CONCEPTS GUIDE

# SL 4K

USAREUR-AF

# CONCEPTS GUIDE — SL 4K COMPANION — KNOWLEDGE MANAGER · MAVEN SMART SYSTEM (MSS)

*Specialist Course Manual*

HEADQUARTERS
UNITED STATES ARMY EUROPE AND AFRICA
(USAREUR-AF)
Wiesbaden, Germany



**26 MARCH 2026**

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

# CONCEPTS GUIDE — SL 4K COMPANION — KNOWLEDGE MANAGER · MAVEN SMART SYSTEM (MSS)

**Forward:** The KM on MSS builds the institutional memory of the formation. The primary output is structured, findable, usable knowledge — lessons learned, SOPs, decision records, after-action content — stored in MSS in a way that supports future operations. If that knowledge cannot be found, it was never really captured. **Purpose:** Develops the mental models required to design, implement, and maintain knowledge management solutions on MSS effectively. Read before beginning SL 4K task instruction. *HQ USAREUR-AF · v1.0 · 2026 · DISTRIB: USG only*

## SECTION 1 — THE KNOWLEDGE MANAGER'S ROLE ON MSS

**BLUF:** The KM on MSS builds the institutional memory of the formation. The primary output is structured, findable, usable knowledge — lessons learned, SOPs, decision records, after-action content — stored in MSS in a way that supports future operations. If that knowledge cannot be found, it was never really captured.

### 1-1. Role Distinctions

| Role | Primary Output | Works In | Key Skill |
|---|---|---|---|
| Maven User (SL 1/SL 2) | Consumes and creates basic content | Workshop, Quiver | Navigation, basic data literacy |
| Builder (SL 2/SL 3) | Dashboards, views, basic transforms | Workshop, Pipeline Builder | Visual design, data connection |
| ORSA (SL 4G) | Analysis products, commander briefs | Notepad, Contour, Python | Quantitative reasoning |
| ML Engineer (SL 4M) | Models, predictions | Code Repos, Model Registry | Statistical methods |
| SWE (SL 4L) | Integrated applications, OSDK | Code Repos, TypeScript | Software engineering |
| **Knowledge Manager (SL 4K)** | **Institutional knowledge architecture** | **Ontology Manager, Workshop, AIP** | **Information architecture, governance** |

The KM is unique in that the primary deliverable is organizational — a designed system for capturing, organizing, and retrieving knowledge — not a single dashboard or analysis. A KM who builds a beautiful lessons-learned form but designs no retrieval interface, no governance process, and no retention policy has built an input with no output.

The data steward role overlaps partially: both care about data quality and governance. The data steward governs operational data — the data used to conduct missions. The KM governs knowledge products — the accumulated learning about how to conduct missions. When a single person fills both functions, the risk is that data stewardship work crowds out knowledge management work because data stewardship has more visible short-term urgency.

**NOTE — FM 6-0 KM Process and Tools (TM-40K Section 1-5b):** TM-40K Section 1-5b codifies the FM 6-0 five-step KM process, three KM tasks, and six KM tool categories as the doctrinal foundation for all MSS knowledge management work. Review before beginning any KM architecture design.

**NOTE — FM 6-22 Developmental Domains:** SL 4K now references FM 6-22 (Army Leadership and the Profession) developmental domains as a framework for expertise profiling in MSS personnel knowledge systems.

**NOTE — FM 3-57 Civil Knowledge Integration (CKI):** SL 4K includes a note on FM 3-57 (Civil Affairs Operations) CKI methodology for capturing and structuring civil information within MSS knowledge architectures supporting civil-military operations.

## 1-2. MSS vs. SharePoint and AKO

Most Army KMs arrive with the SharePoint mental model: folders, files, documents, permissions. MSS operates on a fundamentally different model.

| Dimension | SharePoint Model | MSS/Foundry Model |
|---|---|---|
| Knowledge storage | Documents in folders in sites | Objects — structured data records in the Ontology |
| Retrieval mechanism | File navigation; search if documents are named and tagged correctly (usually not) | Query-based — filter by unit, date, functional area, classification level, operational phase |
| Entry discipline | Drop a Word document into a folder — done | Write structured data into a Lessons Learned Object Type with specific property fields |
| Payoff | Easy to enter; hard to retrieve at scale | More discipline at entry time; retrieval is actually possible |

A G3 preparing for DEFENDER 2026 cannot usefully search a SharePoint site containing 847 Word documents from four exercises. The same G3 can query an MSS Lessons Learned dataset filtered to `operation_type = DEFENDER`, `functional_area = fires`, `unit_level = BCT`, `status =`

`implemented` and get twenty vetted, actionable results in under a minute. That difference is what justifies the discipline cost of structured capture.

