fires the next scheduled cycle requires waiting at least 6 hours after the kickstart. Same-day verification only demonstrates manual-kickstart success. Rely on the daemon-fleet-watchdog for ongoing health monitoring.
- **Log timestamps absent:** `lightrag-outbox-ingest.js` does not emit timestamps to its log file. This makes it impossible to distinguish manually-triggered runs from launchd-autonomous fires in the log tail. Consider adding a timestamp at script start as a follow-up TD.
- **CF Access 302 root cause unresolved:** The public URL `https://lightrag.alai.no` still returns HTTP 302 for host-local service token requests. The localhost bypass is a workaround. If the CF tunnel configuration changes or localhost:9621 changes port, the plist must be updated again. See MC #10298 for the proper fix.

- **com.john.lightrag-monitor DRAFT comment:** The plist still contains a stale "DRAFT — pending Alem approval" comment referencing MC #8545. The daemon IS installed and running. This comment is cosmetic noise but should be cleaned up.
- **AC3 drain was incremental, not single-session:** The 312-entry outbox was drained incrementally across multiple sessions starting 2026-04-17. Any future outbox drain may similarly require multiple passes if entries arrive between runs.

---

6. Watchdog Coverage

The daemon-fleet-watchdog at `~/bin/daemon-fleet-watchdog.sh` covers all 3 LightRAG plists via its glob at line 39:

```
for plist in "$HOME"/Library/LaunchAgents/com.{alai,john}.*.plist
```

This glob automatically includes any new LightRAG LaunchAgents matching the pattern without code changes. The watchdog runs every 15 minutes via `com.alai.daemon-fleet-watchdog`.

Alert states to watch for:

- `calendar_err_256` — daemon exits with code 1 (warnings/errors)
- `calendar_err_512` — daemon exits with code 2 (script error)
- `not_loaded` — plist unloaded from launchd (critical)

Healthy state: `calendar_ok` (LastExitStatus=0, plist loaded)

---

7. Related MCs

MC	Title	Status	Notes
#10286	Fix LightRAG ingest LaunchAgents — drain 312 outbox + add watchdog	DONE (PARTIAL verify)	This fix. Delivered by Kelsey Hightower. Proveo: 3 PASS, 1 PARTIAL, 1 FAIL.
#10298	CF Access service token 302 root cause investigation	OPEN (priority: M)	Why does https://lightrag.alai.no return 302 for local host? Resolves the need for the localhost bypass.

---

8. Evidence Links

- Proveo full report: `/tmp/postflight-10286/proveo-report.md`
- Proveo JSON: `/tmp/proveo-10286-1777555315.json`
- Watchdog glob source: `~/bin/daemon-fleet-watchdog.sh:39`
- Plist (fixed): `~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist` — LIGHTRAG_URL=http://localhost:9621
- Checkpoint DB: `~/system/state/outbox-ingest.sqlite` — 312 rows as of 2026-04-30
- Ingest log: `~/system/logs/lightrag-outbox-ingest.log` — 6286 lines, multi-session history since 2026-04-17
- Watchdog log transitions: `~/system/logs/daemon-fleet-watchdog.log` — 12:33:44Z calendar_ok to not_loaded, 12:44:21Z not_loaded to calendar_ok

Runbook: LightRAG ingest LaunchAgent fix (MC #10286)

Overview

This runbook documents the investigation and fix applied to three LightRAG-related LaunchAgents on the ALAI Mac Studio host in MC #10286. The fix was validated by Proveo (Angie Jones) with a PARTIAL verdict: 3 PASS, 1 PARTIAL (AC3), 1 FAIL (AC4 — same-day unverifiable). CF Access root cause is tracked separately in MC #10298.

