Email-Reactor — Strategic-Inbox Auto-Triage Daemon

Email-Reactor — Strategic-Inbox Auto-Triage Daemon

Why It Exists

Incident: 2026-05-26 — CEO had to *phone* Asmir Merdžanović to learn that Asmir sent critical SEO partnership email three days earlier (email #8421, dated 2026-05-24). This email sat in the database with status 'new' for 72+ hours while we continued building the exact SEO automation partnership Asmir was offering.

> "Niko ne cita i reaguje na mailove. Ovo smo probali vec 4 mjeseca da odradimo. Ako ne uspijemo mozemo zatvorit firmu."
> — CEO Alem Basic, 2026-05-26, after discovering the Asmir email gap

Previous email systems (email-agent, email-briefing, inbox-queue) classified and queued but **no human acted on them**. Email-Reactor solves this by implementing a 3-step security-first pipeline that creates Mission Control tasks with macOS push notifications for revenue-critical emails automatically.

What It Does

Email-Reactor is a daemon that polls `~/system/databases/email-inbox.db` every 5 minutes (via LaunchAgent `no.alai.inbox-watcher`) and processes every new email through a 3-step pipeline:

1. **SECURITY SCAN** (always first) — rule-based phishing/macro/spoof detection → quarantine on fail
2. **KNOWN-CONTACT CHECK** — parallel lookup in Paperless archive.alai.no correspondents + DB email history → if KNOWN, create MC task + push notification

3. **LLM REVENUE CLASSIFIER** (unknown senders only) — Qwen2.5-Coder 32B asks "Is this revenue-relevant?" → YES = MC task + push, NO = queue silently

Strategic override: VIP senders in `~/system/config/strategic-partners.json` skip all steps and go straight to MC + push (tier-1 phone-grade urgency).

Architecture

```
flowchart LR
    A[Email arrives in DB] --> B{Strategic Partner?}
    B -- YES --> Z[Create MC + Push]
    B -- NO --> C[STEP 1: Security Scan]
    C -- FAIL --> Q[Quarantine + Alert]
    C -- PASS --> D{STEP 2: Known Contact?}
    D -- YES
Paperless/DB --> Z
    D -- NO --> E{Newsletter/Transactional?}
    E -- YES --> N[No MC — Audit as llm_no]
    E -- NO --> F[STEP 3: LLM Classifier]
    F -- YES --> Z
    F -- NO --> N
    Q --> X[STOP]
    N --> X
    Z --> X[Done]
```

Components

Component	Path	Purpose
Watcher daemon	`~/system/tools/inbox-watcher.js`	738-line Node.js script, runs every 5 min
LaunchAgent	`~/Library/LaunchAgents/no.alai.inbox-watcher.plist`	Schedules daemon (StartInterval=300s)
Email DB	`~/system/databases/email-inbox.db`	SQLite, emails table, mc_task_id linkage
Strategic allowlist	`~/system/config/strategic-partners.json`	VIP senders (tier-1 = phone-grade), hot-reloaded
Audit log	`~/system/state/inbox-watcher-audit.log`	JSONL, every action: linked/llm_yes/llm_no/quarantine

Component	Path	Purpose
Quarantine log	`~/system/state/inbox-watcher-quarantine.jsonl`	Security failures, phishing attempts
Ops watchdog	`~/system/config/ops-watchdog.json`	Lists no.alai.inbox-watcher in critical_services
Mission Control	`~/system/tools/mc.js`	Task creation, dedup detection, linkage

Routing Logic Detail

Step 1: Security Scan

Rule-based checks (no LLM cost):

- **Phishing keywords:** "urgent password", "verify account", "bitcoin transfer", "lottery winner", "tax refund"
- **Suspicious URLs:** unencrypted (http://), TLDs (.tk, .ml, .ga, .cf)
- **Macro attachment hints:** .docm, .xlsm, .scr, .exe, .lnk, .msi
- **Domain spoofing:** sender name claims "PayPal" but email is @gmail.com

On failure: email goes to `inbox-watcher-quarantine.jsonl`, audit log records `security_quarantine`, processing STOPS (no MC, no push).

Step 2: Known-Contact Check

Parallel signals (first match wins):

1. **Strategic override:** email matches `strategic-partners.json` (Asmir, SnowIT, paying clients) → immediate MC + push
2. **Paperless Correspondents:** HTTPS GET to `https://archive.alai.no/api/correspondents/` with Bitwarden token + Cloudflare Access headers, searches by domain + sender name → if found, contact is KNOWN
3. **DB email history:** SQL query `SELECT COUNT(*) FROM emails WHERE to_addr LIKE '%sender%' AND classification='OWN'` → if we ever emailed this person, they're KNOWN

If KNOWN via *any* signal: create MC task, fire macOS push notification, audit log records source (override/paperless/db).