---

## SECTION 2 — INFORMATION ARCHITECTURE — THE KM'S FOUNDATIONAL MENTAL MODEL

**BLUF:** Before building any form, workflow, or repository on MSS, design the information architecture. Every technical decision downstream depends on getting the architecture right.

### 2-1. The Four Architecture Questions

Before any build begins, answer these four questions in writing:

| Question | What It Produces | Error If Skipped |
|---|---|---|
| **What types of knowledge need to be captured?** | A categorized inventory: lessons learned, SOPs, personnel expertise profiles, decision records, AAR content — each category may require a different Object Type, governance, and retention period | Conflating categories into a single undifferentiated repository produces a system no one uses |
| **Who creates it, and what is their incentive structure?** | Creator profile: who will actually create the knowledge, under what time pressure, with what level of subject-matter familiarity | A 45-minute form will not be completed. A form requiring detailed operational knowledge will be filled out incorrectly by people who lack that knowledge |
| **Who consumes it, and what does "findable" mean to them?** | Consumer search patterns: what terms they use, what filters they apply, what they are trying to accomplish | A record findable to the person who entered it is not findable to a G3 staff officer two years later |
| **How long must it be retained, and what happens at end of life?** | Retention policy designed into the architecture from the start | Knowledge systems fill with outdated content unless retention policy is built in; an SOP superseded in 2024 remaining visible alongside the current 2026 version actively harms users |

### 2-2. Taxonomy Design — The Goldilocks Problem

The taxonomy is the classification scheme making knowledge findable. On MSS, taxonomy decisions manifest as property fields on Object Types — controlled vocabulary dropdowns, tag fields, classification codes.

| Problem | Description | Result |
|---|---|---|
| **Too granular** | 47 functional area codes, 12 unit levels, 8 operation types, 6 classification levels, 4 status values | Most categories contain zero or one entries. Users won't learn the taxonomy. Entries will be miscoded. Search returns misleading results. |
| **Too broad** | Three fields: title, date, and free-text narrative | Repository is unsearchable at scale. Every entry requires reading the full text to determine relevance. |
| **Design target** | The minimum set of controlled-vocabulary fields enabling the most important search use cases, combined with free text only where controlled vocabulary genuinely cannot capture the content | Work backward from the ten most important queries a future consumer will run |

## 2-3. The Architecture Worksheet

Before beginning any MSS knowledge architecture, the KM completes a brief written architecture document:

| Element | Question |
|---|---|
| Knowledge type inventory | What are the 3–5 distinct types of knowledge this system will capture? |
| Creator profile | Who enters data, when, and under what conditions? |
| Consumer use cases | What are the 5–10 most important search/retrieval scenarios? |
| Taxonomy fields | What controlled-vocabulary fields support the consumer use cases? |
| Retention policy | How long is each knowledge type retained? What triggers end-of-life? |

Spending two hours on architecture prevents two months of rebuilding after discovering the system does not support the retrieval use cases anyone actually needs.

**NOTE — Foundry Project Architecture Taxonomy (Palantir Best Practice)**

When designing knowledge management systems, organize Foundry projects by function:

| Type | Naming Convention | KM Relevance |
|---|---|---|
| Datasource | `Datasource - {Name}` | Raw AAR feeds, doctrine source ingestion |
| Data Integration | `Integration - {Name}` | LL deduplication, classification pipelines |
| Ontology | `Ontology - {Name}` | Knowledge object types, expertise registries |
| Application | `Application - {Name}` | Search interfaces, KM dashboards |

| Type | Naming Convention | KM Relevance |
|---|---|---|
| Sandbox | `[sandbox] Name` | Training exercises, KM experimentation |

*Source: Palantir Developer Community — Ontology and Pipeline Design Principles*

## SECTION 3 — STRUCTURED VS. UNSTRUCTURED KNOWLEDGE

**BLUF:** The distinction between structured and unstructured knowledge is the most important design decision a KM makes on MSS. Structured knowledge is queryable and analyzable. Unstructured knowledge is rich but opaque.