---

1. Symptom — How to Detect This Failure

These signals indicate the `com.alai.lightrag-outbox-ingest` LaunchAgent is failing silently:

- **Outbox file grows, doc count does not:** `wc -l ~/system/logs/mc-task-outcomes.jsonl` increases after each `mc.js done`, but `curl http://localhost:9621/documents | jq .total` stays flat over days.
- **SQLite checkpoint stops advancing:** `sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT MAX(ingested_at) FROM processed"` returns a timestamp from days ago.
- **Watchdog calendar_err alert:** Daemon-fleet-watchdog fires a `calendar_err_N` alert for `com.alai.lightrag-outbox-ingest` or `com.john.lightrag-monitor`.
- **HTTP 302 in error log:** `tail ~/system/logs/lightrag-outbox-ingest.err` shows 302 or redirect errors when posting to `https://lightrag.alai.no/documents/text`.
- **PID column is "-" with non-zero LastExitStatus:** `launchctl list | grep lightrag` shows PID="-" with non-zero LastExitStatus for a timer-scheduled daemon (StartInterval) is abnormal; for calendar daemons it is normal between scheduled windows.

---

2. Root Cause

The primary failure was in `com.alai.lightrag-outbox-ingest`:

- The plist `LIGHTRAG_URL` environment variable was set to `https://lightrag.alai.no` (the public Cloudflare-proxied URL).
- CF Access service token was returning HTTP 302 on `POST /documents/text` requests from the local host, causing all upload attempts to time out or silently fail.
- LightRAG itself was healthy at `http://localhost:9621` — this is the correct direct URL for host-local callers.

Workaround applied: Changed `LIGHTRAG_URL` to `http://localhost:9621` in the plist. The CF Access token 302 root cause (why the local host receives a redirect instead of being authorized) is tracked in **MC #10298** (priority: M).

The other two daemons were not functionally broken:

- `com.alai.lightrag-backup`: Calendar Sunday-only schedule. PID="-" between fires is launchd-normal. LastExitStatus=0. No defect.
- `com.john.lightrag-monitor`: exit 256 = bash exit 1 = warnings-only state (Ollama route 302, SSH not configured). These are pre-existing infrastructure gaps, not failures. The script exits 1 to flag warnings; this is by design.

---

3. Fix Procedure

Preconditions: You have shell access to the Mac Studio host. LightRAG is running locally on port 9621.

Step 1: Verify current plist URL

```
grep -A1 "LIGHTRAG_URL" ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
```

If the value is `https://lightrag.alai.no`, proceed. If already `http://localhost:9621`, skip to Step 4.

Step 2: Edit the plist

```
nano ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
```

Change the LIGHTRAG_URL string value from `https://lightrag.alai.no` to `http://localhost:9621`. The correct plist line:

```
<key>LIGHTRAG_URL</key><string>http://localhost:9621</string>
```

Step 3: Unload all 3 lightrag plists

```
launchctl unload ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
launchctl unload ~/Library/LaunchAgents/com.alai.lightrag-backup.plist
launchctl unload ~/Library/LaunchAgents/com.john.lightrag-monitor.plist
```

Step 4: Reload all 3 lightrag plists

```
launchctl load -w ~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist
launchctl load -w ~/Library/LaunchAgents/com.alai.lightrag-backup.plist
launchctl load -w ~/Library/LaunchAgents/com.john.lightrag-monitor.plist
```

Step 5: Drain the outbox manually (if backlog exists)

```
node ~/system/tools/lightrag-outbox-ingest.js
```

The script is idempotent — it uses `outbox-ingest.sqlite` with `correlation_id` as PRIMARY KEY dedup gate. Running it multiple times is safe. Expected output when backlog is cleared: `processed: 0, skipped: N, failed: 0`.

Step 6: Kickstart the ingest daemon to verify immediate fire

```
launchctl kickstart -k gui/$(id -u)/com.alai.lightrag-outbox-ingest
```

Check the log immediately after:

```
tail -20 ~/system/logs/lightrag-outbox-ingest.log
```

Expected: A `[ingest] DONE` line with exit success.

Step 7: Confirm watchdog detects healthy state

```
bash ~/bin/daemon-fleet-watchdog.sh 2>&1 | grep lightrag
```

Expected: All 3 labels in `calendar_ok` state. No `calendar_err_*` or `not_loaded` transitions.

