

Subagent Output Contract: Receipt Discipline for Agent Dispatch

Status: DRAFT **Date seeded:** 2026-08-21 **Author:** Reuben Bowlby (ORCID 0009-0002-5620-1103) **Repository:** hummbl-io/hummbl-papers **Source incident:** apex-nexus docs/research/2026-08-21_subagent-output-contract.

Abstract

Multi-agent LLM systems increasingly dispatch subagents to perform file-producing tasks — writing documents, generating code, building research artifacts. Without output contract discipline, these dispatches produce three recurring failure modes: path drift (files saved to wrong directories), unverifiable claims (self-reported metrics that do not match actual measurements), and naming inconsistency (no convention enforced across a series). This paper presents a case study of a three-faculty dispatch incident in which three `subagent_general` dispatches produced three documents in three different directories with two naming conventions and an 83% line-count overcount (979 reported, 535 measured). We propose a six-section output contract — `OUTPUT_PATH`, `NAMING_CONVENTION`, `SIBLING_CONTEXT`, `VERIFICATION_STEP`, `CLAIM_DISCIPLINE`, `RETURN_FORMAT` — as the receipt discipline for agent dispatch, and map the failure modes to the MAST multi-agent failure taxonomy (Cemri et al. 2025). The strongest finding is not a new framework; it is the governance value of requiring tool-backed receipts for quantitative claims before the subagent returns, and the observation that “self-contained prompt” in existing dispatch literature means “include all task context” but never “include all output contract requirements.” We compare the proposed contract against NabaOS (Basu 2026), which implements the same principle at the cryptographic level using HMAC-signed tool receipts, and argue that the advisory contract and the cryptographic receipt are complementary layers of the same verification discipline.

1. Problem

Subagent dispatch — where a parent agent launches a child agent to perform a self-contained task — is a core primitive of multi-agent LLM systems. When the subagent writes files, three failure modes recur:

1. **Path drift:** The subagent saves the output file to an unexpected directory. Without an explicit output path in the prompt, the subagent chooses a reasonable default that differs from what the parent expected.
2. **Unverifiable claims:** The subagent self-reports quantitative metrics (line count, file size, section count) in its return summary without running verification tools. The parent has no way to distinguish measured values from estimates without independently verifying after return.
3. **Naming inconsistency:** When multiple subagents are dispatched for a series (e.g., three faculty documents), each chooses a different naming convention (hyphens vs underscores, different directory structures), producing a scattered artifact set.

These failure modes are not unique to our system. The MAST taxonomy (Cemri et al. 2025) identifies 14 failure modes across 3 categories in multi-agent LLM systems, validated across 1600+ annotated traces with inter-annotator agreement $\kappa=0.88$. Path drift maps to MAST category (i) — specification and system design failures. Unverifiable claims map to MAST category (iii) — task verification and termination failures. The MAST finding that “improved specification of agent roles” alone is insufficient — that “identified failures require more complex solutions” — motivates the two-layer approach in this paper: a pre-dispatch contract (specification) plus a post-return verification protocol (verification).

2. Prior Art

The output-contract approach builds on four traditions:

- **Multi-agent failure taxonomies** (Cemri et al. 2025; MAS-FIRE 2026): MAST provides the first comprehensive failure taxonomy for multi-agent LLM systems, with 14 failure modes in 3 categories. MAS-FIRE extends this with 15 fault types and 4 fault-tolerant tiers. Our six-section contract addresses MAST categories (i) and (iii) directly.

- **Tool receipt verification** (Basu 2026): NabaOS generates HMAC-signed tool execution receipts that the LLM cannot forge, then cross-references claims against receipts. It detects 94.2% of fabricated tool references and 87.6% of count misstatements at <15ms overhead. NabaOS uses Indian epistemology (Nyāya Śāstra) to classify claims by epistemic source: direct tool output (pratyakṣa), inference (anumāna), external testimony (śabda), absence (abhāva), or ungrounded opinion. Our contract is the advisory version of what NabaOS implements cryptographically.
- **ML reproducibility checklists** (Pineau et al. 2021): the NeurIPS Reproducibility Program established documentation standards for ML experiments. The sibling paper on experiment receipt discipline (Bowlby 2026) applies these to autoresearch loops. This paper applies the same principle to agent dispatch — the receipt is the contract, and the contract is the receipt.
- **Pre-registration standards** (OSF; PCI RR): the registered-reports model pre-declares hypotheses and analysis plans before data collection. Our output contract pre-declares the output path, naming convention, and verification commands before the subagent executes — the contract is prospective, not retrospective.

The contribution is the *application* of receipt discipline to agent dispatch, not the invention of receipts or verification. A human collaborator can be asked “where did you save the file?” and will answer reliably; an autonomous subagent needs the output path specified in the prompt, or it will choose a different one.