### 3-1. The Spectrum in Practice

Knowledge is not binary structured/unstructured. It exists on a spectrum, and the KM makes deliberate choices about where to design each knowledge type:

| Knowledge Type | Fully Unstructured | Partially Structured | Fully Structured |
|---|---|---|---|
| AAR content | One uploaded Word document | Structured header (unit, date, operation) + free-text narrative | Discrete observations each recorded as Object with coded properties |
| Lessons learned | Free narrative in email thread | Form with title, narrative, unit, date | Object with observation, discussion, recommendation, status, functional area, operation type, unit |
| SOP version | PDF in SharePoint folder | Document Object with version number and effective date | Document Object with full version history, approval chain, change log linked to Objects |
| Personnel expertise | Resume PDF | Profile form with skill tags | Expertise Object with skill codes, proficiency levels, certification dates, and unit linkages |

The KM does not always drive toward full structure. The observation narrative in a lessons-learned entry should be free text — the nuance of what happened does not fit a dropdown. But the functional area, unit level, and status absolutely should be controlled vocabulary, because those are the fields consumers will filter on.

### 3-2. AIP as a Bridge

AIP Logic provides a partial bridge between unstructured and structured knowledge. Given a free-text AAR narrative, an AIP workflow can extract candidate lessons, suggest functional area tags, and draft structured recommendations for human review.

The KM who uses AIP for knowledge extraction must understand its limitations: AIP extracts what is explicit in the text and makes plausible inferences about what is implicit. It does not understand operational context, it does not know the unit's specific doctrine, and it does not distinguish between a routine observation and a critical safety finding. **Every AIP extraction requires SME review before the output enters the institutional knowledge base.**

The practical design pattern: use AIP to generate structured draft entries from unstructured inputs → present drafts to human reviewers for validation and correction → write validated entries into the Ontology. This pattern reduces the labor cost of structured capture while preserving human judgment in the loop.

---

## SECTION 4 — FORMS AND ACTIONS AS KNOWLEDGE CAPTURE INSTRUMENTS

**BLUF:** In Foundry, a form is an Action — it writes structured data into the Ontology. Every field is a commitment. Form design discipline is the practice of ruthlessly minimizing that commitment to what is genuinely necessary and genuinely usable.

### 4-1. The Action Mental Model

Workshop forms in MSS are Action configurations that write Object data. When a user submits a lessons-learned entry, a Foundry Action executes: it validates input, creates or updates a Lessons Learned Object in the Ontology, and applies configured rules (routing, notifications, status updates).

Form design and data model design are the same activity. Every field on a form corresponds to a property on an Object Type. Adding a field commits to maintaining, querying, and governing that property for the life of the system.

## 4-2. The Field Commitment

| Cost | Description |
|---|---|
| **Creator time cost** | Every field adds time and cognitive load. A 45-minute form will not be completed during busy operational periods. Likely outcomes: entries not submitted, or entries submitted with fields left blank or filled with garbage. |
| **Maintenance cost** | Controlled vocabulary lists (functional area codes, unit identifiers, operation type values) go stale as the organization evolves. Someone must update them. |
| **Query cost** | Every field creates an implicit promise to consumers that it will be populated consistently enough to support filtering. A field empty 60% of the time — because it is too burdensome to fill in — is worse than no field. It gives consumers false confidence that filtering on it returns complete results. |

## 4-3. Form Design Discipline

**Required fields must be genuinely required.** If the record is useless without the field, make it required. If the record is useful without it, make it optional — or eliminate it. Required fields that cannot always be satisfied will produce incomplete records where the required field contains placeholder garbage.

**Constrained values where possible.** Every free-text field that could be a dropdown should be a dropdown. Free text is appropriate for narrative content where the constraint would destroy the information value — not for functional area codes, unit designations, or operational phase tags.

**Free text is the last resort, not the default.** Reserve free text for the observation narrative, the discussion, the recommendation. Code everything else.

**Design for the worst-case submitter.** The best submitter is a thoughtful, experienced operator with thirty minutes of uninterrupted time. Design for the exhausted staff officer, forty-five minutes after the EXEVAL ends, on a tablet in a vehicle, who cannot remember whether "fires support" or "fire support coordination" is the correct functional area code. The controlled vocabulary must be organized to help that person find the right value quickly.

