mc.js done — Auto-Writeback to HiveMind + LightRAG Outbox

Runbook: MC Done Auto-Writeback to HiveMind

Owner: AgentForge
File: `~/system/tools/mc.js` (done command)
Version: 2.1.0
Last Updated: 2026-05-26

2026-05-26 Reliability Update (MC #102083)

MC completion writeback is now non-blocking on LightRAG availability:

- `~/system/tools/mc.js` delegates completion writeback to `~/system/lib/knowledge-writeback.js`.
- Memory and HiveMind writes remain best-effort, with fallback logging for HiveMind failures.
- Durable RAG writeback is queued through `~/system/lib/rag-outbox.js` on the `mc-outcomes` stream.
- `~/system/tools/rag-drain-worker.js` owns LightRAG upload, retry, direct/local endpoint support, backlog gates, and backpressure alerts.
- `~/system/tools/lightrag-outbox-ingest.js` migrates legacy `mc-task-outcomes.jsonl` entries into the durable SQLite outbox instead of uploading directly.
- Missing evidence sidecar behavior remains preserved via `task-outcomes-pending-evidence.jsonl`.

Evidence bundle for this update: `~/system/evidence/102083/wp4-writeback-reliability-report.md`. P2P verifier evidence: Company Mesh thread `mesh-thr-f759f9d2-a62d-491d-9ecb-677fcfd808fd`.

Purpose

Automatically capture task learnings when `mc.js done <id>` is called and write them to HiveMind + LightRAG pipeline. This closes the learning loop: task completion → knowledge indexing → next agent retrieval → smarter execution.

Before: Task learnings stayed in session logs. Next agent started from zero context.
After: Every completed task enriches the system's institutional memory.

What Gets Captured

When `mc.js done <id>` runs, it extracts and stores:

Field	Source	Example
`task_id`	MC task ID	8020
`title`	Task title	"System Evo T11: Blueprint liveness gate"
`outcome`	Done command message	"Gate implemented. mc.js checks blueprint mtime during done."
`owner`	Task owner	"john"
`duration`	Start → done delta	"2h 34m"
`tags`	Auto-extracted	["mc", "blueprint", "governance"]
`quality_gate`	Proveo validation	"passed" / "bypassed"

Additional context (if available):

- Session ID (links to full conversation log)
- Evidence ref (path to validation artifacts)
- Blueprint ref (if task was blueprint-linked)
- Related tasks (parent/child MC IDs)

Write Destinations

1. HiveMind (Immediate)

Target: `~/system/databases/hivemind.db` → `intel` table

Schema:

```
INSERT INTO intel (
  category,       -- "briefing"
  content,        -- Structured summary
  source,         -- "mc-done"
  metadata,       -- JSON blob
  created_at      -- ISO timestamp
) VALUES (?, ?, ?, ?, ?);
```

Write mode: Fire-and-forget async. Non-blocking.

Example entry:

```
category: "briefing"
content: "Task #8020 (Blueprint liveness gate) completed by john. Outcome: Gate implemented.
mc.js checks blueprint mtime during done. Duration: 2h 34m. Quality gate: passed."
source: "mc-done"
metadata: {"task_id":8020,"owner":"john","tags":["mc","blueprint"],"session_id":"731c913c"}
created_at: "2026-04-16T21:12:03Z"
```

---

2. Outbox (Deferred Bulk Ingest)

Target: `~/system/logs/mc-task-outcomes.jsonl`

Format: JSON Lines (one task per line)

Example:

```
{"task_id":8020,"title":"Blueprint liveness gate","outcome":"Gate
implemented","owner":"john","completed_at":"2026-04-
16T21:12:03Z","duration_minutes":154,"tags":["mc","blueprint","governance"],"quality_gate":"pa
ssed","session_id":"731c913c","evidence_ref":"/Users/makinja/system/evidence/system-evolution-
2026-04-16/"}
```

Consumption: Bulk-uploaded to LightRAG nightly by `lightrag-bulk-upload.js` (cron 03:00).

Why two destinations?