3. The Incident

3.1 Context

Three `subagent_general` dispatches were launched across one session to build faculty documents for the ARCANA Academy project — a pedagogical architecture using AI-mediated multi-persona learning. Each subagent received a detailed task prompt with substantive content requirements (faculty structure, curriculum, seminars, emergent gaps) but no output contract: no explicit output path, no naming convention, no sibling context, no verification step, no claim discipline, no return format.

3.2 Evidence

Faculty	Actual path	Expected directory	Naming	Size
Kitchen & Table	C:\Users\reuben\kitchen-table\	PROJECTS/faculty/faculties/	hyphens	145,614 bytes
Religion & Ritual	C:\Users\reuben\project-religion-ritual\	PROJECTS/arcana/faculties/.md	underscores	101,248 bytes
Death & Mortality	C:\Users\reuben\project-death-mortality\	PROJECTS/mortality/faculties/.md	underscores	158,633 bytes

Three documents, three directories, two naming conventions. None in the canonical project directory.

3.3 The Line-Count Discrepancy

The Death & Mortality subagent’s completion notification reported “979 lines.” PowerShell measurement returned 535 lines — an 83% overcount (444 lines). The file size was approximately correct (158,633 bytes vs. ~159KB claimed). The subagent likely counted wrapped lines or estimated from content rather than running a line-count tool.

Subagent claim: 979 lines, ~159KB **Actual measurement:** 535 lines, 158,633 bytes

There was no verification step in the prompt that would have caught this. The parent discovered the discrepancy only by independently running `Measure-Object` after the subagent returned.

3.4 What the Subagent Got Right

The section structure claim was accurate. The subagent reported “16 sections + Emergent Gaps”; independent verification via `Select-String -Pattern "^## "` confirmed 18 `##` headings (§0 through §16 plus Emergent Gaps). The substantive content quality was high — the document is a well-structured 158KB faculty document with seminars, curriculum architecture, and 21 emergent gaps. The failure was not in the work product but in the self-reporting of the work product.

4. The Output Contract

We propose six mandatory sections for every `subagent_general` prompt that produces a file. Prompts missing any section are considered defective and should be rejected before dispatch.

4.1 The Six Sections

Section	What it specifies	Failure it prevents
OUTPUT_PATH	Exact absolute file path	Path drift (§3.2)
NAMING_CONVENTION	Prior file examples	Naming inconsistency (§3.2)
SIBLING_CONTEXT	Paths of prior files in series	Structural inconsistency
VERIFICATION_STEP	Commands to run after writing	Unverifiable claims (§3.3)
CLAIM_DISCIPLINE	Rules for quantitative claims	Fabricated metrics (§3.3)
RETURN_FORMAT	Structured return with receipts	Disordered summaries

4.2 Pre-Dispatch Checklist

Before launching a file-producing subagent, the parent verifies:

- ☐ `OUTPUT_PATH` is specified as an absolute path
- ☐ `NAMING_CONVENTION` matches prior files or establishes one
- ☐ `SIBLING_CONTEXT` is provided or marked “no prior files”
- ☐ `VERIFICATION_STEP` specifies which commands to run
- ☐ `CLAIM_DISCIPLINE` is included in the prompt
- ☐ `RETURN_FORMAT` is specified

4.3 Post-Return Verification

After the subagent returns, the parent independently verifies:

1. File exists at the claimed path
2. File size matches the claimed size
3. Line count matches the claimed count
4. Section headings match the claimed headings
5. Any discrepancy is flagged

4.4 Relationship to NabaOS

NabaOS (Basu 2026) implements the same principle — tool-backed receipts for agent claims — at the cryptographic level. NabaOS generates HMAC-signed tool execution receipts that the LLM cannot forge, then cross-references claims against receipts in real time. Our contract is advisory: it requires the subagent to run verification commands and include raw output in its return summary, and it requires the parent to independently verify after return.

The two approaches are complementary:

Dimension	Output Contract (this paper)	NabaOS (Basu 2026)
Enforcement	Advisory (rule-based)	Cryptographic (HMAC-signed)
Overhead	Zero runtime cost	<15ms per response
Coverage	Parent must verify post-return	Runtime catches 94.2% of fabrications
Deployability	Any LLM agent system	Requires runtime integration
Epistemic classification	Binary (tool receipt vs estimate)	5-source (pratyakṣa/anumāna/śabda/abhāva/ungrounded)

For systems that cannot integrate NabaOS at the runtime level (e.g., using third-party agent frameworks where the tool layer is not modifiable), the output contract provides the same verification discipline in an advisory form. For systems that can integrate NabaOS, the output contract provides the prompt-level specification that the cryptographic layer enforces.