Step 3: LLM Revenue Classifier (unknown senders only)

Pre-filter heuristic (saves LLM tokens): detect obvious newsletters/transactional via regex patterns:

- **Transactional senders:** no-reply@, noreply@, notification@, alert@, billing@, invoice@, receipt@, kontakt@fiken, support@stripe
- **Newsletter senders:** newsletter@, digest@, news@, marketing@, promo@, tldr, naeringsliv, mail-list
- **Digest subject lines:** "This week in", "Your weekly digest", "Daily digest", "Unsubscribe here", "View in browser", "Automated notification"

If heuristic matches: audit as `llm_no` with reason `newsletter_heuristic` or `transactional_heuristic`, no MC, STOP.

LLM call (if heuristic passes):

- Endpoint: `http://10.0.0.2:11435/v1/chat/completions` (MLX server on FORGE)
- Model: `mlx-community/Qwen2.5-Coder-32B-Instruct-4bit` (non-reasoning instruct model)
- Timeout: 15 seconds
- Prompt: "Is this a business opportunity, paying client request, partner inquiry, invoice, contract, or revenue-relevant? Answer YES or NO."
- Temperature: 0.3 (0.1 on retry)
- Max tokens: 32 (sufficient for terse YES/NO)
- Response parsing: strict regex `^YES$|^NO$` — malformed = retry once with stricter prompt
- Default on error/timeout: **NO** (conservative fail-safe — real opportunities arrive via KNOWN-CONTACT path)

YES → create MC task + push + audit `llm_yes`
NO → audit `llm_no`, no MC

LLM Classifier Fix — 2026-06-22 (MC #102113)

Deployed live: 2026-06-22T08:49:43Z

Bugs fixed:

1. **Wrong model ID:** Code referenced `gemma-4` which does not exist on FORGE MLX (11435) → HTTP 401 "Repository Not Found". Every LLM call failed and defaulted to NO.
2. **Reasoning model + truncation:** `gemma-4-26b` is a reasoning model that returns thinking in `.message.reasoning` and leaves `.message.content` null until reasoning completes. Code read `.content` with `max_tokens: 5` → answer never landed → classifier always defaulted NO → **unknown-sender revenue leads silently**

dropped.

Fix:

- Switched to FORGE MLX endpoint `10.0.0.2:11435` (was already correct)
- Model: `mlx-community/Qwen2.5-Coder-32B-Instruct-4bit` (non-reasoning instruct model)
- `max_tokens: 32` (up from 5, sufficient for terse YES/NO with margin)
- Reads `.choices[0].message.content` (standard OpenAI format)

Verification (3 independent layers, all 5/5 acceptance):

1. AgentForge build run: 4/5 LLM + case1 (GitHub CI) caught by upstream noise filter = 5/5 production
2. John independent curl re-run: newsletter NO, Fiken NO, cold-lead YES, Asmir YES; GitHub CI caught by `/^notification[s]?[-.@]/i`
3. Proveo independent QA (P2P): PASS — md5 unchanged pre-swap, syntax OK, diff logic-equivalent, 5/5 twice

Live deploy:

- Backup: `~/system/tools/inbox-watcher.js.bak-102113-20260622-084943` (md5 `47192c122a42de14eda9c2305016e420`)
- Live file: md5 `ddd6c98c4af2b0e745594e05a7474f6e`
- Daemon: `no.alai.inbox-watcher` loaded, StartInterval 300s (wrapper re-execs each cycle, picks up swapped file automatically)

Known issues:

- FORGE Ollama 11434 stalled (separate task) — classifier uses 11435 MLX instead
- Intentional fail-OPEN on `req.on("error")` (MC #103835): if 11435 dies, unknown mail creates tasks (noise) rather than dropping leads — by design tradeoff

Evidence:

- `/tmp/evidence-102113/DEPLOY-RECORD-20260622.md` (deploy record)
- `/tmp/evidence-102113/CLASSIFIER-BUG-DIAGNOSIS-20260622.md` (root cause)
- `/tmp/evidence-102113/proveo-verify-102113.md` (independent QA verdict PASS)
- `/tmp/evidence-102113/fix-dry-run-results.md` (acceptance 5/5)

Push Path — Live State (MC #102077, 2026-06-08)

> **Status: WIRED + PROVEO PASS** — Push path activated 2026-06-08. Validated by Proveo (Angie Jones lens). Proveo validation SHA256: `d1f4999b`.