- HiveMind: Immediate availability for next agent query (low latency)
- LightRAG: Graph-based retrieval with entity relationships (high richness, 24h lag)

Flow Diagram

```
sequenceDiagram
    participant Agent
    participant mc.js
    participant HiveMind
    participant Outbox
    participant LightRAG
    participant NextAgent

    Agent->>mc.js: done 8020 "outcome text"
    mc.js->>mc.js: Extract task summary
    mc.js->>mc.js: Build intel entry

    par Write to HiveMind
        mc.js->>HiveMind: INSERT intel (fire-and-forget)
        HiveMind-->>mc.js: ACK (or log error)
    and Append to Outbox
        mc.js->>Outbox: Append JSONL line
    end

    mc.js->>Agent: Task marked done

    Note over Outbox,LightRAG: Nightly at 03:00
    LightRAG->>Outbox: Read new JSONL entries
    LightRAG->>LightRAG: Ingest to Neo4j graph

    NextAgent->>HiveMind: discover.js query
    HiveMind-->>NextAgent: Recent task outcomes
    NextAgent->>LightRAG: Graph query
    LightRAG-->>NextAgent: Entity relationships
    NextAgent->>NextAgent: Execute with enriched context
```

Usage

Basic Usage (Automatic)

No changes needed. Writeback happens automatically:

```
node ~/system/tools/mc.js done 8020 "Implemented blueprint liveness gate"
```

Console output:

```
Task #8020 marked as done
✓ Outcome recorded in HiveMind
✓ Queued for LightRAG ingest
```

With Evidence (Recommended)

```
node ~/system/tools/mc.js done 8020 \
  --evidence ~/system/evidence/my-validation/ \
  "All acceptance criteria met. Evidence in attached bundle."
```

Result: Evidence path included in intel metadata + outbox entry.

Bypass Proveo Gate (Emergency)

```
node ~/system/tools/mc.js done 8020 \
  --force "Production incident, validated live with CEO" \
  "Hotfix deployed"
```

Result: `quality_gate: "bypassed"` + force reason in metadata.

Verification

Check HiveMind Writeback

```
# List recent task outcomes
sqlite3 ~/system/databases/hivemind.db <<EOF
SELECT id, substr(content, 1, 80), created_at
```

```
FROM intel
WHERE source = 'mc-done'
ORDER BY id DESC
LIMIT 5;
EOF
```

Expected: Your recently completed task appears in top 5.

Check Outbox Queue

```
tail -n 5 ~/system/logs/mc-task-outcomes.jsonl
```

Expected: Last line is your task in JSON format.

Count pending:

```
wc -l < ~/system/logs/mc-task-outcomes.jsonl
```

Check LightRAG Ingest Status

```
curl -s http://localhost:9621/documents | jq '.statuses | {processed, pending, failed}'
```

Expected (after 03:00 cron):

- `pending` increases by number of new tasks
- `processed` increases over next 6-24h
- `failed` does not increase (or <1% of pending)

End-to-End Test

```
# 1. Create test task
TEST_ID=$(node ~/system/tools/mc.js add "E2E writeback test" --owner john | grep -o '#[0-9]*'
| tr -d '#')

# 2. Mark it done
node ~/system/tools/mc.js done $TEST_ID "Test outcome with unique marker $(date +%s)"

# 3. Check HiveMind (should appear within 1 second)
```

```
sqlite3 ~/system/databases/hivemind.db \
  "SELECT content FROM intel WHERE content LIKE '%E2E writeback test%' ORDER BY id DESC LIMIT
1;"

# 4. Check outbox (should appear immediately)
grep "$TEST_ID" ~/system/logs/mc-task-outcomes.jsonl

# 5. Check retrieval (next day after LightRAG ingest)
node ~/system/tools/discover.js "E2E writeback test"
```

Error Handling

HiveMind Write Failure

Symptom: Console shows:

```
[WARN] Failed to write task outcome to HiveMind: SQLITE_BUSY
Task #8020 still marked done, but intel not recorded.
```

Cause: Database locked (another agent writing).

Impact: Task marked done, but learning not immediately available.

Mitigation:

1. Outbox still written → LightRAG ingest will capture it
2. HiveMind write retried 3x with exponential backoff
3. If all retries fail → logged to `~/system/logs/mc-errors.log`

Recovery: Manual retry:

```
node ~/system/tools/mc.js writeback-retry 8020
```

Outbox Write Failure

Symptom:

```
[ERROR] Failed to append to mc-task-outcomes.jsonl: EACCES
Task marked done, HiveMind updated, but outbox NOT updated.
```

Cause: Permissions issue or disk full.

Impact: Task learning in HiveMind (immediate queries work) but NOT in LightRAG (graph queries miss it).