## 4-4. Vignette — The DEFENDER 25 Lessons-Learned Form

During DEFENDER 25, a G3 section designed a lessons-learned capture form with 23 fields, including 11 free-text fields. After the exercise, 312 entries were submitted from V Corps, 21st TSC, 10th AAMDC, 56th MDC-E, and SETAF-AF. Analysis revealed: - 47% of entries had at least one required field left blank or filled with "N/A" - 68% of the free-text "recommendation" fields contained a single sentence or fewer - The "responsible unit" field contained 34 distinct spellings of unit designations, making unit-based filtering unreliable - Less than 10% of entries had been tagged with a functional area code

The data was structurally present but operationally useless for aggregated analysis. A redesign for DEFENDER 26 reduced the form to 8 fields — 4 required constrained-vocabulary and 4 optional including a narrative text area — and produced 289 entries with a 94% required-field completion rate and consistent taxonomy coding.

**The lesson:** fewer, better-designed fields consistently outperform comprehensive but burdensome forms in operational knowledge capture.

---

## SECTION 5 — LESSONS LEARNED AS AN ONTOLOGY DESIGN PROBLEM

**BLUF:** A lessons-learned system is only as good as its data model. Design the Lessons Learned Object Type for the search and analysis scenarios that matter three years from now, not just for today's entry workflow.

### 5-1. The Standard Structure

A lessons-learned entry has a canonical structure in Army doctrine. The KM's job is to translate that structure into Foundry Object Type properties that support both capture and retrieval:

| Doctrinal Element | Object Property | Field Type | Why It Matters |
|---|---|---|---|
| Observation | `observation_narrative` | Free text | What was observed — the raw factual content |
| Discussion | `discussion_narrative` | Free text | Analysis of why it happened and what it means |
| Recommendation | `recommendation_narrative` | Free text | What should change |
| Unit | `reporting_unit_id` | Object link → Unit | Enables unit-based filtering and analysis |
| Operation | `operation_id` | Object link → Operation | Enables operation-based filtering |
| Date (observed) | `date_observed` | Date | Temporal filtering and trend analysis |
| Date (submitted) | `date_submitted` | Date (auto) | Audit trail |

| Doctrinal Element | Object Property | Field Type | Why It Matters |
|---|---|---|---|
| Functional area | `functional_area` | Enum | Critical for cross-unit search |
| Unit level | `unit_level` | Enum | Filters results by echelon relevance |
| Operation type | `operation_type` | Enum | Enables filtering by exercise type, combat operation, garrison |
| Status | `status` | Enum | Distinguish draft, published, implemented, superseded |
| Classification | `classification_level` | Enum | Required for access control |
| Source type | `source_type` | Enum | EXEVAL, live operation, tabletop, individual observation |

## 5-2. Designing for Future Search

Design the data model not by asking "what information do we want to capture?" but by asking "what queries will a G3 run in 2028 when preparing for an operation?"

Representative likely queries: - "Show me all fire support lessons from DEFENDER exercises at BCT level" - "What lessons from sustainment operations were actually implemented vs. just recorded?" - "Which units submitted the most lessons from the 2025 rotation period, and what functional areas did they cover?" - "What recommendations from the 2024 EUCOM exercise have been marked implemented?"

Each requires specific filterable fields: `functional_area`, `operation_type`, `unit_level`, `status`, `date_observed`. If those fields are not designed into the Object Type from the start, those queries are impossible without reading every record.

**The design test:** draft the ten most important consumer queries before finalizing the Object Type. Each query element must map to a property that will be consistently populated. If a query element cannot be mapped to a defined property, either add the property or accept that the query will not be possible.

## 5-3. Linked Object Design

Lessons learned do not exist in isolation. The Foundry Ontology supports relationships through Object Links. A well-designed lessons-learned system includes:

- **Lesson → Operation:** Links lessons to the operation or exercise where the observation was made
- **Lesson → Unit:** Links lessons to the reporting unit

- **Lesson → Previous Lesson:** Links lessons that supersede, relate to, or contradict each other; enables identifying where the same problem has been observed repeatedly
- **Lesson → SOP:** Links lessons that have been addressed by a specific SOP update; closes the loop between observation and doctrine change

Designing these links from the start costs relatively little. Retrofitting them after two years of data entry is expensive and incomplete.

### 5-4. Status Lifecycle

A lessons-learned entry should move through a defined status lifecycle, not remain in perpetual draft:

`DRAFT` → `SUBMITTED` → `VALIDATED` → `PUBLISHED` → `IMPLEMENTED` or `SUPERSEDED`

Each transition requires a defined authority (who can move an entry from VALIDATED to PUBLISHED?) and a defined action (does implementation require a linked SOP change? a unit commander endorsement?). The KM who does not define lifecycle transitions and authorities at design time will find that every entry sits in SUBMITTED status forever, because no one knows who is supposed to do what next.