Push Channel

All partner/reactor pushes go to **Slack #ceo** via:

```
node ~/system/tools/slack.js send ceo "<message>"
```

Note: There is no mm-bridge and no macOS push-notification for this path. The channel is exclusively Slack #ceo. The existing stale-SLA escalation in `email-agent.js` (~line 1394) also pushes #ceo for all ACTION emails at 24h/48h/72h/96h thresholds — that path is unchanged.

Allowlist — strategic-partners.json

File: `~/system/config/strategic-partners.json`

Structure:

```
{
  "senders": [
    {
      "email": "asmirmc@gmail.com",
      "name": "Asmir Merdžanović",
      "tier": 1,
      "reason": "SEO partnership lead — tier-1 priority"
    }
  ],
  "domains": []
}
```

Matching rules (in `matchStrategicPartner(fromAddr)`):

- Exact email match (case-insensitive) against `senders[].email`
- Domain suffix match against `domains[]` entries

Current allowlist (as of 2026-06-08): `asmirmc@gmail.com` (Asmir Merdžanović, tier-1). Test senders removed by Proveo after validation.

How to Add a Strategic Partner

1. Open `~/system/config/strategic-partners.json`
2. Append a new object to the `senders` array:

```
{
  "email": "partner@company.no",
  "name": "Partner Name",
  "tier": 1,
  "reason": "Business reason — e.g., paying client, key integration partner"
}
```

3. Save the file. **No daemon reload needed** — `loadStrategicPartners()` reads the file fresh on every ingest cycle.
4. To add a whole domain: append to the `domains` array instead (e.g., `"snowit.no"`).

Trigger and Ingest Path

The push fires inside `~/system/daemons/email-agent.js` at the ingest insert path (line ~2393):

1. New email row inserted into `email-inbox.db` (id assigned)
2. If `dbCategory === 'ACTION'` and not `--dryRun`: calls `matchStrategicPartner(fromAddr)`
3. If match found: calls `setPartnerTier(id, tier)` (sets `partner_tier` column) then `fireReactorPush()`
4. `fireReactorPush()` checks `row.reactor_pushed_at` — if already set, skips (dedup gate)
5. Push fires: `node slack.js send ceo "[TIER-1 PARTNER] <name> emailed <account> — ..."`
6. On success: calls `markReactorPushed(id, tier)` which sets `reactor_pushed_at = NOW()`
7. Rate-limit: at most 10 pushes per daemon cycle (`REACTOR_CYCLE_LIMIT = 10`, tracked via `reactorPushedThisCycle` Set)

Schema Additions (email-inbox.db emails table)

Column	Type	Default	Purpose
`partner_tier`	INTEGER	0	0 = not a partner; 1+ = tier level from allowlist
`reactor_pushed_at`	TEXT	NULL	ISO timestamp of first push; NULL = not yet pushed; set = dedup gate (no re-push)

Indexes: `idx_emails_partner_tier`, `idx_emails_reactor_pushed`

New helper functions exported from `email-inbox.js`:

- `markReactorPushed(id, tier)` — sets both `partner_tier` and `reactor_pushed_at`

- `setPartnerTier(id, tier)` — sets `partner_tier` only (used at ingest time before push)
- `getReactorPending(hoursThreshold)` — returns ACTION emails from partner/high-priority senders unanswered longer than N hours (used by digest)

Daily Digest

File: `~/system/tools/email-reactor-digest.js`

LaunchAgent: `~/Library/LaunchAgents/com.john.email-reactor-digest.plist` (fires daily at 08:00 local)

Behaviour:

- Calls `getReactorPending(6)` — finds ACTION emails from partners OR high-priority senders that are unanswered for more than 6 hours
- Formats two sections: Strategic Partner Emails / High-Priority Emails
- Pushes a single digest message to Slack #ceo
- Same-day dedup: state file `~/system/logs/email-reactor-digest-state.json` stores `last_sent_date`; skips if already sent today unless `--force` is passed

Manual usage:

```
# Dry run (no push, shows what would be sent)
node ~/system/tools/email-reactor-digest.js --dry-run

# Force re-send even if already sent today
node ~/system/tools/email-reactor-digest.js --force

# Check LaunchAgent
launchctl list | grep email-reactor-digest
```

