DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

TECHNICAL MANUAL

# SL 4K

USAREUR-AF

## TM-40K — MAVEN SMART SYSTEM (MSS)

*Specialist Course Manual*

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

DRAFT — NOT FOR OFFICIAL USE. FOR TRAINING PLANNING PURPOSES ONLY.

**26 MARCH 2026**

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

# TM-40K — MAVEN SMART SYSTEM (MSS)

**Forward:** This manual trains Army Knowledge Managers (37F MOS) and staff officers in KM roles to design, build, and sustain knowledge management systems on the Maven Smart System (MSS). KMs are responsible for ensuring the formation's institutional knowledge is captured, organized, findable, and usable — not simply stored. **Prereqs:** SL 1, Maven User; SL 2, Builder; SL 3, Advanced Builder; Data Literacy Technical Reference (all required); CONCEPTS_GUIDE_TM40K_KNOWLEDGE_MANAGER (read before this manual). *HQ USAREUR-AF · v1.0 · 2026 · DISTRIB: USG only · AUTH: C2DAO/UDRA v1.1*

**WARNING**

Knowledge management systems that fail silently are more dangerous than systems that fail loudly. A broken dashboard is obvious. A knowledge repository that captures incomplete lessons, routes to wrong audiences, or returns stale doctrine will not announce its failure — it will quietly degrade the formation's institutional memory over months and years. Build with validation checkpoints at every stage.

## CHAPTER 1 — INTRODUCTION: THE KNOWLEDGE MANAGER ROLE ON MSS

### 1-1. Knowledge Manager Specialist Manual

**BLUF:** This manual trains Army Knowledge Managers (37F MOS) and staff officers in KM roles to design, build, and sustain knowledge management systems on the Maven Smart System (MSS). KMs are responsible for ensuring the formation's institutional knowledge is captured, organized, findable, and usable — not simply stored.

This manual covers knowledge architecture design, AAR and lessons learned systems, AIP-assisted knowledge workflows, search and discovery interfaces, doctrine version management, personnel expertise mapping, and PCS/ETS knowledge transfer. It does not cover general Foundry platform operation (see SL 1 through SL 3) or developer-level coding (see SL 4G, SL 4H).

**SL 4K covers** knowledge ontology design: Object Types for documents, lessons, AARs, SOPs, expertise profiles; AAR capture systems using Workshop forms and structured data pipelines; lessons learned ingestion, deduplication, tagging, and routing to relevant units; AIP Logic for document summarization,

lesson extraction, content tagging, and knowledge Q&A; search and discovery: Quiver and Contour configurations, Workshop browse interfaces; doctrine and SOP currency tracking, change workflows, unit-level versioning; personnel expertise Object Types, SME identification, and expertise gap analysis; knowledge transfer workflows for PCS/ETS transitions, key person dependency mapping; and knowledge metrics: capture rates, access patterns, gap analysis.

**SL 4K does NOT cover** general Workshop application construction — see SL 2 and SL 3; Python or PySpark coding — see SL 4G, SL 4H; SQL query writing — see SL 4H; AIP Agent Studio development — see SL 4H; TypeScript or OSDK development — see SL 4L; or data pipeline construction from scratch — see SL 3 (UI) or SL 4L (code).

**NOTE**

SL 4K graduates design knowledge systems that outlast their assignments. Every architecture decision should be evaluated against the question: "Will the Soldier replacing me be able to understand, maintain, and extend this system after a two-week handoff?"

## 1-2. Curriculum Position, Advanced Track, and WFF Context

**Prerequisite:** SL 3 (Advanced Builder) is REQUIRED — not recommended. SL 1 and SL 2 are assumed complete.

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

**Peer specialist tracks:** The KM has significant overlap with the Software Engineer (SL 4L) on knowledge ontology design and pipeline implementation. Coordinate with SL 4L when knowledge capture or routing workflows require custom code beyond Pipeline Builder's no-code capability. Coordinate with SL 4H (AI Engineer) when designing AIP-assisted knowledge extraction and summarization workflows — the KM defines the knowledge architecture and human review requirements; the AI Engineer implements the AIP Logic pipeline. Coordinate with SL 4G (ORSA) when knowledge metrics (capture rates, gap analysis, lesson implementation rates) require quantitative analysis products.

**WFF awareness:** KMs on MSS serve as institutional memory architects for WFF-qualified personnel (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 requiring tailored ontology design and retrieval interfaces. Design knowledge taxonomies with WFF functional areas as primary filter dimensions. The WFF-aligned staff section is both a primary contributor to and a primary consumer of MSS knowledge products.

## 1-3. The KM Role in USAREUR-AF

USAREUR-AF is the Army Service Component Command (ASCC) to United States European Command (USEUCOM) and United States Africa Command (USAFRICOM), executing theater land operations across the European and African AOR, integrating with NATO Allied Command Operations (ACO), and supporting Joint All-Domain Command and Control (JADC2). The formation conducts major exercises — Combined Resolve, Defender Europe, Swift Response — in rotation, generating large volumes of after-action data, lessons learned, and TTPs each cycle.

The institutional challenge: USAREUR-AF has a persistently high PCS tempo. Hard-won knowledge from Combined Resolve rotations departs with the personnel who participated. Each new exercise cohort risks relearning lessons already paid for. Knowledge Managers are the formation's defense against this institutional forgetting.

**The KM role in the USAREUR-AF data chain:**

```
OPERATIONAL EXPERIENCE (Exercise, Operation, Garrison)
              |
              v
    KNOWLEDGE CAPTURE (KM)          <- You are here
    (AAR systems, lessons pipelines,
     SOP workflows, expertise profiles)
              |
              v
    KNOWLEDGE REPOSITORY (MSS)
    (Structured Foundry objects, tagged,
     linked, searchable, version-controlled)
              |
              v
    KNOWLEDGE DISSEMINATION (KM)    <- Also you
    (Search interfaces, AIP Q&A,
     routing to relevant units)
              |
              v
    OPERATIONAL APPLICATION
    (Planner uses prior lessons, Soldier
     finds current SOP, commander finds SME)
```

**KM roles in USAREUR-AF that use MSS:**

| Role | Primary MSS KM Functions |
|---|---|
| 37F KMO (Division/Corps) | Full knowledge architecture design, AIP workflow config, enterprise repo management |
| S3 KM NCO (Brigade/Battalion) | AAR capture, lessons routing, SOP currency tracking |
| S6 Data NCOIC | Knowledge system integration with data pipelines, search indexing |

| Role | Primary MSS KM Functions |
|---|---|
| G2 KM Analyst | Operational lessons, TTP extraction, classification-appropriate handling |
| Civil Affairs KM Coordinator | Partner nation knowledge sharing, lessons exchange workflows |

## 1-4. Knowledge Architecture Responsibilities

The KM's core architecture responsibility is defining and maintaining the knowledge ontology — the set of Object Types, Link Types, and properties that give the formation's institutional knowledge a consistent, queryable structure.

Without a knowledge ontology, MSS becomes a file share with search. With a well-designed ontology, it becomes a system where a planner can ask: "What lessons from Combined Resolve 25 are relevant to an air assault operation in complex terrain, and who in the formation has done it before?" and receive a structured, citable answer.

**The four domains of KM architecture on MSS:**

| Domain | Covers | Key Object Types |
|---|---|---|
| Operational knowledge | AARs, lessons learned, TTPs, exercise reports | `Lesson`, `AAR`, `Exercise`, `TTP` |
| Procedural knowledge | SOPs, checklists, regulations, doctrine | `SOP`, `DoctrinePub`, `Checklist`, `PolicyDoc` |
| Personnel knowledge | Skills, expertise, past assignments, qualifications | `ExpertiseProfile`, `Qualification`, `Assignment` |
| Organizational knowledge | Unit history, lineage, structure, relationships | `UnitRecord`, `KnowledgeDomain`, `SMERelationship` |

Each domain requires its own Object Types, property schemas, Link Types, and access controls. Chapter 2 covers the full design methodology.

## 1-5. Governing References

| Document | Relevance |
|---|---|
| FM 7-0 (Training) | AAR process, training management, lessons learned integration |
| AR 25-1 (Army Enterprise Technology Management) | Information management policies, records management alignment |

| Document | Relevance |
|---|---|
| USAREUR-AF C2DAO Guidance | Theater-level architecture standards for MSS knowledge products |
| NATO STANAG 4778 (Lessons Learned) | Standardized format for NATO-compatible lessons learned exchange |
| learn-data.armydev.com | Army data literacy training portal — reference for self-directed follow-on training |
| 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 |
| DA PAM 25-1-1 (Army IT Implementation Instructions) | Ch 4 covers data management discipline and data governance structure |
| NATO Core Metadata Specification / STANAG 5636 | Structured metadata standards for NATO-compatible knowledge products |
| NATO ADatP-34 / NISP | C3 interoperability standards for data cataloging and knowledge exchange |
| STANAG 5643 (proposed) — MIM Governance Standard | NATO MIP Information Model governance — data model versioning, change proposals, national extensions |
| FM 6-0 (Commander and Staff Organization and Operations, May 2022) | Ch 5 defines the Army KM framework: 5-step process, 3 KM tasks, 6 tool categories |
| FM 6-22 (Leader Development, Oct 2015) | Three developmental domains (institutional, operational, self-development) — governs how KM products support leader growth |
| FM 3-57 (Civil Affairs Operations, Jul 2021) | Civil Knowledge Integration (CKI) — doctrinal analog to the KM function across Army integrating processes |

### 1-5a. Strategic Guidance

The following are strategic guidance documents — not doctrine — that inform MSS training design and operational context.

| Document | Authority | Relevance |
|---|---|---|
| Army CIO Data Stewardship Policy (April 2, 2024) | Army CIO | Data product standards, domain ownership, stewardship hierarchy |

| Document | Authority | Relevance |
|---|---|---|
| UDRA v1.1 (February 2025) | Army Enterprise | Unified Data Reference Architecture — federated governance, domain structure |
| DoD Data Strategy (2020) | OSD | VAULTIS-A framework (supersedes VAUTI) — knowledge products must be Visible, Accessible, Understandable, Linked, Trusted, Interoperable, Secure, Auditable. 8 dimensions per DDOF Playbook v2.2 (Dec 2025); 85% weighted avg = DDOF Phase 3 quality gate. |
| NATO Data Strategy for the Alliance (Feb 2025) | NATO | Alliance-wide data governance mandate — governs coalition knowledge sharing in EUCOM AOR |

**NOTE**

Knowledge management on MSS intersects with records management requirements under AR 25-400-2. Coordinate with your unit RMO (Records Management Officer) before standing up any persistent knowledge repository to ensure retention schedules are applied correctly.

## 1-5b. Doctrinal KM Frameworks

**BLUF:** Army doctrine defines the KM function in FM 6-0, Ch 5. Every MSS knowledge system the KM builds should map to this doctrinal framework. This section establishes the doctrinal foundation; subsequent chapters apply it to MSS.