Recovery:

```
# Fix permissions
chmod 644 ~/system/logs/mc-task-outcomes.jsonl

# Backfill from HiveMind
node ~/system/tools/backfill-outbox.js --since "2026-04-16"
```

LightRAG Ingest Failure

Symptom (next day):

```
curl -s http://localhost:9621/documents | jq .statuses.failed
# Shows increase in failed count
```

Diagnosis:

```
docker logs lightrag | grep -i error | tail -20
```

Common causes:

- Malformed JSON in outbox (rare, jsonl validated on write)
- Neo4j out of memory (high load)
- Ollama timeout during entity extraction

Recovery:

1. Check Neo4j health: `docker logs neo4j --tail 50`
2. Restart LightRAG if needed: `docker restart lightrag`
3. Re-submit failed docs: `node ~/system/tools/lightrag-bulk-upload.js --retry-failed`

Performance & Overhead

Latency Impact on mc.js done

Before writeback: `mc.js done` took ~50ms
After writeback: `mc.js done` takes ~120ms (+70ms)

Breakdown:

- HiveMind write (async): 30ms
- Outbox append: 20ms
- Data extraction: 20ms

User perception: Negligible (< 200ms total).

Disk Usage

Outbox growth: ~500 bytes/task

Projection:

- 100 tasks/day = 50 KB/day = 18 MB/year
- Outbox rotation: Archive lines older than 90 days (cron)

HiveMind growth: ~800 bytes/task

Projection:

- 100 tasks/day = 80 KB/day = 29 MB/year
- HiveMind pruning: None (intel never deleted, indexed forever)

LightRAG Ingest Load

Nightly bulk upload:

- 100 tasks/day = 100 new documents
- Ingest time: ~2 min (LightRAG processes 50 docs/min under normal load)

Current backlog: 63,359 pending (from research phase). Drain time: ~21 hours at 50 docs/min.

Strategy: Prioritize today's tasks (outbox JSONL) → backlog second.

Configuration

Disable Writeback (Not Recommended)

Environment variable:

```
export MC_DISABLE_WRITEBACK=true
node ~/system/tools/mc.js done 8020 "outcome"
# HiveMind + outbox writes skipped
```

Use case: Testing mc.js in isolation without side effects.

Change Outbox Path

Environment variable:

```
export MC_OUTBOX_PATH=/tmp/my-outbox.jsonl
node ~/system/tools/mc.js done 8020 "outcome"
```

Use case: Custom ingest pipelines.

Tune HiveMind Retry Logic

File: `~/system/tools/mc.js` (around line 420)

```
const HIVEMIND_RETRIES = 3;           // Default 3
const HIVEMIND_RETRY_DELAY_MS = 500;  // Default 500ms
```

Increase if HiveMind frequently locked.

Integration with Other Systems

Session Logs Linkage

Writeback includes `session_id` (if available). Full conversation context in:

```
~/system/memory/sessions/2026-04-16-<session_id>.md
```

Use case: Debugging why agent made specific decision → trace back to full session.

Evidence Bundles

Writeback includes `evidence_ref` (if provided). Structure:

```
~/system/evidence/system-evolution-2026-04-16/
  ├── SUMMARY.md
  ├── v1-lightrag-health.json
  ├── v2-intel-tail.txt
  └── ...
```

Use case: Proveo validation requires evidence → MC done links to bundle → LightRAG indexes → future agents discover "how system-evolution was validated."

Blueprint Linkage

Writeback includes `blueprint_ref` (if task was created with `--blueprint-ref`).

Use case: Agent query "which tasks updated Plock blueprint?" → LightRAG returns graph of related tasks.

Troubleshooting

Issue 1: No Intel Appearing in HiveMind

Diagnosis:

```
# Check last HiveMind write
sqlite3 ~/system/databases/hivemind.db \
  "SELECT MAX(created_at) FROM intel WHERE source='mc-done';"

# Check mc.js error log
```

```
grep "HiveMind write" ~/system/logs/mc-errors.log | tail -10
```

Possible causes:

- HiveMind DB locked (check for long-running agent)
- Disk full (`df -h ~/system`)
- mc.js old version (check `git log ~/system/tools/mc.js | head -1`)

Issue 2: Outbox Growing But LightRAG Not Ingesting

Diagnosis:

```
# Check LightRAG ingest rate
curl -s http://localhost:9621/documents | jq '.statuses | {processed, pending}'

# Check cron job status
launchctl list | grep lightrag-bulk-upload
```

Possible causes:

- Cron job dead (restart: `launchctl load ~/Library/LaunchAgents/com.alai.lightrag-bulk-upload.plist`)
- LightRAG backlog too large (prioritize: `lightrag-bulk-upload.js --priority-outbox`)
- Neo4j out of disk (`docker exec neo4j df -h /data`)

Issue 3: Duplicate Intel Entries

Symptom:

```
sqlite3 ~/system/databases/hivemind.db \
  "SELECT COUNT(*) FROM intel WHERE content LIKE '%Task #8020%';"
# Returns: 3 (expected: 1)
```

Cause: Agent called `mc.js done 8020` multiple times (idempotency bug).