---

## SECTION 6 — INSTITUTIONAL MEMORY IN A ROTATING FORCE

**BLUF:** USAREUR-AF operates with significant force rotation — units and personnel change on 12-month cycles, with key leaders often on shorter timelines. The KM's strategic function is ensuring that operational knowledge survives rotations. MSS makes this possible only if the architecture is designed to capture tacit knowledge in explicit form.

### 6-1. The Rotation Problem

The specific challenge is not that knowledge is unavailable — it is that knowledge walks out the door. An experienced targeting officer who has run DEFENDER three times carries enormous operational knowledge: which coordination mechanisms actually work, what breaks under the time pressure of the first 72 hours, which liaison relationships are essential and which are ceremonial. When that officer rotates, it leaves with them. The incoming officer starts with doctrine, which describes the model; the experienced officer's knowledge describes the deviation from the model that real operations require.

## 6-2. Tacit vs. Explicit Knowledge

Tacit knowledge is what people know but cannot easily articulate. Explicit knowledge has been written down, codified, and made transferable. The KM's challenge is not capturing explicit knowledge (that is a filing problem) — it is surfacing tacit knowledge and converting enough of it into explicit form to be useful to the next rotation.

**Practical KM tools for this:**

| Tool | Purpose |
|---|---|
| Structured AARs going beyond what happened to why it happened | The "why" column captures judgment, not just event sequence |
| Departure interview workflows | A defined process for capturing key lessons from departing personnel before they out-process — a knowledge capture event, not a personnel action |
| Decision record templates | For major decisions (why approach X over approach Y), a brief structured record captures the reasoning that will not survive in meeting minutes or email |
| Named expert identification | An expertise Object Type linking personnel to specific knowledge domains enables the organization to know who to ask — and to identify when that expertise is about to leave |

## 6-3. Making Knowledge Capture Feel Useful

The single biggest barrier to knowledge capture in a rotating force is motivational. Personnel who are out-processing, transitioning units, or managing heavy operational tempo have no immediate incentive to spend time on knowledge documentation.

**The design principle:** knowledge capture activities must be integrated into existing workflows, not added as separate overhead. An AAR already happens after every significant event — the KM designs the capture form so the AAR itself produces structured knowledge records, not a separate data entry step. A departure brief already happens at PCS — the KM designs a 30-minute knowledge capture session as part of the existing outprocessing checklist.

**The motivational lever that works:** make the system visibly useful to the person entering data. If a departing officer can see that the knowledge they are entering will be surfaced to their replacement — that their replacement will arrive with access to the lessons they learned rather than having to repeat the same mistakes — the activity has clear, tangible value. Build the retrieval interface before the capture system. Let people see what the product of their entry will look like before asking them to enter it.

### 6-4. Vignette — The V Corps RIP/TOA Knowledge Gap

A V Corps G3 section ran a retrospective analysis after a major unit rotation and found that the replacement unit repeated three of the top-five lessons identified by their predecessors in the first 60 days. All three lessons were documented — in a SharePoint folder, in AFTER ACTION REVIEW documents, as Word files with non-descriptive names. None of the three incoming unit staff officers had found or read the relevant documents.

The section designed an MSS-based knowledge handoff workflow: departing units submit a knowledge package (top-10 operational lessons, critical relationship map, known issues with pending resolution) as structured Object entries in MSS. Incoming units receive a tailored view of those entries, filtered to their functional responsibilities, delivered as a Workshop application available during the RIP/TOA week.

After the next rotation cycle, the retrospective found zero repeated lessons from the prior top-five list. The knowledge existed in both cases; the architecture was the difference. SETAF-AF adopted the same model for its Africa AOR rotational forces, where RIP/TOA cycles are shorter and the operational context — partner force relationships, austere logistics networks, security cooperation engagement history — is even harder to reconstruct without structured knowledge transfer.

## SECTION 7 — VERSION CONTROL AND RECORDS MANAGEMENT FOR KM PRODUCTS

**BLUF:** SOPs, policies, and reference documents change. Treating version control as an administrative detail rather than an architectural requirement produces knowledge bases filled with outdated content that can actively mislead future users.