### FM 6-0 Knowledge Management Process (May 2022, Ch 5)

FM 6-0 defines knowledge management as a five-step continuous process:

| Step | Action | MSS Platform Equivalent |
|---|---|---|
| 1 — Assess | Determine knowledge gaps, user needs, and current state | Quiver/Contour analytics on existing knowledge objects; `KnowledgeGap` Object Type |
| 2 — Design | Define knowledge architecture, workflows, and governance | Ontology Manager — Object Types, Link Types, property schemas (Ch 2 of this manual) |
| 3 — Develop | Build knowledge products, capture tools, and repositories | Pipeline Builder, Workshop forms, AIP Logic workflows (Chs 3–5) |
| 4 — Pilot | Test with a limited user group; validate usability and accuracy | Staging environment deployment, user acceptance testing (Section 2-7) |
| 5 — Implement | Field the system; train users; establish sustainment plan | Production deployment, user training via SL 1/SL 2, sustainment SOPs (Ch 9) |

FM 6-0 further defines three core KM tasks:

1. **Create Knowledge.** Transform raw data and information into actionable knowledge through analysis, synthesis, and context. On MSS: AIP-assisted summarization, lesson extraction pipelines, AAR structured capture.
2. **Retain Knowledge.** Archive, label, and identify knowledge products so they survive personnel turnover. On MSS: Ontology-enforced tagging, controlled vocabularies, `reviewDate` lifecycle tracking, version-controlled SOPs.
3. **Transfer Knowledge.** Move knowledge to the right person at the right time. On MSS: search interfaces (Quiver/Contour), AIP Q&A agents, role-based routing workflows, PCS/ETS handoff packages.

FM 6-0 identifies six categories of KM tools. The table below maps each to MSS capabilities:

| FM 6-0 KM Tool Category | Definition | MSS Equivalent |
|---|---|---|
| Information Systems | Systems that store and process data | Foundry datasets, Ontology, Pipeline Builder |
| Collaboration Tools | Tools enabling shared work and discussion | Workshop applications, Slate dashboards |
| Data-Analysis Tools | Tools for examining data to extract meaning | Contour, Code Workbook, AIP Logic |
| Search & Discovery Tools | Tools for locating knowledge products | Quiver, Contour filters, Workshop browse views |
| Expertise-Location Tools | Tools for identifying who knows what | `ExpertiseProfile` Object Type, personnel search apps (Ch 8) |
| Expertise-Development Tools | Tools for growing organizational knowledge | MSS training courses (SL 1 through SL 5), self-study addenda |

**NOTE (FM 6-22 — Three Developmental Domains):** FM 6-22 (Leader Development) defines three domains in which leaders grow: **Institutional** (schoolhouse training), **Operational** (unit application and OJT), and **Self-Development** (individual study). MSS knowledge products must support all three. Instructor-led courses (SL 1 through SL 5) serve the institutional domain. Unit application of MSS during exercises and operations serves the operational domain. Self-study addenda and self-paced materials serve the self-development domain. A KM who designs knowledge products for only one domain leaves two-thirds of the force development model unsupported. Source: FM 6-22, Ch 1.

**NOTE (FM 3-57 — Civil Knowledge Integration):** Civil Knowledge Integration (CKI) — the process of analyzing, evaluating, and organizing collected civil information for operational relevance — is a doctrinal validation of the KM function. CKI integrates through all five Army integrating processes: intelligence preparation of the operational environment (IPOE), information collection, targeting, risk management,

and knowledge management (FM 3-57). KMs should recognize CKI as a peer discipline that validates the same organize-contextualize-disseminate pattern applied across the KM enterprise. Civil Affairs KM Coordinators (see Section 1-3, Table) use CKI methodology when building MSS knowledge products for civil information. Source: FM 3-57.

### 1-6. TM-40K Chapter Guide

| Chapter | Task Area | When You Need It |
|---|---|---|
| Chapter 2 | Knowledge Architecture Design | Starting a new KM system from scratch |
| Chapter 3 | AAR Capture Systems | Exercise or operation cycle planning |
| Chapter 4 | Lessons Learned Pipelines | Multi-source lessons ingestion and routing |
| Chapter 5 | AIP-Assisted Knowledge Work | Automating summarization, tagging, Q&A |
| Chapter 6 | Search and Discovery | Building findable knowledge interfaces |
| Chapter 7 | Doctrine and SOP Management | SOP versioning, doctrine currency |
| Chapter 8 | Personnel Expertise Mapping | SME identification, skills gap analysis |
| Chapter 9 | Knowledge Transfer and Continuity | PCS/ETS season, key person risk |

## CHAPTER 2 — KNOWLEDGE ARCHITECTURE DESIGN

**NOTE — Palantir Developers reference:** *Palantir Ontology Overview* — A foundational overview of the Palantir Ontology model — Object Types, Link Types, and property design — that underpins all knowledge architecture work in this chapter. KMs designing a knowledge ontology should review this before the design task in Section 2-7. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

**NOTE — Palantir Developers reference:** *Product Launch: Ontology Foundations | DevCon 5* — Covers the latest Palantir approach to ontology design and the foundational principles that govern Object Type schema decisions. Directly reinforces the knowledge ontology design methodology in Sections 2-2 through 2-7. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

## 2-1. Purpose

**BLUF:** Before building any knowledge system on MSS, design the ontology. Object Types, properties, and Link Types defined without a deliberate architecture will require painful rework after data has been captured. Design once, build once, maintain continuously.

This chapter covers the full methodology for designing a knowledge ontology on Foundry: identifying knowledge domains, defining Object Types and their properties, establishing Link Types that make knowledge navigable, and applying access control that matches the sensitivity of the content.

## 2-2. Knowledge Domain Analysis

Before opening Foundry, define the knowledge domains your system will serve. A knowledge domain is a bounded area of institutional knowledge with a defined owner, a defined consumer population, and a defined lifecycle.

**Conduct a knowledge domain analysis using the following steps:**

**Step 1 — Identify knowledge consumers.** Who needs to find and use this knowledge? (Examples: planners, S3 staff, incoming personnel, partner nations, training audience.) Different consumers have different search behaviors, classification access, and format preferences.

**Step 2 — Identify knowledge producers.** Who generates the knowledge your system will capture? (Examples: AARs from exercise participants, TTPs from G3 operations, SOPs from unit leadership, lessons from CALL submissions.) Producers determine your ingestion design.

**Step 3 — Identify knowledge lifecycle.** How long is this knowledge valid? SOPs change with policy. Lessons learned remain relevant for years. Personnel expertise is updated with each assignment. Define a review cycle for each domain.

**Step 4 — Identify sensitivity level.** Will any content be classified or controlled unclassified? Sensitivity determines access control design before any objects are created.

**Step 5 — Identify the domain owner.** Who is responsible for quality and currency in this domain? Every knowledge domain requires a named owner accountable for content accuracy. No owner, no production deployment.

**CAUTION**

Building a knowledge repository without defined domain ownership creates an abandoned system. Ownerless knowledge repositories accumulate stale content, are never reviewed, and actively mislead users who assume the content is current. Do not deploy to production without a named owner and a documented review cycle.

## 2-3. Core Knowledge Object Types

The following Object Types form the foundation of a USAREUR-AF knowledge ontology. Adapt properties to your specific use case; do not deviate from the core type structure without C2DAO coordination, as these types support cross-unit and enterprise search.

### Table 2-1. Core Knowledge Object Type Definitions

| Object Type | Purpose | Key Properties |
|---|---|---|
| `Lesson` | A single extracted lesson — an observation, insight, or recommendation derived from operational experience | `lessonID`, `title`, `observation`, `discussion`, `recommendation`, `sourceEvent`, `classification`, `approvalStatus`, `relevantMOS`, `operationalDomain`, `tags`, `dateExtracted`, `reviewDate` |
| `AAR` | A structured after-action review record, linked to an exercise or event | `aarID`, `eventName`, `eventType`, `dateOfEvent`, `unit`, `participants`, `sustainActions`, `improveActions`, `correctiveActions`, `classification`, `status` |
| `Exercise` | An exercise or training event that generates knowledge products | `exerciseName`, `exerciseType`, `startDate`, `endDate`, `hostNation`, `participatingUnits`, `AOR`, `JCETnumber` |
| `SOP` | A standing operating procedure — a living document tracked for currency | `sopID`, `title`, `owningUnit`, `version`, `effectiveDate`, `reviewDate`, `status`, `classification`, `functionalArea`, `supersedes` |
| `DoctrinePub` | An Army or joint doctrine publication referenced by the unit | `pubNumber`, `title`, `proponent`, `currentVersion`, `publishDate`, `changeDate`, `relevantAreas` |
| `TTP` | A tactic, technique, or procedure — operationally derived, more granular than doctrine | `ttpID`, `title`, `description`, `conditions`, `standards`, `sourceLesson`, `validatedBy`, `classification`, `operationalDomain` |
| `ExpertiseProfile` | A personnel expertise record — skills, qualifications, and knowledge areas for an individual | `personID`, `rank`, `MOS`, `name` (see Privacy Act note), `skills`, `qualifications`, `pastAssignments`, `languages`, `certifications`, `availableAsAdvisor` |
| `KnowledgeGap` | A documented gap in the formation's knowledge base — a known unknown | `gapID`, `description`, `domain`, `identifiedDate`, `priority`, `mitigationPlan`, `owner` |

**NOTE (Privacy Act):** The `ExpertiseProfile` Object Type captures PII (name, rank, assignment history). Before deploying this Object Type to any system, obtain Privacy Act System of Records Notice (SORN) guidance from your unit legal officer. At minimum: restrict access to authorized users, do not expose this Object Type to any coalition-facing or MPE-accessible interface, and document the legal basis for collection.

## 2-4. Core Link Types

Link Types make knowledge navigable. Without links, objects are isolated records. With links, a user can traverse from an `Exercise` to all `AAR` records it generated, to all `Lesson` objects extracted from those AARs, to the `TTP` objects derived from those lessons, to the `ExpertiseProfile` objects of personnel who have applied those TTPs.

### Table 2-2. Core Knowledge Link Type Definitions

| Link Type | Source | Target | Cardinality | Description |
|---|---|---|---|---|
| `generatedAAR` | `Exercise` | `AAR` | One-to-Many | Exercise produced these AARs |
| `extractedLesson` | `AAR` | `Lesson` | One-to-Many | Lesson derived from this AAR |
| `informedTTP` | `Lesson` | `TTP` | Many-to-Many | Lesson contributed to this TTP |
| `referencedIn` | `DoctrinePub` | `SOP` | Many-to-Many | Doctrine publication cited in SOP |
| `supersedes` | `SOP` | `SOP` | One-to-One | New version replaces prior version |
| `hasExpertIn` | `ExpertiseProfile` | `KnowledgeGap` | Many-to-Many | Person has expertise relevant to gap |
| `addressedBy` | `KnowledgeGap` | `TTP` | Many-to-Many | TTP addresses this knowledge gap |
| `participatedIn` | `ExpertiseProfile` | `Exercise` | Many-to-Many | Person participated in this exercise |
| `relevantTo` | `Lesson` | `SOP` | Many-to-Many | Lesson suggests update to this SOP |

## 2-5. Property Schema Design

Property schemas must balance completeness (capturing all useful context) against capture burden (complexity that causes Soldiers to skip fields or enter low-quality data). Follow these principles:

**Required vs. optional fields.** Define a minimum required set. Every `Lesson` must have a title, observation, and recommendation at minimum. Optional fields enrich searchability but must not block submission.

**Controlled vocabularies.** Use enumerated value sets for tag fields wherever possible. Free-text tags are inconsistent and unsearchable. Define `operationalDomain` as an enumeration: [Air Assault, Sustainment, CBRN, Fires, Intelligence, C2, Communications, Medical, Engineer, Cyber, Other]. Define `eventType` as: [Exercise, JCET, Operation, Garrison Training, Staff Ride, Wargame].