---

4. Verification Commands

```
# 1. All 3 plists loaded with LastExitStatus=0
launchctl list | grep lightrag

# 2. Checkpoint DB row count (should match mc-task-outcomes.jsonl line count)
sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT count(*) FROM processed"

# 3. Most recent ingest timestamp
sqlite3 ~/system/state/outbox-ingest.sqlite "SELECT MAX(ingested_at) FROM processed"

# 4. LightRAG pipeline health
curl http://localhost:9621/documents/pipeline_status

# 5. LightRAG document total count
curl http://localhost:9621/documents | jq .total

# 6. Outbox log last run summary
grep "DONE" ~/system/logs/lightrag-outbox-ingest.log | tail -5

# 7. Watchdog recent transitions for lightrag
grep lightrag ~/system/logs/daemon-fleet-watchdog.log | tail -20
```

---

5. Known Limitations

- **AC4 cannot be verified same-day:** `com.alai.lightrag-outbox-ingest` fires on `StartInterval=21600` (6 hours). Verifying that launchd autonomously fires the next scheduled cycle requires waiting at least 6 hours after the kickstart. Same-day verification only demonstrates manual-kickstart success. Rely on the daemon-fleet-watchdog for ongoing health monitoring.
- **Log timestamps absent:** `lightrag-outbox-ingest.js` does not emit timestamps to its log file. This makes it impossible to distinguish manually-triggered runs from launchd-autonomous fires in the log tail. Consider adding a timestamp at script start as a follow-up TD.
- **CF Access 302 root cause unresolved:** The public URL `https://lightrag.alai.no` still returns HTTP 302 for host-local service token requests. The localhost bypass is a workaround. If the CF tunnel configuration changes or localhost:9621 changes port, the plist must be updated again. See MC #10298 for the proper fix.

- **com.john.lightrag-monitor DRAFT comment:** The plist still contains a stale "DRAFT — pending Alem approval" comment referencing MC #8545. The daemon IS installed and running. This comment is cosmetic noise but should be cleaned up.
- **AC3 drain was incremental, not single-session:** The 312-entry outbox was drained incrementally across multiple sessions starting 2026-04-17. Any future outbox drain may similarly require multiple passes if entries arrive between runs.

---

6. Watchdog Coverage

The daemon-fleet-watchdog at `~/bin/daemon-fleet-watchdog.sh` covers all 3 LightRAG plists via its glob at line 39:

```
for plist in "$HOME"/Library/LaunchAgents/com.{alai,john}.*.plist
```

This glob automatically includes any new LightRAG LaunchAgents matching the pattern without code changes. The watchdog runs every 15 minutes via `com.alai.daemon-fleet-watchdog`.

Alert states to watch for:

- `calendar_err_256` — daemon exits with code 1 (warnings/errors)
- `calendar_err_512` — daemon exits with code 2 (script error)
- `not_loaded` — plist unloaded from launchd (critical)

Healthy state: `calendar_ok` (LastExitStatus=0, plist loaded)

---

7. Related MCs

MC	Title	Status	Notes
#10286	Fix LightRAG ingest LaunchAgents — drain 312 outbox + add watchdog	DONE (PARTIAL verify)	This fix. Delivered by Kelsey Hightower. Proveo: 3 PASS, 1 PARTIAL, 1 FAIL.
#10298	CF Access service token 302 root cause investigation	OPEN (priority: M)	Why does https://lightrag.alai.no return 302 for local host? Resolves the need for the localhost bypass.

---

8. Evidence Links

- Proveo full report: `/tmp/postflight-10286/proveo-report.md`
- Proveo JSON: `/tmp/proveo-10286-1777555315.json`
- Watchdog glob source: `~/bin/daemon-fleet-watchdog.sh:39`
- Plist (fixed): `~/Library/LaunchAgents/com.alai.lightrag-outbox-ingest.plist` — LIGHTRAG_URL=http://localhost:9621
- Checkpoint DB: `~/system/state/outbox-ingest.sqlite` — 312 rows as of 2026-04-30
- Ingest log: `~/system/logs/lightrag-outbox-ingest.log` — 6286 lines, multi-session history since 2026-04-17
- Watchdog log transitions: `~/system/logs/daemon-fleet-watchdog.log` — 12:33:44Z calendar_ok to not_loaded, 12:44:21Z not_loaded to calendar_ok