### 7-1. The Document Lifecycle on MSS

| Phase | Description | Access | KM Action |
|---|---|---|---|
| DRAFT | Under development, not yet reviewed | Restricted (authors only) | Manage version numbers, track reviewers |
| REVIEW | Circulated for comment | Restricted (designated reviewers) | Track review status, consolidate comments |
| APPROVED | Reviewed and signed | Read (organization) | Publish, set effective date, notify stakeholders |
| CURRENT | Active governing document | Read (all authorized users) | Monitor for currency, schedule review cycle |

| Phase | Description | Access | KM Action |
|---|---|---|---|
| SUPERSEDED | Replaced by newer version | Archive (restricted) | Link to replacement, retain for audit trail |
| CANCELLED | Withdrawn without replacement | Archive (restricted) | Document cancellation authority and rationale |

The KM does not manage the content of SOPs and policy documents — that is the proponent's responsibility. The KM manages the lifecycle status, access controls, version history, and notification workflows.

### 7-2. The Supersession Problem

The most common records management failure: when SOP 2025-03 is replaced by SOP 2026-01, both documents continue to appear in search results unless the supersession is explicitly recorded and the old document is archived or restricted.

**The design requirement:** the Document Object Type must include `document_status` as a required field, and the workflow for publishing a new version must include an automated or semi-automated action to mark the previous version SUPERSEDED and link it to the replacement. This is not optional — it is the core of records management.

### 7-3. Branching for KM Products

Foundry supports branching — creating an isolated workspace to develop changes before committing to the main dataset. For KM products, branching enables:

- **Draft development without polluting the published knowledge base.** A draft SOP revision exists on a branch until approved, then merged to the main dataset with a status change to CURRENT.
- **Parallel review workflows.** Multiple reviewers see the draft without it being visible to general users on the main branch.
- **Change audit trail.** Branch history records who made what changes and when, supporting approval documentation requirements.

The KM who treats documents as blobs (upload the PDF, done) loses all of these capabilities. Documents managed as structured Objects with version properties and lifecycle status are the foundation of defensible records management.

### 7-4. Retention Policy Implementation

Every knowledge category requires a defined retention policy. General guidance flows from Army records management doctrine (AR 25-400-2, The Army Records Information Management System). The KM's job is to translate that guidance into system-level controls:

| Control | Description |
|---|---|
| Retention period | How long is this category retained? (Lessons learned: minimum 7 years after operation completion is a common standard — confirm with G6/legal for your specific category) |
| Retention trigger | What starts the retention clock? (Date of final approval, date of operation conclusion, date of last access) |
| Disposition action | What happens at end of retention period? (Permanent archive, deletion, transfer to ARIMS) |
| Review cycle | How often is content reviewed for currency before end of retention? (Annual review for active SOPs; before each major exercise for exercise-specific lessons) |

Building these as Object properties and automated alerts means the system surfaces content due for review or disposition rather than requiring manual tracking.

---

## SECTION 8 — WORKFLOW DESIGN — MOVING KNOWLEDGE THROUGH THE ORGANIZATION

**BLUF:** Capturing knowledge is necessary but not sufficient. Knowledge must move through the organization — from observation to validated lesson, from draft SOP to approved document. Workflow design is where knowledge management becomes a change management problem.

### 8-1. What Workflow Design Actually Is

A workflow in KM context is the defined sequence of human decisions and system actions that moves a knowledge artifact from raw input to published institutional record. In MSS/Foundry, workflows are implemented using Action configurations, conditional logic, notifications, and status transitions on Object Types.

The hard part is organizational: defining who has the authority to move an artifact to the next stage, what criteria must be met for each transition, and what the system does when a workflow item is stuck. A workflow that is technically correct but organizationally underdefined will stall.

### 8-2. A Lessons-Learned Workflow — Anatomy

| Stage | Actor | Action | System Result |
|---|---|---|---|
| **1 — Submission** | Any authorized user | Submits entry via Workshop form | Creates Lessons Learned Object with `status = SUBMITTED`; sends notification to KM |

| Stage | Actor | Action | System Result |
|---|---|---|---|
| **2 — KM Review** | KM | Reviews for completeness, correct taxonomy coding, appropriate classification | If incomplete or miscoded: returns to submitter with comments. If complete: advances to `VALIDATED`, routes to functional area proponent |
| **3 — Proponent Review** | Functional area proponent (G-staff section owning that functional area) | Reviews for accuracy and operational relevance | Approves for publication, requests modification, or rejects with rationale; approval advances status to `APPROVED FOR PUBLICATION` |
| **4 — Publication** | KM | Sets `status = PUBLISHED`, sets `publication_date`, confirms taxonomy coding, makes record visible in enterprise search | Record visible to all authorized users |
| **5 — Implementation Tracking** | Assigned proponent | Implements the recommendation by a target date; provides implementation evidence | Final status transition to `IMPLEMENTED`, `NOT IMPLEMENTED — ACCEPTED RISK`, or `SUPERSEDED BY NEW LESSON` |