**Date management.** Every knowledge object requires both a `dateCreated` (auto-populated by Foundry) and a `reviewDate` (set by the KM at creation, triggers a workflow review at expiry). Stale knowledge is worse than no knowledge — reviewDate enforcement is non-negotiable.

**Classification fields.** Every object type in a classification-aware knowledge system requires a `classification` property: enumeration of [UNCLASSIFIED, CUI, SECRET, NATO SECRET, RELEASABLE TO [country codes]]. Access control on Foundry datasets enforces this; the property provides a visible audit trail and supports filtering in search interfaces.

**CAUTION**

Property schemas, once adopted and populated with data, are difficult to change. Adding a property is safe. Renaming, removing, or changing the type of an existing property breaks downstream pipelines and search indexes. Design property schemas carefully before any data is loaded. Use a staging environment and a test dataset to validate the schema before production deployment.

**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 |
| Sandbox | `[sandbox] Name` | Training exercises, KM experimentation |

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

## 2-6. Access Control Design

Knowledge systems in USAREUR-AF operate across a spectrum of classification and releasability. Design access control at the dataset level in Foundry, not at the application level.

**Access control layers for knowledge systems:**

| Layer | Mechanism | KM Responsibility |
|---|---|---|
| Dataset permissions | Foundry role-based access on underlying datasets | Coordinate with C2DAO; document access groups |
| Object Type visibility | Ontology-level permissions on Object Types | Configure in Foundry Ontology admin |
| Workshop application | Application-level role configuration | Limit application access to authorized user groups |
| Coalition/MPE | MPE environment separation | No coalition access to any knowledge system without explicit C2DAO approval and classification review |

**Standard access groups for USAREUR-AF knowledge systems:**

| Group | Access Level | Notes |
|---|---|---|
| KM-Admin | Read/Write all objects, manage schemas | KMO and designated deputies only |
| KM-Contributor | Create and edit objects in assigned domain | Exercise participants, SOP authors |
| KM-Consumer | Read-only, all unclassified objects | All USAREUR-AF users |
| KM-Sensitive | Read access to CUI objects | Named personnel, approved by KMO |

## 2-7. Task: Design a Knowledge Ontology

**CONDITIONS:** You are establishing a new knowledge management system on MSS for a Brigade Combat Team preparing for Combined Resolve. You have completed domain analysis and have C2DAO coordination approval.

**STANDARDS:** A complete knowledge ontology design package includes: domain map, Object Type definitions with all properties, Link Type map with cardinality, access control design, and a named owner for each domain.

**EQUIPMENT:** Foundry Ontology Editor (SL 3 access required), C2DAO coordination memo, domain analysis worksheet.

**PROCEDURE:**

1. Open Foundry and navigate to the Ontology Editor.

2. Create a new Ontology branch named: `km-bcteam-[unit]-[year]-design`. Never design directly on the main branch.
3. Create each Object Type defined in your domain analysis. Start with `Lesson`, `AAR`, and `Exercise` — these are the highest-priority types for exercise preparation.
4. For each Object Type, add all required properties first. Mark required properties as required in the schema editor.
5. Add optional properties. For enumerated fields, define the value set in the property configuration.
6. Add `reviewDate` as a required property to every Object Type. Set a default review cycle (lessons: 2 years; SOPs: 1 year; TTPs: 18 months; ExpertiseProfile: 6 months).
7. Create Link Types per Table 2-2. Verify cardinality is correctly set for each link.
8. Submit the Ontology branch for C2DAO review before merging to main.
9. Document the access control design in the knowledge system architecture document. Submit to the unit KM authority for signature.
10. After C2DAO approval and signature, merge the branch and configure dataset permissions per the access control design.

> **NOTE**
>
> Step 8 is not optional. Ontology changes that affect shared enterprise search indexes or downstream AIP Logic workflows require architecture review. A well-designed ontology submitted through proper channels is approved quickly. An ad hoc ontology deployed without review will be rolled back.

# CHAPTER 3 — AAR CAPTURE SYSTEMS

## 3-1. Purpose

**BLUF:** Structured AAR capture converts the verbal after-action review process (FM 7-0) into persistent, queryable data. The capture system must be fast enough to use in the field, structured enough to enable search and analysis, and simple enough that Soldiers actually complete it.

This chapter covers Workshop form design for AAR input, structured data capture standards, automated pipeline routing, and quality validation workflows.

## 3-2. AAR Process Integration

The FM 7-0 AAR process defines four questions: 1. What was supposed to happen? 2. What actually happened? 3. Why were there differences? 4. What will we sustain or improve?

The MSS AAR capture system provides digital infrastructure for this process — it does not replace the human conversation. The conversation happens first. The system captures the output. KMs who attempt to run AARs through a form without the facilitated discussion first will capture structured garbage.

**Integration points between FM 7-0 AAR and MSS capture:**

| FM 7-0 Step | MSS Integration | Timing |
|---|---|---|
| Plan the AAR | Facilitator creates Exercise/Event object in MSS | Before event |
| Prepare participants | Pre-populate event context in capture form | 24-48 hrs before AAR |
| Conduct the AAR | Discussion is human; scribe captures key points | During AAR |
| Document results | Facilitator/scribe completes Workshop form | Immediately post-AAR |
| Distribute results | Automated routing pipeline (Chapter 4) | Within 24 hrs of submission |

## 3-3. AAR Workshop Form Design

The AAR capture form in Workshop must balance completeness and speed. An AAR form that takes more than 15 minutes to complete will not be completed in the field. Design for the minimum required data first; use optional fields for depth.

**Core AAR form sections:**

**Section 1 — Event Identification (auto-populated or quick select)** - Event name (linked to Exercise object if pre-created) - Date of event - Unit (auto-populated from user profile) - Location/grid (optional — country, grid square if applicable) - Event type (enumerated: Tactical Movement, Breach, Air Assault, Logistics, C2 Exercise, etc.)

**Section 2 — Participants** - Number of personnel involved (numeric) - Lead element (free text or linked ExpertiseProfile for key leaders) - Observer/Controller (if applicable — link to ExpertiseProfile or free text)

**Section 3 — Sustain Actions** - Free text entry, minimum 1 item required - Structured prompt: "What worked? Why did it work? Repeat this action." - KM note: Soldiers default to vague sustain entries ("Good comms"). Train facilitators to push for specific, replicable descriptions.

**Section 4 — Improve Actions** - Free text entry, minimum 1 item required - Structured prompt: "What did not work? Why? What specifically would improve it?"

**Section 5 — Corrective Actions** - For each Improve Action: who owns it, what action they will take, by when - At least one corrective action required per improve item before submission

**Section 6 — Lesson Extraction Flag** - Toggle: "Does this AAR contain a lesson for broader distribution?" (Yes/No) - If Yes: brief description of the potential lesson (populates the lessons-learned pipeline in Chapter 4)

**Section 7 — Classification** - Required selection: UNCLASSIFIED / CUI / SECRET - If SECRET is selected: warning banner displayed, submission routed to classified handling workflow

**WARNING**

Do not submit classified AAR content to an UNCLASSIFIED system. If an exercise generates AARs with classified observations, the capture form on the unclassified system must capture only unclassified content. A separate classified capture workflow must be established for SECRET content. Classify based on the most sensitive element in the entry, not the overall event classification.

## 3-4. Form Validation Logic

Configure Workshop form validation to enforce data quality at the point of capture. Validation logic applied at entry is far cheaper than data cleaning after the fact.

**Required validations on AAR submission:**

| Field | Validation Rule | Error Message |
|---|---|---|
| Event date | Cannot be in the future; cannot be older than 90 days | "Event date must be within the last 90 days. Contact KMO for older entries." |
| Unit | Must match a valid unit code from the unit registry object | "Unit not recognized. Select from the unit list." |
| Sustain actions | Minimum 1 entry, minimum 10 characters | "Enter at least one sustain action with a full description." |
| Improve actions | Minimum 1 entry, minimum 10 characters | "Enter at least one improve action with a full description." |
| Corrective actions | At least one improve action must have a corrective action with a named owner | "Each improve action requires a corrective action and owner." |
| Classification | Required selection, no default | "Select classification before submitting." |

**CAUTION**

Do not set restrictive validation rules before piloting with actual users. Overly strict validation (e.g., requiring 50-character minimum entries) causes workarounds — users pad entries with nonsense text to pass validation. Pilot with the intended user population, observe behavior, and calibrate validation rules against actual submission patterns.

---

## 3-5. AAR Pipeline: Submission to Object Creation

When a Soldier submits an AAR form in Workshop, a Pipeline Builder workflow converts the form submission into structured Foundry objects. This pipeline is designed at SL 3 level and implemented at SL 4L level if custom logic is required.

**Standard AAR pipeline stages:**

**Stage 1 — Intake:** Form submission lands in a staging dataset. All fields captured, submission timestamp added, submitter user ID logged.

**Stage 2 — Validation:** Pipeline checks all required fields. Invalid records flagged with error codes, routed to the KM review queue in Workshop. Valid records proceed.

**Stage 3 — Object Creation:** Valid submission creates an `AAR` object in the Ontology. Object links to the `Exercise` object (matched by event name/date). Submitter ExpertiseProfile linked via `participatedIn` link.

**Stage 4 — Lesson Flag Check:** If the lesson extraction flag is set to Yes, the pipeline creates a pending `Lesson` object with status = "Draft" and routes it to the lessons-learned pipeline (Chapter 4).

**Stage 5 — Corrective Action Routing:** For each corrective action with a named owner, the pipeline creates a task notification. If the unit uses an integrated task management workflow, tasks are pushed there. Otherwise, an email notification is triggered.

**Stage 6 — Aggregation Update:** The Exercise object's summary properties (total AARs submitted, sustain count, improve count) update via a scheduled aggregation transform.

**NOTE**

Stage 5 corrective action routing requires coordination with the unit's task management system. If no integration exists, a simple Workshop "My Corrective Actions" view filtered by submitter provides minimum viable tracking without a full integration.

---

### 3-6. Mobile and Field Considerations

USAREUR-AF exercises take place across the European AOR, often with limited connectivity. AAR capture forms must be designed for degraded conditions.

**Design requirements for field-deployable AAR forms:**

- Form must function on standard government tablet or laptop (not smartphone-dependent)
- Where connectivity is available, submit in real time
- Where connectivity is limited, design a local-save / batch-upload workflow using an MSS-approved offline method. Coordinate with S6 for approved data handling
- Forms should not require more than 3G-equivalent bandwidth to load
- Pre-populate all lookup fields (unit codes, event names) before deploying to the field — do not rely on live lookups against large datasets during the exercise

**CAUTION**

Do not design AAR capture workflows that depend on real-time AIP processing in the field. AIP Logic calls require connectivity and introduce latency. Capture first; process after connectivity is restored.

## CHAPTER 4 — LESSONS LEARNED PIPELINES

### 4-1. Purpose

**BLUF:** Lessons learned are only valuable when they reach the personnel who can act on them. A pipeline that captures lessons but cannot route them to the right unit at the right time adds no operational value. This chapter covers multi-source ingestion, deduplication, tagging, approval workflows, and targeted distribution.

### 4-2. Lessons Learned Sources in USAREUR-AF

USAREUR-AF generates lessons learned from multiple sources simultaneously. A mature lessons pipeline ingests all sources into a common structure.