Fix (immediate):

```
# Dedupe
sqlite3 ~/system/databases/hivemind.db <<EOF
```

```
DELETE FROM intel
WHERE id NOT IN (
  SELECT MIN(id) FROM intel
  WHERE source='mc-done'
  GROUP BY json_extract(metadata, '$.task_id')
);
EOF
```

Fix (permanent): Add unique constraint (MC #8051):

```
CREATE UNIQUE INDEX idx_intel_mc_task ON intel (
  (json_extract(metadata, '$.task_id'))
) WHERE source = 'mc-done';
```

Best Practices

1. Always Provide Outcome Message

```
# BAD (generic)
node ~/system/tools/mc.js done 8020

# GOOD (specific)
node ~/system/tools/mc.js done 8020 \
  "Blueprint liveness gate implemented. mc.js now checks mtime. 10 existing blueprints
audited."
```

Why: Outcome text flows to HiveMind → LightRAG → future agent queries. Generic "done" loses context.

---

2. Link Evidence When Available

```
node ~/system/tools/mc.js done 8020 \
  --evidence ~/system/evidence/my-validation/ \
  "All acceptance criteria met"
```

Why: Evidence link enables future agents to learn HOW task was validated, not just THAT it was done.

3. Tag Retroactively if Needed

Outbox uses auto-extracted tags, but you can add manual tags:

```
node ~/system/tools/mc.js update 8020 --tags "critical,production,hotfix"
node ~/system/tools/mc.js done 8020 "Emergency fix deployed"
```

Tags flow to HiveMind + outbox.

---

4. Review Outbox Weekly

```
# Check for stale entries (> 7 days, not ingested)
awk -F, '{print $1}' ~/system/logs/mc-task-outcomes.jsonl | \
  while read line; do
    TIMESTAMP=$(echo "$line" | jq -r .completed_at)
    # (date comparison logic)
  done
```

Action: If > 500 stale entries → investigate LightRAG pipeline.

Monitoring & Alerts

Key Metrics

Metric	Command	Threshold
Writeback success rate	`grep "✓ Outcome recorded" ~/system/logs/mc.log | wc -l`	> 95%
Outbox size	`wc -l ~/system/logs/mc-task-outcomes.jsonl`	< 10,000 lines
LightRAG ingest lag	Compare outbox timestamp vs. LightRAG processed count	< 48h

Alerting (future):

```
# Add to daily briefing
if [ $(wc -l < ~/system/logs/mc-task-outcomes.jsonl) -gt 5000 ]; then
```

```
  echo "WARN: MC outbox has 5000+ pending tasks. LightRAG ingest lagging."
fi
```

Future Enhancements

Planned (MC #8052)

1. **Smart tag extraction:** Use LLM to extract domain tags ("fintech", "RAG", "mobile")
2. **Outcome quality scoring:** Flag low-quality outcomes ("done" with no context)
3. **Session summary:** Auto-generate 3-sentence summary from full session log

Under Consideration

- **Real-time LightRAG push:** Skip outbox, ingest to LightRAG immediately (high load concern)
- **Federated writeback:** Write to HiveMind + external systems (Notion, BookStack)
- **Agent performance metrics:** Track task duration by agent type → identify bottlenecks

---

Related Documentation

- **System evolution upgrade:** `~/system/docs/system-evolution-2026-04-16.md`
- **LightRAG default-on:** `~/system/docs/runbooks/lightrag-default-on.md`
- **HiveMind schema:** `~/system/databases/hivemind-schema.sql`
- **Outbox format spec:** `~/system/specs/mc-outbox-format.md` (upcoming)

Changelog

Date	Version	Change
2026-04-16	2.0.0	Auto-writeback implemented (MC #8020 Task 7)
2026-03-15	1.8.0	Added `--evidence` flag to mc.js done
2026-02-10	1.5.0	HiveMind integration

Questions? Contact Chip Huyen (AgentForge lead) or Petter Graff (team lead).

Revision #3
Created 2026-04-16 21:44:06 UTC by John
Updated 2026-05-26 14:41:57 UTC by John