## 8-3. Workflow Design as Change Management

The workflow above requires a 37F or designated KM to review every submission, requires G-staff sections to have a named proponent for each functional area, and requires someone to own the implementation tracking loop. None of those roles may exist before the workflow is designed. **The KM must negotiate the organizational design before building the technical implementation.**

The practical approach: design the workflow, then brief to leadership with the explicit ask — "This workflow requires a named proponent in each G-staff section. Each proponent needs to plan for approximately two hours per month of review time. Do I have leadership support to designate those proponents?" The answer determines whether the workflow is viable.

## 8-4. Designing for Workflow Failure

| Mechanism | Purpose |
|---|---|
| **Escalation timers** | If a submission has been in Stage 2 for more than seven days without action, the workflow sends an escalation notification to the KM's supervisor |
| **Backup reviewer designation** | Each proponent role has a primary and an alternate. If the primary is unavailable, submissions route to the alternate. |

| Mechanism | Purpose |
|---|---|
| **Dormant item reports** | A weekly automated report surfaces all submissions that have been in the same stage for more than the defined service level period |
| **Audit trail** | Every status transition is logged with timestamp and actor. If a submission is stuck, the KM can identify exactly where and when it stalled. |

These mechanisms are the system's self-monitoring capability. A workflow without them will fill with stalled items that no one notices until someone asks why nothing has been published in three months.

## SECTION 9 — COMMON KM FAILURE MODES ON MSS

**BLUF:** Knowledge management failures on MSS are mostly predictable and mostly preventable.

| Failure Mode | Root Cause | Prevention |
|---|---|---|
| **The Repository No One Uses** | The system was not integrated into the actual workflow where knowledge is generated and consumed. The lessons-learned form exists but is not part of the AAR process. The SOP repository exists but staff officers still share SOPs by email. | Design knowledge capture into the existing workflow, not alongside it. If MSS is optional, it will not be used. |
| **The Form That Generates Garbage** | Form was designed by asking "what information would we like to have?" rather than "what information will actually be entered, by the real people who will submit it, under the real conditions they operate in?" | Pilot the form with actual submitters before deployment. Watch the pilot session. Count how long it takes. Note which fields cause confusion. Remove fields that consistently produce bad data. |
| **Knowledge That Cannot Be Found** | The taxonomy fields that would support filtering were not included in the data model, were optional and left mostly blank, or were designed for capture convenience rather than retrieval use cases. | Design for retrieval before designing for capture. Draft the ten most important queries the system must support. Test the query against sample data during design — not after two years of entry. |
| **The System** | The system was designed for knowledge capture but not for | Assign a named owner to every knowledge category at design time. Build automated alerts that surface content |

| Failure Mode | Root Cause | Prevention |
|---|---|---|
| **That Fills With Outdated Content** | knowledge lifecycle management. Retention policy was not built in. Review cycles were not automated. No one has ownership of ongoing governance. | approaching end of its review period. Include `last_reviewed_date` and `review_due_date` as properties on long-lived Objects. |
| **KM as a One-Time Build** | Knowledge management was treated as a project rather than a function. No transition plan, no documented maintenance procedures, no designated successor. | The final deliverable of any KM build is not the system — it is the governance plan. Write it before the system goes live. It specifies: who owns each knowledge category, the maintenance schedule, how to update the taxonomy when organizational changes occur, how to train the next KM, and the escalation path when the system encounters problems outside the maintainer's capability to resolve. |

## SUMMARY — THE KNOWLEDGE MANAGER'S DESIGN CHECKLIST

Before beginning any MSS knowledge management build, verify:

**Architecture** - ☐ Four architecture questions answered in writing (types, creators, consumers, retention) - ☐ Taxonomy designed from consumer search use cases, not creator entry convenience - ☐ Object Type properties include all fields required for the ten priority queries - ☐ Free text reserved for genuinely narrative content; all codable fields are coded