**Table 4-1. Lessons Learned Sources**

| Source | Format | Volume | Ingestion Method |
|---|---|---|---|
| MSS AAR capture (Chapter 3) | Structured Foundry objects | Medium — exercise-driven | Native (pipeline stage 4) |
| CALL (Center for Army Lessons Learned) | PDF reports, structured web exports | Medium — continuous | SL 4L pipeline required |
| Unit-submitted lessons (email/form) | Unstructured text, email attachments | Variable | Workshop intake form + manual KM review |
| Exercise observer/controller reports | Word/PDF documents, OC/T structured formats | High during exercises | Batch ingest pipeline |
| NATO LLDB (Lessons Learned Database) | STANAG 4778 formatted records | Low — periodic | Coordinate with CA/POLAD for access |
| JLLIS (Joint Lessons Learned Info System) | Structured web export | Medium | SL 4L pipeline required; coordinate with J7 |
| Partner nation reports | Variable format, variable language | Variable | Manual ingest with AIP translation assist (Chapter 5) |

**NOTE**

CALL and JLLIS ingestion pipelines require SL 4L developer involvement for automated connectors. Until connectors are established, the KM uses a manual ingest workflow: download reports, use the AIP-assisted extraction workflow (Chapter 5) to structure content, submit via the Workshop intake form for KM review and approval.

## 4-3. Lessons Learned Data Model

Before building an ingestion pipeline, confirm the `Lesson` Object Type schema (Section 2-3) is finalized and deployed. The pipeline must produce objects that conform exactly to the schema.

**Standard lesson object lifecycle:**

```
SUBMITTED (from any source)
      |
      v
DRAFT (pending KM review)
      |
      v
IN REVIEW (KM has assigned for SME review)
      |
      v
APPROVED (SME and KM confirmed)
```

```
      |
      v
PUBLISHED (available to all authorized consumers)
      |
      v
ARCHIVED (past review date, no longer current)
```

Every `Lesson` object must include the current lifecycle status as a property. Status transitions trigger routing workflows (Section 4-5).

---

## 4-4. Deduplication

Multiple ingestion sources generate duplicate lessons. An AARs from three different units covering the same Combined Resolve event will likely produce similar lessons. Without deduplication, the repository accumulates redundant records that dilute search quality and inflate coverage metrics.

**Deduplication approach:**

**Step 1 — Exact match detection.** Pipeline hashes the `observation` field. Exact matches (same hash) are flagged as duplicates; the later-arriving record is linked to the original and marked duplicate-suppressed, not deleted. The link preserves evidence that multiple units observed the same issue.

**Step 2 — Semantic near-match flagging.** AIP Logic can flag near-duplicate lessons for KM review. When two lessons have high semantic similarity (AIP-assessed), both enter the KM review queue with a "Potential Duplicate" flag. The KM decides: merge, keep separate, or mark as corroborating. This is KM judgment — do not auto-merge on semantic similarity.

**Step 3 — Cross-source deduplication.** A lesson from CALL that is substantively identical to a unit-generated lesson should be linked, not merged. The CALL version carries external validation. Link them with a `corroborates` Link Type and mark the CALL version as the authoritative source.

> **NOTE**
>
> Deleted lessons are unrecoverable in most production configurations. Always suppress, not delete. A suppressed duplicate preserves the evidence of multi-unit observation — which is itself analytically valuable (how many units observed this issue?). Hard deletes are reserved for lessons containing errors that could cause harm if accessed.

---

## 4-5. Tagging and Classification

Lessons are only findable if they are well-tagged. Tagging at ingestion is the single highest-leverage point in the knowledge management workflow. Poor tags at ingest compound over time — retroactive retagging of thousands of lessons is a project, not a task.

**Required tags on every `Lesson` object:**

| Tag Field | Type | Values |
|---|---|---|
| `operationalDomain` | Enum | Air Assault, Sustainment, Fires, C2, Intel, Communications, Medical, Engineer, Cyber, CBRN, Logistics, Other |
| `relevantMOS` | Multi-select list | MOS codes (11B, 25U, 91B, etc.) — select all applicable |
| `exerciseOrEvent` | Link | Link to `Exercise` object |
| `tacticalLevel` | Enum | Team/Squad, Platoon, Company, Battalion, Brigade, Division, Corps, Theater |
| `operationalEnvironment` | Multi-select | Complex Terrain, Urban, Cold Weather, High Altitude, Degraded Comms, Peer Threat, CBRN Contaminated |
| `classification` | Enum | UNCLASSIFIED, CUI, SECRET |
| `lessonType` | Enum | Sustain, Improve, New TTP, Doctrine Gap, Equipment Issue, Training Gap |

**AIP-assisted tagging (Chapter 5):** AIP Logic can suggest tags based on lesson text. KM reviews and confirms suggestions before object is published. AIP tagging suggestions are inputs to KM judgment, not replacements for it.

---

## 4-6. Distribution Routing

A lesson approved and published must reach the units and personnel for whom it is relevant. Passive distribution (available if searched) is necessary but not sufficient. Active routing ensures high-priority lessons reach target audiences without requiring them to search.

**Routing logic for published lessons:**

**Rule 1 — Unit routing:** If `tacticalLevel` is Brigade or below, route to all units of that type currently in USAREUR-AF. Routing delivers a notification to the unit KM and adds the lesson to the unit's "New Lessons" queue in Workshop.

**Rule 2 — MOS routing:** For lessons tagged with specific MOS codes, route to all personnel with those MOS codes who are opted into the lessons notification system. This requires ExpertiseProfile integration (Chapter 8).

**Rule 3 — Exercise follow-on routing:** For each upcoming exercise in the system, a scheduled query identifies all lessons whose tags match the exercise type and operational environment. These lessons are bundled into a pre-exercise knowledge package and pushed to participating units 30 days before the exercise.

**Rule 4 — SOP update flag:** If a lesson is tagged as `lessonType = Doctrine Gap` or `lessonType = Training Gap`, it is automatically added to the SOP review queue for the relevant functional area (Chapter 7). The SOP owner receives a notification.

> **NOTE**
>
> Routing rules require Workshop Action configuration and, for notification integration, coordination with the S6 on messaging system integration. Start with the pre-exercise bundle (Rule 3) as the highest-value routing workflow — it has a clear trigger event, a defined audience, and measurable impact on exercise preparation.

---

## 4-7. Task: Configure a Lessons-Learned Intake Pipeline

**CONDITIONS:** The unit has completed an exercise and has 40+ pending AAR-flagged lessons in Draft status. You are the Brigade KMO. The `Lesson` Object Type and AAR pipeline (Chapter 3) are deployed and functional.

**STANDARDS:** All 40 lessons are reviewed, tagged, deduplicated where appropriate, and in Approved or Archived status within 5 business days of exercise completion. Distribution routing is configured for the next exercise cycle.

**EQUIPMENT:** Foundry Ontology Editor, Pipeline Builder (SL 3 access), Workshop (SL 2 access), KM Review Queue Workshop application.

**PROCEDURE:**

1. Open the KM Review Queue application. Filter by `status = Draft` and `sourceEvent = [exercise name]`.
2. For each Draft lesson: a. Review the observation, discussion, and recommendation fields for completeness. b. Confirm all required tags are populated. Add missing tags. c. Check for duplicates using the duplicate flag queue. For flagged potential duplicates, compare and decide: merge, suppress, or corroborate. d. Assign to the relevant domain SME for review. Change status to `IN REVIEW`.
3. Monitor the In Review queue. Follow up with SMEs who have not responded within 48 hours.
4. For each SME-reviewed lesson: review SME comments. If approved with no changes, set status to `APPROVED`. If revisions required, return to contributor.
5. For Approved lessons: verify the classification field is correct. Set status to `PUBLISHED`.
6. Verify that routing rules fired correctly: check the "New Lessons" queues for target units and confirm notifications were delivered.
7. For any lessons tagged as `Doctrine Gap` or `Training Gap`: confirm the SOP review notification was delivered to the correct SOP owner. Log the notification in the lesson's `correctiveActions` field.

8. Run the coverage report (Chapter 6) to verify distribution metrics are updating correctly.

# CHAPTER 5 — AIP-ASSISTED KNOWLEDGE WORK

## 5-1. Purpose

**NOTE — Palantir Developers reference:** *AIP with Jeg: Adding RAG to a Simple Notes Application* — A step-by-step walkthrough of building a retrieval-augmented generation (RAG) pipeline, directly applicable to the AIP Logic document summarization and Q&A configurations in this chapter. Good procedural companion for KMs configuring their first AIP knowledge workflow. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

**NOTE — Palantir Developers reference:** *Building with Palantir AIP: Logic Tools for RAG/OAG* — Covers the Logic tools available within AIP for retrieval-augmented and ontology-augmented generation pipelines, reinforcing the AIP Logic workflow configurations in Sections 5-3 and 5-4. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

**NOTE — Palantir Developers reference:** *Building with Palantir AIP: Data Tools for RAG/OAG* — Covers data preparation tools for RAG pipelines — the data-side complement to the Logic tools reference above. Review alongside Section 5-3 when configuring the document summarization input pipeline. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

**NOTE — Palantir Developers reference:** *Deep Dive: Advanced Agent Reasoning with Knowledge Nodes x First Solar | DevCon 4* — Demonstrates how AIP Knowledge Nodes enable agents to reason over structured organizational knowledge, directly relevant to AIP-assisted knowledge Q&A design in Section 5-4. Shows a real deployment of knowledge-node-based agent reasoning at production scale. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

**BLUF:** AIP Logic significantly accelerates knowledge management tasks — document summarization, lesson extraction from unstructured text, content tagging, and knowledge-base Q&A. AIP is a force multiplier for a KM managing high-volume lesson pipelines. AIP is not a replacement for KM judgment, domain expertise, or SME validation.

## 5-2. AIP Logic in the Knowledge Context

AIP Logic on MSS provides natural language AI capabilities connected to the Foundry Ontology. For knowledge management, the primary AIP use cases are:

| Use Case | AIP Capability | Human Role |
|---|---|---|
| Document summarization | Summarize uploaded PDFs or AAR text into structured key points | Review and edit summary before attaching to object |
| Lesson extraction | Parse an unstructured AAR document and suggest structured lesson objects | Review each suggested lesson, edit, approve/reject |
| Content tagging | Suggest tags for a lesson based on its text | Accept, modify, or reject tag suggestions before publishing |
| Near-duplicate detection | Flag semantically similar lessons for KM review | Make the merge/keep/corroborate decision |
| Knowledge Q&A | Answer natural language questions against the knowledge repository | Verify source citations; treat answers as starting points, not final answers |
| Translation assist | Summarize or translate partner nation documents (unstructured) | Review translations with a bilingual SME for accuracy |

**WARNING**

AIP outputs on sensitive or classified topics require SME review before any action is taken. AIP can misclassify operational context, misinterpret military-specific terminology, or hallucinate citations. A KM who publishes AIP-generated lesson content without review is not managing knowledge — they are propagating untested AI output as institutional knowledge. Review is non-negotiable.

## 5-3. Configuring AIP Logic for Document Summarization

Use this configuration when you receive an unstructured document (CALL report, exercise report, partner nation after-action summary) and need to convert it to structured knowledge objects.

**Prerequisites:** AIP Logic access (SL 3 qualified), uploaded document in Foundry Files or connected dataset, `Lesson` Object Type deployed.