Dedup — Three Independent Layers

Layer	Mechanism	Scope
1. Ingest cycle Set	`reactorPushedThisCycle` (in-memory Set, cleared each cycle)	Within a single 5-min daemon run
2. DB timestamp	`reactor_pushed_at` column — if set, `fireReactorPush()` returns immediately	Permanent — survives restarts
3. Digest date file	`last_sent_date` in `email-reactor-digest-state.json`	Once per calendar day

Proveo Validation Evidence (2026-06-08)

Check	Result	Notes
email-inbox.js columns + helpers	PASS	Syntax OK; exports confirmed; SHA256 `39f67c25`
email-agent.js reactor wired into insert path	PASS	Syntax OK; line 2393 confirmed; SHA256 `f27fc932`
email-reactor-digest.js exists	PASS	6215 bytes; syntax OK; SHA256 `6e63a2e9`
LaunchAgent loaded (launchctl)	PASS	`com.john.email-reactor-digest` active; StartCalendarInterval Hour=8
Push fired to #ceo (independent test)	PASS	Receipt: ✓ Sent to #ceo (Proveo row id=9218)
Dedup — reactor_pushed_at set, no re-push	PASS	Second cycle skips; confirmed via code + DB
Digest push to #ceo	PASS	50 items; Receipt: ✓ Sent to #ceo
Digest same-day dedup	PASS	"Already sent today — skipping"
19-account ingest not regressed	PASS	COUNT(email_accounts)=19; all last_checked 2026-06-08
Test senders cleaned from allowlist	PASS	Only asmirmc@gmail.com remains; SHA256 `289922b8`
No push storm	PASS	3 independent dedup layers confirmed

Overall Proveo verdict: **PASS**. Blocker items: none.

Audit Log Codes

Action	Meaning	MC Created?
`linked`	Known contact, MC task created (first time)	YES
`relinked_via_dedup`	Duplicate MC task found, linked to existing (no new push)	NO (existing)
`security_quarantine`	Failed security scan (phishing/macro/spoof)	NO
`llm_yes`	LLM classified as revenue-relevant	YES
`llm_no`	LLM classified as NOT revenue-relevant (or heuristic match)	NO

Action	Meaning	MC Created?
`newsletter_heuristic`	Pre-LLM heuristic detected newsletter/digest	NO
`transactional_heuristic`	Pre-LLM heuristic detected automated notification/billing	NO
`dry_run`	--dry-run mode, would have created MC	NO (test mode)
`create_failed`	mc.js add command failed	NO (error)
`update_failed`	DB update (mc_task_id linkage) failed	YES (orphaned)

Debug Runbook

Query Audit Log

```
# Last 50 actions
tail -50 ~/system/state/inbox-watcher-audit.log | jq .

# Count actions by type (last 24h)
grep "$(date -u +%Y-%m-%d)" ~/system/state/inbox-watcher-audit.log | \
  jq -r .action | sort | uniq -c | sort -rn

# Find specific email
grep '"email_id":8421' ~/system/state/inbox-watcher-audit.log | jq .
```

Query Quarantine Log

```
# Show all quarantined emails
cat ~/system/state/inbox-watcher-quarantine.jsonl | jq .

# Count by reason
cat ~/system/state/inbox-watcher-quarantine.jsonl | jq -r .reason | sort | uniq -c
```

Check Reactor Push State

```
# All emails that were partner-pushed
sqlite3 ~/system/databases/email-inbox.db \
```

```
  "SELECT id, from_addr, subject, partner_tier, reactor_pushed_at FROM emails WHERE
partner_tier > 0 ORDER BY reactor_pushed_at DESC LIMIT 20;"

# Pending reactor pushes (ACTION emails from partners not yet pushed)
sqlite3 ~/system/databases/email-inbox.db \
  "SELECT id, from_addr, subject, classification FROM emails WHERE partner_tier > 0 AND
reactor_pushed_at IS NULL;"

# Digest state (last sent date)
cat ~/system/logs/email-reactor-digest-state.json
```

Manual Trigger (Dry-Run)

```
node ~/system/tools/inbox-watcher.js --dry-run
```

Shows what *would* happen without creating tasks or updating DB.

Manual Trigger (Live)

```
node ~/system/tools/inbox-watcher.js
```

Check Daemon Status

```
launchctl list | grep inbox-watcher
launchctl list | grep email-reactor-digest
```

Expected output: `no.alai.inbox-watcher` with recent PID; `com.john.email-reactor-digest` with PID `-` (correct for CalendarInterval — fires at 08:00 only).