**Capture** - ☐ Form piloted with actual submitters in realistic conditions - ☐ Required fields verified to be consistently completable - ☐ Controlled vocabulary organized for usability under time pressure - ☐ AIP extraction (if used) has defined human review step before Ontology write

**Lifecycle** - ☐ Document status lifecycle defined with explicit transition authorities - ☐ Supersession workflow: publishing new version automatically marks prior version superseded - ☐ Retention periods defined per knowledge category, aligned with AR 25-400-2 - ☐ Review cycle triggers and responsible owners assigned at design time

**Workflow** - ☐ All non-KM workflow roles identified and organizational agreement secured - ☐ Escalation timers and backup designation configured for each workflow stage - ☐ Dormant item reporting configured - ☐ Every status transition logged with timestamp and actor

**Continuity** - ☐ Governance plan written before system goes live - ☐ Named owner for each knowledge category - ☐ Maintenance documentation complete enough for a new KM to operate the system after a two-week handoff - ☐ Departure knowledge capture process integrated into unit outprocessing workflow

## CURRICULUM NOTES

**Prerequisite:** SL 3 (Advanced Builder) is REQUIRED — not recommended. The KM who has not designed Ontology models and Workshop applications cannot effectively architect knowledge systems on MSS.

**Advanced track:** SL 4K graduates should pursue **SL 5K (Advanced Knowledge Manager)** for advanced topics including enterprise-scale knowledge taxonomy design, cross-command knowledge federation, NATO Lessons Learned Database (LLDB) integration, and AI-assisted knowledge synthesis at theater level.

**Peer specialist cross-references:** - **SL 4L (Software Engineer):** High-frequency coordination point. The KM designs the knowledge ontology and pipeline architecture; the SWE implements pipelines requiring custom code. Review KM ontology design with SL 4L before build begins — ontology changes have direct impact on downstream pipeline code. - **SL 4H (AI Engineer):** Coordinate on AIP-assisted knowledge extraction and summarization workflows. The KM defines the knowledge architecture, human-review requirements, and taxonomy validation standards; the AI Engineer implements the AIP Logic pipeline. Every AIP-extracted knowledge output requires KM-defined SME review before Ontology write. - **SL 4G (ORSA):** Coordinate when knowledge metrics (capture rates, lesson implementation rates, gap analysis) require quantitative decision products rather than descriptive monitoring dashboards.

**WFF awareness:** KMs on MSS build institutional memory systems for WFF-qualified users (SL 4A through SL 4F — Intelligence, Fires, Movement and Maneuver, Sustainment, Protection, and Mission Command). Each WFF function generates domain-specific lessons, TTPs, and SOPs. Design knowledge ontology taxonomies with WFF functional areas as primary filter dimensions — a G3 (Operations/Mission Command, SL 4F) staff officer and a fires coordinator (SL 4B) have different knowledge retrieval needs that a single undifferentiated repository cannot serve.

**KM-specific governing references:**

| Publication | Title | Relevance |
|---|---|---|
| ATP 6-01.1 | Techniques for Effective Knowledge Management | The Army's dedicated KM publication; covers KM solution design, content management, and KM SOPs |
| AR 25-400-2 | Army Records Management Program (Oct 2022) | Records lifecycle management, ARIMS compliance, electronic records |
| DA PAM 25-403 | Army Guide to Recordkeeping (Nov 2022) | Implementing guidance for AR 25-400-2; electronic records, scheduling, transfer |

| Publication | Title | Relevance |
|---|---|---|
| NATO Core Metadata Specification / STANAG 5636 | Structured Metadata Standards | NATO-standard metadata schema — required for coalition-compatible knowledge products |
| NATO ADatP-34 / NISP | C3 Interoperability Standards | Data cataloging and knowledge exchange standards for NATO interoperability |

**NOTE — New Doctrine Content in SL 4K:** SL 4K now includes FM 6-0 KM doctrine (5-step process, 3 KM tasks, 6 tool categories mapped to MSS, section 1-5b), FM 6-22 three developmental domains (Institutional/Operational/Self-Development), and FM 3-57 Civil Knowledge Integration as a doctrinal validation of the KM function.

*This guide is a prerequisite companion to SL 4K. Read in full before beginning SL 4K task instruction. The mental models developed here underpin every task in SL 4K and are not repeated in the task manual.*

*For questions about this guide, contact C2DAO, Wiesbaden, Germany.*