**Configuration steps:**

1. In Foundry, open AIP Logic and create a new workflow: `[UnitName]-Document-Summarizer`.
2. Connect the input: configure the trigger as "When a document is uploaded to [knowledge-intake] dataset."
3. Add the Summarization step:
4. Prompt template: "You are a knowledge management assistant for a U.S. Army unit. Summarize the following document and extract discrete lessons learned. For each lesson, identify: (1) the observation — what happened; (2) the discussion — why it happened; (3) the recommendation — what should be done differently or repeated. Format each lesson as a numbered list entry with these three

components clearly labeled. If a lesson is unclear, flag it as 'Requires KM Review' rather than fabricating detail. Document text: {document_text}"

5. Output type: Structured JSON with lesson array
6. Add the Object Creation step: for each lesson in the output array, create a `Lesson` object with `status = Draft`, `sourceType = AIP-Extracted`, `requiresReview = true`.
7. Add the Review Queue step: route all created Draft lessons to the KM Review Queue application.
8. Test with a non-sensitive exercise report from a previous cycle. Validate that extracted lessons match what a human KM would extract. Adjust the prompt if extraction quality is poor.
9. Document the workflow configuration, prompt version, and test results. Save to the workflow's documentation object.

**CAUTION**

Prompt changes to AIP workflows in production require testing in a staging environment first. A prompt change that causes the AIP to misformat output will flood the review queue with malformed objects. Version-control your prompts (include a version number in the prompt text) and test before deploying changes.

---

## 5-4. Configuring AIP Logic for Knowledge Q&A

A knowledge Q&A interface allows users to ask natural language questions against the knowledge repository. This is the highest-visibility AIP knowledge application — it makes the repository's value immediately apparent to users who would otherwise never search it.

**Architecture of the knowledge Q&A workflow:**

```
USER QUESTION (Workshop interface)
        |
        v
AIP LOGIC WORKFLOW
  |- Classify intent: question type (doctrine lookup, lesson search, SME
identification, etc.)
  |- Query relevant Object Types based on intent
  |- Retrieve matching objects and their linked objects
  |- Synthesize answer from retrieved context
  |- Include citations: which Lesson/SOP/TTP IDs were used
        |
        v
ANSWER + CITATIONS (displayed in Workshop)
        |
        v
USER FEEDBACK (Did this answer your question? Yes/No)
        |
```

```
        v
FEEDBACK DATASET (used by KM to improve retrieval quality)
```

**Key configuration requirements:**

**Grounding:** The AIP workflow must be configured to answer only from retrieved knowledge objects — not from AIP's general training knowledge. Configure the system prompt: "Answer the user's question using only the knowledge objects provided below. If the answer is not contained in the provided objects, respond: 'I do not have a lesson or document that addresses this question. Consider submitting a knowledge gap report.' Do not speculate. Cite the object ID for every claim you make."

**Citation enforcement:** Every AIP answer must include the IDs of the source objects used. This enables the user to navigate to the source and the KM to verify answer quality.

**Feedback loop:** Implement a thumbs-up/thumbs-down feedback widget on every Q&A response. Route negative feedback to the KM review queue with the question text and answer. KMs review weekly — a pattern of negative feedback on a topic indicates either a knowledge gap (no good lessons exist) or a retrieval quality problem (lessons exist but are not being found).

---

## 5-5. Configuring AIP Logic for Automatic Tagging

Consistent tagging drives retrieval quality. AIP-assisted tagging reduces the burden on contributors while improving tag completeness.

**Configuration for AIP tag suggestion:**

1. Create an AIP Logic workflow: `[UnitName]-Tag-Suggester`.
2. Trigger: "When a new `Lesson` object is created with `status = Draft`."
3. Prompt template: "You are a tagging assistant for an Army knowledge management system. Read the following lesson and suggest tags. Available tag values are listed below. Select only from the available values — do not create new tags. If you are uncertain about a tag, leave it blank rather than guessing. Lesson text: {observation} {discussion} {recommendation}. Available operationalDomain values: [list]. Available relevantMOS values: [list]. Available tacticalLevel values: [list]. Available operationalEnvironment values: [list]. Available lessonType values: [list]. Return your suggestions as JSON."
4. Action: Populate `suggestedTags` property on the Lesson object with AIP output. Do not overwrite existing tags.
5. In the KM Review Queue application, display the `suggestedTags` alongside the empty tag fields. KM clicks "Accept" to copy suggested tags to the actual tag fields, or manually enters tags.
6. Log accepted vs. modified vs. rejected tag suggestions to a feedback dataset for quality tracking.

---

## 5-6. AIP Prompt Quality Standards

Prompt quality directly determines AIP output quality. KMs are responsible for prompt design and maintenance.

**Table 5-1. AIP Prompt Quality Checklist**

| Criterion | Requirement |
|---|---|
| Role definition | Prompt must define the AI's role in the first sentence ("You are a knowledge management assistant...") |
| Scope constraint | Prompt must limit the AI to a defined scope ("Answer only from provided documents...") |
| Uncertainty instruction | Prompt must direct AI behavior when uncertain ("If you are not sure, flag for review rather than guessing") |
| Output format | Prompt must specify output format (JSON, numbered list, paragraph with citations) |
| Military terminology | Prompt must define key military terms used in the domain (e.g., "TTPs are Tactics, Techniques, and Procedures — operationally derived guidance more specific than doctrine") |
| Version tracking | Prompt must include a version number in a comment line at the top |

## 5-6a. Document Intelligence for Knowledge Extraction (GA Q1 2026)

**BLUF:** AIP Document Intelligence — generally available as of Q1 2026 — automates the document parsing, chunking, and embedding pipeline that KMs previously configured manually or coordinated with AI engineers (SL 4H) to build. Document Intelligence allows KMs to ingest document collections into the knowledge repository with structured, searchable, AI-retrievable content without custom code.

**What Document Intelligence provides to KMs:**

| Capability | KM Application |
|---|---|
| Automated document parsing | Extract text from uploaded AARs, SOPs, doctrine PDFs, exercise reports, and partner nation documents without manual conversion |
| Configurable chunking | Split extracted text into coherent passages that align with knowledge retrieval needs — chapter-level for SOPs, observation-level for AARs |
| Platform-managed embedding | Embed document chunks for semantic similarity search without sourcing, maintaining, or validating a custom embedding model |
| Integrated retrieval | AIP Logic workflows (Sections 5-3 and 5-4) and Agent Studio agents can query Document Intelligence directly for relevant passages |

**Impact on KM knowledge workflows:**

Document Intelligence changes the document ingestion phase of the lessons learned pipeline (Chapter 4) and the AIP-assisted knowledge workflows in this chapter. Previously, ingesting an unstructured document into the knowledge repository required either (a) manual extraction by a KM or (b) coordination with a SL 4H AI engineer to build a custom parsing and embedding pipeline. Document Intelligence reduces this to a configuration task.

**Updated document ingestion pattern:**

```
DOCUMENT SOURCE (CALL report, exercise AAR, doctrine PDF)
        |
        v
UPLOAD TO FOUNDRY FILES
        |
        v
AIP DOCUMENT INTELLIGENCE
  |- Parse document into structured text
  |- Chunk text into coherent passages
  |- Embed passages for semantic retrieval
  |- Index in managed vector store
        |
        v
AIP LOGIC WORKFLOW (Section 5-3 Summarization / Section 5-5 Tagging)
  |- Retrieve relevant chunks from Document Intelligence
  |- Summarize, extract lessons, suggest tags
  |- Create Draft Lesson objects with status = Draft, requiresReview = true
        |
        v
KM REVIEW QUEUE (unchanged — KM validates all extracted content)
```

**Configuration guidance:**

1. Coordinate with the MSS platform team to confirm Document Intelligence is activated on your enrollment.
2. Upload document collections to a designated Foundry Files location (one per knowledge domain recommended).
3. Configure Document Intelligence to index the target file collections. Select chunking strategy appropriate to document type: paragraph-level for AARs (captures individual observations), section-level for SOPs (preserves procedural context).
4. Update existing AIP Logic summarization workflows (Section 5-3) to use Document Intelligence as the retrieval source instead of raw file text injection.
5. Update the Knowledge Q&A workflow (Section 5-4) to include Document Intelligence as a retrieval source alongside ontology object queries.

**NOTE**

Document Intelligence supplements — it does not replace — the structured knowledge ontology (Chapter 2). Parsed and embedded document chunks enable AI-assisted retrieval and summarization. Structured `Lesson`, `SOP`, and `TTP` objects in the Ontology remain the authoritative knowledge records. Document Intelligence feeds the pipeline; the Ontology stores the validated output. KMs must continue to review, validate, and publish extracted content as structured knowledge objects.

**CAUTION**

AIP-generated content from Document Intelligence retrieval requires the same human review standard defined in Section 5-2. Automated parsing does not equal validated knowledge. Every extracted lesson, summary, or tag suggestion must pass through the KM Review Queue before publication.

# CHAPTER 6 — SEARCH AND DISCOVERY SYSTEMS

**NOTE — Palantir Developers reference:** *Building with Palantir AIP: Semantic Search* — Covers the semantic search capabilities within Palantir AIP that form the foundation of knowledge retrieval applications. Review before building the AIP Q&A widget in Section 6-6, as semantic search is the underlying retrieval mechanism. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

**NOTE — Palantir Developers reference:** *Build with AIP: Semantic Search* — A more recent implementation reference for semantic search, covering updated API patterns. Use alongside the reference above when configuring the knowledge browser retrieval layer. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

**NOTE — Palantir Developers reference:** *Building with Palantir AIP: Advanced Search* — Covers advanced search patterns for document retrieval, including ranking, filtering, and hybrid search approaches. Relevant to the Workshop browse interface design in Section 6-3 and the AIP Q&A system in Section 6-6. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

## 6-1. Purpose

**BLUF:** Knowledge captured but not findable is knowledge wasted. Search and discovery design is not an afterthought — it is the primary way users access knowledge and the primary value proposition of the knowledge management system. Build search interfaces before finalizing the knowledge architecture; test findability from the beginning.

## 6-2. Search Architecture Overview

MSS provides multiple tools for knowledge discovery. Each serves a different search behavior:

| Tool | Search Behavior | Best For |
|---|---|---|
| Workshop (filtered list view) | Browse with structured filters | Users who know the category but not the exact item |
| Workshop (free text search widget) | Full-text search across object properties | Users who remember a key phrase or term |
| Quiver | Structured aggregation, drill-down analysis | KMs analyzing coverage, patterns, and gaps |
| Contour | Multi-dimensional exploration, pivot analysis | Analysts examining lesson patterns across exercises and domains |
| AIP Q&A (Chapter 5) | Natural language question answering | Users who want an answer, not a list of documents |

A complete knowledge discovery system uses all five modalities. Users approach knowledge with different questions; design for the full range.

## 6-2a. Core Object Views (GA February 2026)

**BLUF:** Core Object Views — generally available as of February 2026 — provide standardized, platform-managed views for browsing and exploring Ontology objects. Core Object Views give KMs a consistent, out-of-the-box interface for knowledge browsing and exploration without building custom Workshop applications for every Object Type.

**What Core Object Views provide:**

| Capability | KM Application |
|---|---|
| Standard object detail view | Consistent display of object properties, linked objects, and action buttons across all knowledge Object Types — no custom Workshop page required per type |
| Configurable property layout | KMs define which properties appear, their display order, and grouping — applied platform-wide to every surface that renders the object |
| Linked object navigation | One-click traversal from a Lesson to its source AAR, related TTPs, associated SOPs, and relevant SME profiles — the link chain from Section 6-3 rendered automatically |

| Capability | KM Application |
|---|---|
| Inline action execution | Users can execute configured Actions (flag for review, save to collection, suggest SOP update) directly from the object view without navigating to a separate application |
| Consistent cross-application experience | The same object view renders in Workshop, Agent Studio responses, search results, and notification links — users learn one layout |

**Impact on knowledge browser design:**

Core Object Views change the build strategy for the knowledge browser described in Section 6-3 and the task in Section 6-6. Previously, KMs built custom detail views in Workshop for each knowledge Object Type. Core Object Views provide the detail view as a platform-managed component, reducing Workshop configuration effort and ensuring consistency.