5. Mapping to the MAST Taxonomy

The MAST taxonomy (Cemri et al. 2025) identifies 14 failure modes in 3 categories. Our incident maps to two:

5.1 Specification and System Design Failures (MAST Category i)

Failure: The prompt did not specify the output path, naming convention, or sibling context.

MAST mapping: This is a specification failure — the parent agent did not provide sufficient specification for the subagent to produce the expected output. The subagent behaved correctly given the specification it received; the specification was defective.

Contract section that prevents it: OUTPUT_PATH + NAMING_CONVENTION + SIBLING_CONTEXT.

5.2 Task Verification and Termination Failures (MAST Category iii)

Failure: The subagent self-reported 979 lines when the actual count was 535. No verification step was in the prompt.

MAST mapping: This is a task verification failure — the subagent’s self-report was not verified against the actual output before the task was considered complete. The MAST finding that “improved specification” alone is insufficient — that failures “require more complex solutions” — is exactly why our contract has two layers: pre-dispatch specification (the six sections) and post-return verification (the parent’s independent check).

Contract section that prevents it: VERIFICATION_STEP + CLAIM_DISCIPLINE + RETURN_FORMAT.

5.3 What MAST Does Not Cover

MAST identifies failure modes but does not prescribe output contracts. The MAST paper’s two interventions — “improved specification of agent roles” and “enhanced orchestration strategies” — are both insufficient per the authors’ own finding. Our contract is a concrete instantiation of “improved specification” that the MAST paper calls for but does not provide.

6. The “Self-Contained” Gap

The existing dispatch infrastructure uses the phrase “self-contained” in three places:

Source	Quote	What it means	What it doesn't mean
swarm-subagent skill	"A self-contained prompt (must include all context — no shared state)"	Subagent needs all task context	No output contract
dispatch skill	"Each subagent receives an explicit, self-contained prompt with all necessary context"	Same as above	Same gap
subagent-dispatch rule	Implies self-containment via tool permission rules	Tools must be available	No file output contract

The gap: "Self-contained" is interpreted as "include all task context" but never as "include all output contract requirements." A subagent can receive a perfectly self-contained task description — all the substantive content it needs — and still have no idea where to save the file, what to name it, or how to verify its output.

This gap is the same class of failure documented in:

- **AAR 20260731-batch-47** (HUMMBL internal): `swarm-subagent` dispatched subagents reported `dod_check.actual_sources` as 15-30 when actual evidence was 0. False self-attestation with no verification.
- **PR #498/#499** (HUMMBL internal): AAR claimed a branch-switch hook that did not exist. Unverified causal claim.

Pattern: Agents self-report metrics without tool receipts when not explicitly required to produce them. The fix is structural — require the receipts in the prompt, verify them after return.

7. Corpus Quantification

7.1 The Incident

Metric	Value
Total subagent dispatches	3
Files in correct directory	0/3 (0%)
Naming convention consistent	2/3 (67%) — two used underscores, one used hyphens
Line-count claims verified	0/3 (0%) — no verification step in any prompt
Line-count discrepancy (Death & Mortality)	83% overcount (979 claimed, 535 actual)
Section structure claims verified	1/1 (100%) — accurate when checked
Substantive content quality	High — all three documents are well-structured

7.2 Remediation

After the incident, all three files were copied to the canonical directory (`PROJECTS/arcana/faculties/`) with consistent naming (`snake_case_faculty.md`). The candidate rule was filed. The next dispatch using the

contract would be the test case.

7.3 Prior Incidents (Same Pattern)

Incident	Date	Failure	Root cause
intel-surge batch-47	2026-07-31	<code>dod_check.actual_source</code> reported 15-30, actual 0	No insertion-time validator; false self-attestation
PR #498 AAR	2026-08-15	AAR claimed branch-switch hook that didn't exist	Causal claim not verified against source
This incident	2026-08-21	Line count 979 reported, 535 actual	No verification step in prompt

Three incidents, same pattern: self-attestation without tool receipts. The output contract is the structural fix.

8. Limitations

1. **Single-operator, single-incident.** The case study is a single session’s three dispatches. There is no controlled comparison between contract-bearing and contract-free dispatches. The incident is illustrative, not statistically powered.
2. **Advisory, not enforced.** The contract is a rule that the parent agent must follow. There is no runtime enforcement — a parent agent can dispatch without the contract, and a subagent can ignore the verification step. NabaOS (Basu 2026) shows that cryptographic enforcement is feasible; the contract is the advisory layer that works when cryptographic enforcement is not available.
3. **No automated validation.** The pre-dispatch checklist is manual. A validator script that checks the prompt for the six mandatory sections before `run_subagent` is called would reduce the parent’s burden. This is future work.
4. **The line-count discrepancy is a single data point.** We observed one overcount (83%). We do not know the distribution of overcounts across subagent dispatches. The MAST corpus (1600+ traces) would be a better source for distributional claims.
5. **The contract has not been tested in a controlled experiment.** The next dispatch using the contract will be the first test case. The review criteria for promoting the candidate rule to canonical status are: (a) file lands in correct location, (b) line count claims match actual measurements, (c) path drift does not recur.