Restart Daemon

```
launchctl unload ~/Library/LaunchAgents/no.alai.inbox-watcher.plist
launchctl load ~/Library/LaunchAgents/no.alai.inbox-watcher.plist
```

Tail Daemon Logs

```
tail -f ~/system/logs/inbox-watcher.out.log
tail -f ~/system/logs/inbox-watcher.err.log
tail -f ~/system/logs/email-reactor-digest.log
```

Check Email DB for Pending

```
sqlite3 ~/system/databases/email-inbox.db <<EOF
SELECT id, from_addr, subject, status, created_at
FROM emails
WHERE mc_task_id IS NULL
  AND status = 'new'
  AND created_at > datetime('now', '-7 days')
ORDER BY created_at DESC
LIMIT 20;
EOF
```

Failure Modes & Alerts

Failure	Symptom	Alert Mechanism	Recovery
Daemon crash	`launchctl list` shows no PID	ops-watchdog auto-restart (critical_services)	Auto (watchdog), or manual reload plist
Paperless 401	Log shows "HTTP 401"	WARN in out.log, no Slack (non-blocking)	Refresh Bitwarden /tmp/bw-session token
Ollama FORGE down	LLM timeout 15s	Log WARN, defaults to NO (safe)	SSH to FORGE, restart Ollama service
MC duplicate flood	Many relinked_via_dedup in audit	None (expected behavior)	Normal — dedup prevents task spam
DB locked	SQLite BUSY error	ERROR in err.log	Wait 5min (next cycle), or restart daemon
Strategic override miss	VIP email not getting Slack push	CEO notices delay	Verify strategic-partners.json email exact match (case-insensitive); check reactor_pushed_at not already set from an old test row
Slack push fails	No receipt in logs; no #ceo message	WARN in email-agent.log	Check slack.js connectivity; verify Slack token in config

Failure	Symptom	Alert Mechanism	Recovery
Digest not firing at 08:00	No digest in #ceo after 08:10	None (silent)	Run manually: `node ~/system/tools/email-reactor-digest.js --force`; check plist loaded via launchctl

Known Limitations

1. **LLM is safety net, not primary path.** Real opportunities should arrive via KNOWN-CONTACT (Paperless correspondents + DB history). LLM classifier is conservative: defaults to NO on error to avoid false-positive task spam. If a genuine new opportunity is missed by LLM, it will appear in email DB and CEO can manually promote to MC.
2. **Paperless lookup is best-effort.** If Bitwarden token expires or Cloudflare Access headers are missing, Paperless signal fails silently and daemon falls back to DB-history-only KNOWN check. This is by design (non-blocking).
3. **Default NO on malformed LLM response.** Policy changed 2026-05-26 after 6 false positives from verbose LLM responses. Strict regex parsing + retry ensures only clean YES/NO answers create tasks. This may miss 1 real opportunity but prevents 6 noise tasks.
4. **No auto-reply generation.** Out of scope for Phase 2. Email-Reactor creates MC tasks; human writes replies.
5. **30-day recency filter.** Only processes emails from last 30 days to avoid re-scanning old newsletter backlog every 5-min cycle. Older emails must be manually triaged.
6. **Single-account scope.** Currently queries all accounts in email-inbox.db, but strategic-partners.json does not differentiate by account. Future: add account-specific allowlists if needed.
7. **Reactor push is email-agent ingest only.** The push fires on fresh ingest in email-agent.js. It does NOT retroactively push emails already in the DB from before MC #102077. Historical partner emails must be found via digest or manual DB query.

References

- **MC #102077** — Push path wiring (Slack #ceo via slack.js) — COMPLETE 2026-06-08
- **MC #102113** — LLM classifier fix (model + token budget) — DEPLOYED LIVE 2026-06-22
- **Incident email:** #8421 (Asmir Merdžanović, 2026-05-24)
- **Peer review:** /tmp/alai/p2p-pairing-evidence/mesh-thr-102113-peer-ask.md
- **Build evidence:** /tmp/evidence-102077/flowforge-build.md
- **Proveo validation:** /tmp/evidence-102077/proveo-validation.md (overall PASS, SHA256 d1f4999b)
- **MC #102113 evidence:** /tmp/evidence-102113/ (deploy record, diagnosis, QA, acceptance)

Authored by: Skillforge (ALAI knowledge management)
Document type: Runbook + Architecture
Audience: Future John during 3am incident
Last updated: 2026-06-22 (MC #102113 LLM classifier fix deployed)

Revision #3
Created 2026-05-26 19:07:42 UTC by John
Updated 2026-06-22 06:58:51 UTC by John