**Updated knowledge browser architecture:**

| Component | Before Core Object Views | With Core Object Views |
|---|---|---|
| Results list (center panel) | Custom Workshop Object List widget | Custom Workshop Object List widget (unchanged) |
| Detail view (right panel) | Custom Workshop property display, manually configured per Object Type | Core Object View — configure once per Object Type, rendered automatically |
| Link chain navigation | Custom Link Explorer widgets, manually connected | Automatic linked object rendering in Core Object View |
| Actions (flag, save, suggest update) | Custom Action buttons wired to each detail page | Actions configured in Core Object View, rendered inline |
| Filter panel (left panel) | Custom Object Set Filter widgets | Custom Object Set Filter widgets (unchanged) |

**Configuration guidance:**

1. For each knowledge Object Type (`Lesson`, `AAR`, `SOP`, `TTP`, `ExpertiseProfile`), configure a Core Object View in the Ontology Manager.
2. Define the property layout: group properties by function (identification, content, metadata, governance). Place the most-used properties at the top.
3. Configure the linked objects section: prioritize the Lesson-AAR-Exercise-TTP-SOP link chain. Display linked objects in the order users most commonly traverse them.
4. Register Actions on each Core Object View: "Flag for Review," "Save to Unit Collection," and type-specific actions (e.g., "Suggest SOP Update" on Lesson views).
5. Test the Core Object View by navigating to a published object from multiple surfaces — Workshop, search results, Agent Studio response links — and confirm consistent rendering.

6. Update the knowledge browser application (Section 6-6) to use Core Object Views for the detail panel instead of custom property display widgets. Retain the custom filter panel and results list.

> **NOTE**
>
> Core Object Views do not replace the Workshop knowledge browser application. The browser's filter panel, results list layout, and AIP Q&A integration remain custom Workshop components. Core Object Views replace the per-Object-Type detail panel, reducing maintenance and ensuring that every surface where a knowledge object appears renders the same view. Design the Core Object View as the single source of truth for how a knowledge object is displayed.

## 6-3. Workshop Browse Interface Design

The Workshop knowledge browser is the primary interface for most users. Design it for two primary behaviors: browsing (I want to explore lessons in a domain) and known-item retrieval (I know what I'm looking for).

**Standard knowledge browser layout:**

**Left panel — Filter pane:** - Operational domain (multi-select) - Tactical level (multi-select) - Lesson type (multi-select) - MOS relevance (multi-select) - Date range (date picker — event date, not capture date) - Classification (radio — show based on user's clearance) - Exercise/event (searchable dropdown linked to Exercise objects) - Status (default: Published only; admin users can see Draft/In Review)

**Center panel — Results list:** - Each result card: Title, operational domain tag(s), tactical level, date, lesson type icon - Result count with current filter state displayed - Sort options: Most recent, Most accessed, Highest-rated (requires feedback widget) - Pagination — 20 results per page maximum; large result sets degrade usability

**Right panel — Detail view (on selection):** - Full lesson content: observation, discussion, recommendation - All tags displayed - Link chain: Source AAR → Source Exercise → Related TTPs → Related SOPs - "Suggest SOP Update" action (routes to Chapter 7 SOP review workflow) - "Mark as relevant to my unit" action (adds to unit bookmark collection) - "This lesson needs review" flag action (routes to KM review queue)

> **NOTE**
>
> The right panel link chain is the highest-value navigation feature in the knowledge browser. A planner reviewing a lesson about air assault sustainment failures should be one click away from the SOP that the lesson suggests updating and one click from the SME in the formation who has relevant expertise. Design the link traversal as a first-class navigation feature, not a footnote.

## 6-4. Quiver Knowledge Coverage Dashboard

The Quiver knowledge coverage dashboard gives the KMO an operational picture of the knowledge repository: what is covered, what is missing, and where quality is declining.

**Standard KMO coverage dashboard panels:**

**Panel 1 — Lesson volume by domain (bar chart):** - X-axis: operational domain - Y-axis: count of Published lessons - Second series: count of Draft/In Review (pending publication) - Interpretation: imbalances indicate collection gaps in specific domains

**Panel 2 — Lesson age heatmap (grid):** - X-axis: operational domain - Y-axis: lesson age buckets (0-6 months, 6-12 months, 1-2 years, 2-5 years, 5+ years) - Color: red (5+ years, likely stale) to green (0-6 months, current) - Interpretation: red cells in high-priority domains indicate knowledge refresh requirement

**Panel 3 — Review date expiry timeline (line chart):** - X-axis: calendar (next 18 months) - Y-axis: count of lessons expiring per month - Alert threshold: months with expiry > 20 lessons flagged in red - Purpose: allows KM to plan review workload before deadlines

**Panel 4 — AIP Q&A feedback trend (line chart):** - X-axis: calendar (last 90 days) - Y-axis: percent positive feedback on AIP Q&A responses - Trend below 60% positive triggers KM review of retrieval quality

**Panel 5 — Knowledge gap tracker (table):** - All active `KnowledgeGap` objects - Columns: domain, priority, age, mitigation status, owner - Sort by priority descending

---

## 6-5. Contour Analysis for Lessons Learned Patterns

Contour provides multi-dimensional analysis that Quiver's dashboard format cannot. Use Contour for post-exercise analysis and trend identification.

**Standard Contour analyses for KMs:**

**Analysis 1 — Cross-exercise lesson recurrence:** - Pivot: lesson title keywords vs. exercise name - Identifies lessons that recur across multiple exercises — these are systemic issues, not one-time events, and require escalation to doctrine or training authority

**Analysis 2 — MOS coverage gap analysis:** - Pivot: MOS code vs. lesson count - MOS codes with zero or low lesson coverage are knowledge gaps — either the MOS is not participating in AAR processes or their lessons are not being captured at the right level

**Analysis 3 — Tactical level distribution:** - Histogram: lessons by tactical level - Healthy distribution: lessons at all levels from Team to Brigade. Skew toward Company/Battalion suggests higher-level knowledge is not being captured

**Analysis 4 — Lesson-to-TTP conversion rate:** - Count of Approved lessons vs. count of linked TTPs - Low conversion rate indicates a pipeline failure between lessons and TTPs — either the TTP derivation process is not functioning or the link is not being created

---

## 6-6. Task: Build a Knowledge Browser Application

**CONDITIONS:** The `Lesson`, `AAR`, `Exercise`, `TTP`, and `SOP` Object Types are deployed. You have SL 2 Workshop builder qualifications and SL 3 advanced Workshop design qualifications.

**STANDARDS:** A functional knowledge browser application with full filter panel, results list, and detail view. AIP Q&A widget integrated. Application published to the KM consumer user group.

**EQUIPMENT:** Foundry Workshop, AIP Logic workflow (Q&A, from Section 5-4), all knowledge Object Types in Ontology.

**PROCEDURE:**

1. In Workshop, create a new application: `[UnitName] Knowledge Browser`.
2. Create the main page with a three-column layout: filter panel (left, 20%), results list (center, 50%), detail view (right, 30%).
3. Build the filter panel using Object Set Filter widgets. Connect each filter to the Lesson Object Set. Configure multi-select filters for domain, level, type, and MOS. Configure date range picker for event date. Configure classification filter using conditional visibility rules based on user group.
4. Build the results list using an Object List widget. Display: title, domain tag, tactical level, date, lesson type icon. Connect to the filtered Object Set. Set pagination to 20 results.
5. Build the detail view using a Selected Object Detail widget. Add property display fields for all visible lesson properties. Add a Related Objects section using Link Explorer widgets for AAR, Exercise, TTP, and SOP links.
6. Add Action buttons to the detail view: "Suggest SOP Update" (triggers SOP review Action), "Flag for Review" (triggers KM queue routing Action), "Save to Unit Collection" (triggers bookmark Action).
7. Add the AIP Q&A widget at the top of the center panel. Connect to the Q&A AIP Logic workflow (Section 5-4). Configure the feedback widget below the answer panel.
8. Set page-level access: KM-Consumer group (read only); KM-Contributor group (full); KM-Admin group (full + admin panel).
9. Test with a set of published lessons. Verify filters narrow results correctly. Verify detail view displays all fields. Verify links navigate to linked objects. Verify AIP Q&A returns cited answers.
10. Publish to the KM-Consumer user group.

---

# CHAPTER 7 — DOCTRINE AND SOP MANAGEMENT

## 7-1. Purpose

**BLUF:** Doctrine and SOPs are only effective when current, version-controlled, and accessible. A unit operating from a superseded SOP is a training and safety risk. MSS provides the infrastructure for doctrine currency tracking, change management workflows, and unit-level SOP versioning. The KM is responsible for ensuring this infrastructure functions continuously.

## 7-2. SOP Lifecycle Management

Every SOP has a lifecycle. MSS tracks the lifecycle through the `SOP` Object Type and associated workflow Actions.

**SOP lifecycle stages:**

| Status | Meaning | Next Step |
|---|---|---|
| Draft | Being written; not yet effective | Review and approval workflow |
| In Review | Submitted for approval; under SME/leadership review | Approval or return for revision |
| Current | Approved and in effect; this is the authoritative version | Periodic review cycle |
| Under Revision | A new version is in development; current version remains effective | New version completes Review cycle |
| Superseded | Replaced by a newer version; retained for historical reference | Archive |
| Archived | No longer current; retained for compliance/reference | No further action |

**CAUTION**

Archiving a SOP does not delete it. Archived SOPs remain in the repository as historical records. Ensure the search interface defaults to filtering for Current status only — users who accidentally access an Archived SOP and follow its procedures may be operating against outdated guidance. Label Archived and Superseded SOPs with a prominent banner in the detail view.

DRAFT

## 7-3. SOP Version Control

Each SOP revision creates a new `SOP` object linked to the prior version via the `supersedes` Link Type. The link chain provides a complete version history without overwriting prior versions.

**SOP version control workflow:**

**Step 1 — Initiation.** A revision is initiated by: a scheduled review date, a lessons-learned flag (Section 4-6, Rule 4), a doctrine change notification, or leadership direction. Change the status of the current SOP to `Under Revision`. Create a new Draft SOP object linked to the current version.

**Step 2 — Drafting.** The SOP author edits the content in the Draft object. The Current version remains in effect and accessible during drafting. Do not change the Current version during the revision cycle — changes are drafted in the new object.

**Step 3 — Review.** Draft SOP enters the review workflow. Reviewer(s) assigned as linked `ExpertiseProfile` objects. Workshop Action notifies reviewers. Reviewers use the Workshop comment widget to document their feedback directly on the Draft object.

**Step 4 — Approval.** Appropriate commander or authority approves via Workshop Action (digitally recorded, with timestamp and user ID). Status changes to Approved. Effective date is set.

**Step 5 — Publication.** On the effective date, a scheduled pipeline changes the status of the prior Current SOP to Superseded and the new Approved SOP to Current. Both versions remain in the repository; search defaults surface the Current version.

**Step 6 — Archive.** Superseded SOPs are eligible for archiving after 2 years. Archival is a KM action, not automated — review for any pending references or cited litigation before archiving.