9. Relationship to the Experiment Receipt Paper

The sibling paper on experiment receipt discipline (Bowlby 2026) applies the same principle — receipt discipline for agent-generated artifacts — at a different scale:

Dimension	Experiment Receipt (Bowlby 2026)	Output Contract (this paper)
Artifact governed	ML experiments	Subagent dispatch outputs
Receipt fields	7 (hypothesis, baseline, noise floor, outcome, decision, config, provenance)	6 (output path, naming, siblings, verification, claim discipline, return format)
Prospective gate	Noise-floor gate blocks architecture experiments	Pre-dispatch checklist blocks defective prompts

Dimension	Experiment Receipt (Bowlby 2026)	Output Contract (this paper)
Retrospective check	Receipt completeness audit	Post-return verification protocol
Failure prevented	Cherry-picking, baseline drift, noise-floor blindness	Path drift, unverifiable claims, naming inconsistency

Both papers instantiate the portfolio’s unifying thesis: **governed reasoning is the requisite-variety guardrail on an infinite generative space**. The experiment receipt paper governs the generative space of hypotheses and patches; the output contract paper governs the generative space of file-producing subagent dispatches. The guardrail in both cases is a receipt schema that separates observation from interpretation and requires tool-backed evidence for quantitative claims.

10. Open Questions

- Can the six-section contract be validated automatically (e.g., a pre-dispatch validator that checks the prompt for all six sections before `run_subagent` is called)?
- Does the contract generalize beyond `subagent_general` (read-write) to `subagent_explore` (read-only)? Read-only subagents still make quantitative claims in their return summaries that could be unverifiable.
- What is the distribution of line-count discrepancies across subagent dispatches? Our single data point (83% overcount) is illustrative, not distributional.
- Should the CLAIM_DISCIPLINE section adopt NabaOS’s five-source epistemic classification (pratyakṣa/anumāna/śabda/abhāva/ungrounded) instead of the binary “tool receipt vs estimate”?
- Should the contract be unified with the AAR command’s causal-claim verification rule into a single “claim verification” rule that applies across AARs, subagent returns, and any agent self-reporting context?
- What is the minimum sample size for a controlled experiment comparing contract-bearing vs contract-free dispatches on path drift and claim accuracy?

11. External Replication

The incident is reproducible: any agent system that dispatches subagents to write files without specifying the output path will produce path drift. The line-count overcount is reproducible in any system where the subagent self-reports metrics without running verification tools. A replicator can:

1. Dispatch three `subagent_general` instances with substantive task prompts but no output contract
2. Record where each subagent saves its output
3. Compare self-reported line counts to `wc -l` or `Measure-Object` results
4. Then dispatch three more with the six-section contract and compare

The candidate rule (`rules/subagent-output-contract.md` in the apex-nexus repo) contains the full contract text, an annotated example prompt, and the pre-dispatch checklist.

Public Source Artifacts

- Candidate rule: `rules/subagent-output-contract.md` (apex-nexus repo, PR #62)
- Incident report: `docs/research/2026-08-21_subagent-output-contract.md` (apex-nexus repo)
- arXiv sweep: `docs/research/2026-08-21_arxiv-sweep-arcana-subagent-contract.md` (apex-nexus repo)
- Cognitive ledger entry: `clp-1de570615a3d`
- Research notebook

References

- Basu, A. (2026). Tool Receipts, Not Zero-Knowledge Proofs: Practical Hallucination Detection for AI Agents. arXiv:2603.10060.
- Cemri, M. et al. (2025). Why Do Multi-Agent LLM Systems Fail? arXiv:2503.13657.
- Xiao, H. et al. (2026). TARL: Transaction-Aware Reliable Ledgers for Executable Memory Management in Long-Term Agents. arXiv:2608.03699.
- MAS-FIRE (2026). Fault Injection and Reliability Evaluation for LLM-Based Multi-Agent Systems. arXiv:2602.19843.
- OrchestraBench (2026). Evaluating Multi-Agent Orchestration Failure Modes, Recovery, and Decomposition Quality. arXiv:2608.05263.
- Pineau, J. et al. (2021). Improving Reproducibility in Machine Learning Research. JMLR 22(164).
- Bowlby, R. (2026). Experiment Receipt Discipline for Small-Model Autoresearch. hummbl-papers.
- Center for Open Science. Open Science Framework: Pre-Registration Standards.