---

## 7-4. Doctrine Currency Tracking

Army doctrine publications change. USAREUR-AF units must identify when the doctrine they train against has been updated and ensure their SOPs and TTPs reflect current doctrine.

**Doctrine currency tracking system design:**

The `DoctrinePub` Object Type tracks Army, joint, and NATO doctrine publications relevant to the unit. Link `SOP` objects to the doctrine publications they cite via the `referencedIn` Link Type.

When a doctrine publication is updated (captured via a Change Notice object or manual KM update), an AIP Logic workflow can: 1. Identify all SOPs linked to the updated doctrine publication 2. Create a `ReviewNotification` for each linked SOP 3. Route notifications to SOP owners 4. Add the update to the doctrine change log

**Quiver dashboard panel: Doctrine Currency Status** - All `DoctrinePub` objects in the unit's library - Traffic light status: Green (current), Yellow (change notice issued, review pending), Red (SOP linked to this doctrine not reviewed within 90 days of change notice) - Sort by Red status first

**Table 7-1. Standard Doctrine Publications in USAREUR-AF KM System**

| Publication | Relevance | Review Trigger |
|---|---|---|
| FM 7-0 (Train to Win) | AAR process, training management | Major revision notification |
| FM 3-0 (Operations) | Operational doctrine basis for TTPs | Major revision notification |
| FM 6-0 (Commander and Staff Organization) | C2 SOPs, battle rhythm procedures | Major revision notification |
| ADP 6-22 (Army Leadership) | Leader development SOPs | Major revision notification |
| ATP 2-91.7 (Intelligence Support) | G2/S2 TTPs, intelligence SOPs | Major revision notification |
| STANAG 2014 (Orders format) | Allied / coalition SOPs | NATO change notification |
| STANAG 4778 (Lessons Learned) | Lessons learned exchange format | NATO change notification |

## 7-5. Unit SOP Repository Structure

USAREUR-AF units maintain SOPs across multiple functional areas. The MSS SOP repository should mirror the functional area structure of the unit.

**Standard USAREUR-AF BCT SOP functional area taxonomy:**

| Functional Area | Scope |
|---|---|
| Operations (S3) | Battle rhythm, operations order formats, decision cycle SOPs |
| Intelligence (S2) | Collection management, reporting formats, IPB SOPs |
| Sustainment (S4) | Supply request procedures, maintenance SOPs, fuel/water SOPs |
| Communications (S6) | PACE plans, network management SOPs, COMSEC procedures |
| Medical (S1/Medical) | CASEVAC/MEDEVAC procedures, medical reporting SOPs |
| Legal/JAG | ROE implementation, detention handling procedures |
| KM | Knowledge capture procedures, AAR submission SOPs, lessons submission procedures |

| Functional Area | Scope |
|---|---|
| Safety | Risk management procedures, range SOPs, safety briefing standards |

---

### 7-6. Task: Configure an SOP Review Notification Workflow

**CONDITIONS:** The unit SOP library has 45 Current SOPs. Quarterly review cycle is required for all SOPs. The `SOP` Object Type is deployed. S3 and functional area NCOICs are designated SOP owners.

**STANDARDS:** Automated notifications delivered to SOP owners 30 days before review date. Overdue notifications escalated to KMO at 7 days past due. KMO Quiver dashboard showing review compliance status.

**EQUIPMENT:** Foundry Actions (SL 3 access), Workshop notification widgets, Quiver.

**PROCEDURE:**

1. Verify all 45 SOP objects have `reviewDate` populated and `owner` linked to the correct `ExpertiseProfile`.
2. In Pipeline Builder, create a scheduled dataset: `sops_due_next_30_days` — query all Current SOPs where `reviewDate` is between today and today + 30 days.
3. Create an Action: `Send-SOP-Review-Notification`. Trigger: when a SOP appears in the `sops_due_next_30_days` dataset. Action: create a notification object linked to the SOP and the owner ExpertiseProfile. Send a Workshop inbox notification to the owner.
4. Create a second Action: `Escalate-Overdue-SOP-Review`. Trigger: when a SOP has `reviewDate` past today by 7 days and `reviewStatus != Reviewed`. Action: notify the KMO. Create an overdue flag on the SOP object.
5. In Quiver, create the review compliance panel:
6. Table: all Current SOPs with columns for title, functional area, review date, owner, review status
7. Conditional formatting: Green (reviewed on time), Yellow (due in next 30 days), Red (overdue)
8. Test the notification Action by temporarily setting one SOP's review date to tomorrow. Verify notification fires and lands in the owner's Workshop inbox.
9. Deploy. Validate within the first review cycle that at least one real notification fires correctly.

---

# CHAPTER 8 — PERSONNEL EXPERTISE MAPPING

## 8-1. Purpose

**BLUF:** The hardest question in a high-PCS-tempo organization is: "Who in the formation knows how to do this?" Personnel expertise mapping answers that question systematically. MSS can maintain a live, queryable expertise directory that survives personnel turnover and makes the formation's human knowledge visible.

---

## 8-2. ExpertiseProfile Object Type

The `ExpertiseProfile` Object Type is the foundation of personnel knowledge mapping. It is also the highest-risk Object Type in the knowledge architecture from a Privacy Act and personnel data perspective. Design, access control, and retention must all be reviewed with legal counsel before deployment.

**Privacy Act requirements for ExpertiseProfile:**

- Collection authority: document the legal authority (Army regulation or statute) for collecting each data element
- Purpose: clearly state the purpose of collection (KM system, not HR system — do not duplicate or replace eMILPO/iPERMS data)
- Access restriction: strict access controls; only KM-Admin and KM-Sensitive groups
- No coalition access: under no circumstances should ExpertiseProfile data be accessible to coalition or partner nation systems
- Retention schedule: coordinate with RMO for approved schedule; typical is 2 years post-reassignment
- Individual access: Soldiers should be able to view and correct their own profile

**ExpertiseProfile properties (beyond Table 2-1):**

| Property | Type | Notes |
|---|---|---|
| `primaryMOS` | Text | Current duty MOS |
| `additionalSkills` | Multi-select | Skills beyond MOS (e.g., Language, Medical, Cybersecurity, Mountaineering) |
| `languages` | Multi-select | Language proficiency codes (DLPT scores if appropriate) |
| `certifications` | Multi-select | Relevant certifications (HAZMAT, Engineer, etc.) |

| Property | Type | Notes |
|---|---|---|
| `deploymentHistory` | Link to Exercise/Operation objects | Combat/operational deployments — links to Exercise objects |
| `smeDomains` | Multi-select | Operational domains where individual is recognized as SME |
| `availableAsAdvisor` | Boolean | Has the individual agreed to be contacted as an SME? |
| `advisorContactMethod` | Text | Preferred contact for SME queries (email, DSN) |

**WARNING**

Never auto-populate ExpertiseProfile from HR systems without individual verification. Personnel data imported from external systems may be stale, incorrect, or misclassified. Each profile should be reviewed and confirmed by the individual before being marked as active. An inaccurate expertise directory is worse than no directory — it routes queries to the wrong people and erodes user trust in the system.

## 8-3. Skills Taxonomy Design

A skills taxonomy defines the controlled vocabulary for `additionalSkills`, `certifications`, and `smeDomains`. Without a controlled taxonomy, the skills fields become free text — unsearchable and inconsistent.

**Building the skills taxonomy:**

1. Conduct a skills inventory: survey key leaders and functional area NCOICs to identify the skills and expertise categories most frequently needed in the unit. Focus on cross-domain skills that are not captured by MOS code.
2. Organize into categories:
3. Technical skills (cybersecurity, GIS, SIGINT systems, specific equipment)
4. Language skills (language + proficiency level)
5. Operational skills (SERE, HALO/HAHO, combat diving, CBRN response)
6. Professional skills (legal, medical, contracting, civil affairs)
7. Education (advanced degrees, specialized courses — NWC, CAS3, ILE, CGSC)
8. Define controlled values for each category. Limit to skills that are operationally relevant and verifiable. Avoid subjective assessments ("good communicator") — stick to discrete, verifiable skills.

9. Publish the taxonomy as a reference dataset in Foundry. AIP tag suggestion prompts reference this dataset for available skill values. Quarterly review cycle for the taxonomy.

---

## 8-4. SME Identification Workflow

The primary operational use of ExpertiseProfile is answering the question: "I need to find a Soldier with [specific skill or expertise] — who in the formation can help?"

**SME identification via Workshop:**

Build a dedicated "Find an Expert" Workshop page: - Filter by `smeDomains` (multi-select from taxonomy) - Filter by `additionalSkills` (multi-select) - Filter by `languages` (multi-select) - Filter by `availableAsAdvisor = true` (default on) - Results list: rank, name (display only to authorized users), MOS, matched skills, contact method - Result includes link to their deployment/exercise history (via `participatedIn` links to Exercise objects)

**SME identification via AIP Q&A:**

Configure the knowledge Q&A workflow to recognize SME queries: "Who knows about [topic]?" should trigger an ExpertiseProfile search rather than a lesson search. AIP workflow classification step: if question matches patterns indicating an SME query, route to ExpertiseProfile search rather than Lesson search.

---

## 8-5. Expertise Gap Analysis

Expertise gap analysis answers: "What skills does the formation lack? Where are we vulnerable to personnel transitions?"

**Performing an expertise gap analysis:**

**Step 1 — Define required expertise.** For each operational domain in the unit's mission set, define the minimum required expertise: how many personnel with each skill are needed for the unit to execute its core tasks?

**Step 2 — Query current coverage.** Use Contour to analyze ExpertiseProfile objects: - Count of personnel per skill - Distribution by rank (key skills concentrated in one rank band are succession risks) - Distribution by unit (skills concentrated in one sub-unit create cross-unit dependency)

**Step 3 — Identify gaps.** Where current coverage is below the minimum required, create a `KnowledgeGap` object: - `description`: specific skill or expertise area lacking - `priority`: High / Medium / Low (based on operational impact of the gap) - `mitigationPlan`: training, assignment actions, or external support required - `owner`: who is responsible for closing the gap

**Step 4 — PCS risk assessment.** For each high-priority skill, identify personnel with that skill who are within 12 months of PCS eligibility. These are the high-risk succession dependencies that Chapter 9 addresses.

---

### 8-6. Task: Build the SME Directory Application

**CONDITIONS:** ExpertiseProfile Object Type is deployed with Privacy Act review complete. Initial profiles have been created for 75% of the unit's senior NCOs and officers (SL 2 qualified) and functional SMEs. Legal review complete.

**STANDARDS:** A functional SME directory application accessible to KM-Consumer users that allows skill-based SME search and returns contact information for available advisors. Expertise gap dashboard accessible to KM-Admin.

**EQUIPMENT:** Foundry Workshop, ExpertiseProfile Object Type, skills taxonomy dataset.

**PROCEDURE:**

1. Create a new Workshop application: `[UnitName] Expert Directory`.
2. Page 1 — SME Search: build the filter panel per Section 8-4. Set default filter: `availableAsAdvisor = true`. Build results list: rank, masked name (first initial, last name), MOS, matched skills. On selection, show contact method and deployment history links.
3. Page 2 — Expertise Gap Dashboard (KM-Admin only): Contour embed showing skill distribution analysis. KnowledgeGap object list filtered by `priority = High`. PCS-risk table: ExpertiseProfile records where `availableAsAdvisor = true` and PCS within 12 months (requires PCS date field or manual flag).
4. Configure access: Page 1 — KM-Consumer; Page 2 — KM-Admin only. Verify that Page 2 is not accessible to Consumer-role users.
5. Add a "Request SME Contact" Action on the detail view: submits a structured request to the SME and the KMO, logs the request as a linked event on the ExpertiseProfile.
6. Publish and brief the application to S3, functional area chiefs, and the command team.

> **NOTE**
>
> Brief the application thoroughly at first deployment. Personnel who do not know it exists will not use it. The first successful use case — a planner finds an expert they would otherwise have spent days searching for — is the most powerful advertisement for the system. Identify a likely near-term use case and position the tool to enable it visibly.

---

# CHAPTER 9 — KNOWLEDGE TRANSFER AND CONTINUITY

## 9-1. Purpose

**BLUF:** USAREUR-AF has a persistently high PCS tempo. Hard-won operational knowledge departs with personnel. MSS-based knowledge transfer workflows reduce this loss by providing structured handoff processes, key-person dependency mapping, and systematic knowledge capture before personnel transition out of the organization.

---

## 9-2. Key Person Dependency Analysis

A key person dependency exists when critical knowledge, relationships, or capabilities are concentrated in one or very few individuals. When those individuals PCS or ETS, the capability degrades until a replacement is up to speed — or, if the knowledge was never captured, it is lost entirely.

**Identifying key person dependencies:**

Using the ExpertiseProfile and KnowledgeGap Object Types:

1. Run the expertise gap analysis (Section 8-5) with a filter for skills held by three or fewer personnel in the unit.
2. Cross-reference against personnel within 12 months of PCS eligibility.
3. The intersection — a critical skill held by one or two people who are near PCS — is a high-risk key person dependency.
4. Create a `KnowledgeGap` object for each identified dependency with `priority = High` and a mitigation plan that includes knowledge capture actions.

**Common key person dependencies in USAREUR-AF:**

| Role Type | Common Knowledge Gap | Mitigation |
|---|---|---|
| Senior KMO | Full knowledge architecture, AIP workflow configs, access control design | Knowledge transfer package; successor onboarding before departure |
| S6 Data NCOIC | Pipeline configurations, data source contacts, error handling procedures | SL 4K qualified successor; documented pipeline runbook |
| G2/S2 All-Source | Analytical products, source network context, AOR-specific intelligence | Classified knowledge transfer; product handoff protocol |
| Master Gunner | Equipment-specific TTPs, range procedures | TTP library in MSS; successor certification |

| Role Type | Common Knowledge Gap | Mitigation |
|---|---|---|
| Language-qualified personnel | Native language proficiency for partner nation liaison | Language SME registry; DLPT requirements for replacements |

## 9-3. Knowledge Transfer Package Design

A knowledge transfer (KT) package is the structured set of knowledge products a departing Soldier prepares before transitioning. On MSS, the KT package is a curated set of Foundry objects — SOPs, lessons, TTP entries, system documentation, and the ExpertiseProfile — organized for rapid orientation by a successor.

**KT package components:**

| Component | Object Type | Purpose |
|---|---|---|
| Role overview | `SOP` (type: Role Overview) | What this role does, key responsibilities, authorities |
| Critical SOPs | `SOP` (linked to role) | All SOPs in the role's functional area |
| Lessons contributed | `Lesson` (authored or reviewed) | Lessons this individual generated or validated |
| Active knowledge gaps | `KnowledgeGap` (owned) | Gaps this individual is tracking — successor assumes ownership |
| Key contacts | Free text or `ExpertiseProfile` links | External contacts, counterparts, liaisons not in the MSS directory |
| AIP workflow documentation | `SOP` (system runbook) | How the role's AIP workflows are configured, how to maintain them |
| Pending items | Action log | Any open Actions, pending reviews, or uncompleted workflows |

**KT package creation is a mandatory task in the departing timeline, not an optional suggestion.** KMs must build the KT package creation workflow into the unit's departure SOP.

## 9-4. PCS Handoff Workflow

The PCS handoff workflow is an MSS Workshop-based guided process that walks a departing Soldier through all required knowledge transfer steps and notifies the KMO when the package is complete.

**PCS handoff workflow stages:**

**Stage 1 — Initiation (60 days before departure):** - Departing Soldier or supervisor initiates the workflow via Workshop Action - System creates a `KnowledgeTransfer` event object linked to the departing Soldier's ExpertiseProfile - KMO notified; KT checklist object created

**Stage 2 — Knowledge audit (45 days before departure):** - Automated query: all SOPs with this Soldier as owner → list added to KT checklist - All active KnowledgeGap objects with this Soldier as owner → list added to KT checklist - All AIP workflows this Soldier administers → prompted for runbook documentation

**Stage 3 — Documentation (30 days before departure):** - For each SOP the Soldier owns: verify currency; if Under Revision, accelerate or hand off revision to designated successor - For each KnowledgeGap: update mitigation status; reassign to successor or KMO - Prepare role overview SOP if not current

**Stage 4 — Successor onboarding (14 days before departure):** - Link successor's ExpertiseProfile to the KT event - Successor reviews KT package in Workshop; marks each component as "reviewed" - Departing Soldier and successor complete at least one joint work session — logged as an event

**Stage 5 — Completion certification (departure day):** - KMO reviews KT package completeness - KMO certifies completion via Workshop Action - Departing Soldier's ExpertiseProfile status updated to "Departed" - Successor ExpertiseProfile updated with inherited skill areas

> **NOTE**
>
> Stage 4 joint work session is the most important step and the most commonly skipped. A Workshop-certified KT package is not a substitute for direct human knowledge transfer. The package tells the successor what exists; the joint session shows them how things actually work. Enforce this step as a command requirement, not a KM suggestion.

---

## 9-5. Organizational Memory Preservation

Beyond individual PCS transitions, KMs are responsible for preserving organizational memory across command changes, unit reorganizations, and exercise cycles.

**Organizational memory products:**

**Unit history objects.** Create `UnitRecord` objects documenting each exercise, deployment, and significant event in the unit's history. Link all AARs, lessons, and key personnel to the event. This is the institutional record — it should exist regardless of whether any individual who participated is still assigned.

**Command team knowledge transfer.** Command changes are high-risk knowledge transitions. Brief incoming commanders on the MSS knowledge architecture, the current state of the knowledge repositories, and the active knowledge gaps within 30 days of assuming command. Document the brief

as a `KnowledgeTransfer` event object.

**Exercise cycle knowledge products.** After each Combined Resolve or Defender Europe exercise, produce three mandatory knowledge products: 1. **Exercise AAR Summary** — aggregate summary of all AARs from the exercise, organized by domain and tactical level 2. **Lessons Extracted** — all lessons from the exercise in Published status, tagged and routed 3. **Pre-exercise Knowledge Package** — bundled from Lesson objects, prepared for the next unit rotating through the same exercise

These three products ensure each exercise cycle builds on the previous one rather than restarting from zero.

---

## 9-6. Metrics: Knowledge Health Assessment

The KM conducts a quarterly knowledge health assessment and briefs the command team on results. The assessment measures the health of the knowledge repository and identifies required actions.

**Table 9-1. Knowledge Health Metrics**

| Metric | Measurement Method | Target | Red Threshold |
|---|---|---|---|
| Lesson capture rate | AARs with at least one lesson extracted / total AARs submitted | > 70% | < 40% |
| Lesson publication rate | Lessons in Published status / total lessons submitted | > 80% within 30 days | < 50% |
| SOP currency rate | SOPs in Current status with review date not past | 100% | < 85% |
| ExpertiseProfile coverage | Active personnel with profiles / total unit personnel | > 75% | < 50% |
| KT package completion | PCS departures with certified KT packages / total PCS departures | > 90% | < 70% |
| AIP Q&A positive feedback | Positive / total feedback responses | > 65% | < 45% |
| Knowledge gap closure rate | KnowledgeGap objects marked resolved in last 90 days / total High-priority gaps | > 30% per quarter | 0% (no closures) |

Present the health assessment as a Quiver dashboard. Brief the command team quarterly. Red thresholds trigger an improvement plan with specific actions and timelines.

---

# APPENDIX A — KNOWLEDGE ARCHITECTURE DESIGN CHECKLIST

Use this checklist before submitting any knowledge system design for C2DAO review. All items must be checked before production deployment.

## Domain and Ownership

- ☐ All knowledge domains are identified and documented
- ☐ Each domain has a named owner with acknowledged responsibility
- ☐ Each domain has a defined review cycle
- ☐ Consumer population is identified for each domain
- ☐ Producer sources are identified for each domain

## Object Types

- ☐ All Object Types derived from or aligned to USAREUR-AF standard knowledge Object Types
- ☐ All required properties defined and marked required in schema
- ☐ Controlled vocabularies defined for all enumerated fields
- ☐ `reviewDate` is a required property on every Object Type
- ☐ `classification` is a required property on every Object Type
- ☐ Privacy Act review completed for ExpertiseProfile (if used)
- ☐ Object Type design submitted for C2DAO review before deployment

## Link Types

- ☐ All Link Types defined with correct cardinality
- ☐ Link chain between Exercise → AAR → Lesson → TTP is complete
- ☐ Link chain between Lesson → SOP (`relevantTo`) is configured
- ☐ `supersedes` link for SOP versioning is configured

## Access Control

- ☐ Access control design documented
- ☐ Named access groups configured in Foundry
- ☐ No coalition/MPE access to any knowledge object without C2DAO approval
- ☐ ExpertiseProfile access restricted to KM-Sensitive and KM-Admin groups

DRAFT

- ☐ Classification filter configured in all search interfaces

### AIP Logic

- ☐ All AIP prompts include role definition, scope constraint, uncertainty instruction, and output format specification
- ☐ AIP prompts are version-controlled
- ☐ AIP outputs route to human review before any automated publishing
- ☐ AIP Q&A grounded to knowledge objects only (no general AI knowledge responses)
- ☐ AIP feedback dataset configured

### Governance

- ☐ Records management retention schedule confirmed with RMO
- ☐ Privacy Act SORN reviewed for any PII-containing Object Types
- ☐ C2DAO coordination memo complete
- ☐ Knowledge system architecture document signed by unit KM authority

## APPENDIX B — AAR DATA MODEL REFERENCE

### B-1. AAR Object Type — Full Property Schema

| Property | Type | Required | Controlled Values | Notes |
|---|---|---|---|---|
| `aarID` | String | Yes | Auto-generated | Format: AAR-[UNIT]-[YYYYMMDD]-[SEQ] |
| `eventName` | String | Yes | Free text | Link to Exercise object if available |
| `eventType` | Enum | Yes | Tactical Movement, Breach, Air Assault, Logistics, C2 Exercise, Comms Exercise, Staff Ride, Wargame, Other | |
| `dateOfEvent` | Date | Yes | | Cannot be future date; cannot be older than 90 days from submission |

| Property | Type | Required | Controlled Values | Notes |
|---|---|---|---|---|
| `unit` | String | Yes | Unit registry | Must match a valid unit code |
| `classification` | Enum | Yes | UNCLASSIFIED, CUI, SECRET | |
| `sustainActions` | Text Array | Yes | | Min 1 entry, min 10 characters |
| `improveActions` | Text Array | Yes | | Min 1 entry, min 10 characters |
| `correctiveActions` | Object Array | Yes | | Min 1 entry with owner and target date per improve action |
| `participants` | Integer | No | | Count of personnel involved |
| `observerController` | String | No | Free text or ExpertiseProfile link | |
| `lessonExtractionFlag` | Boolean | Yes | True/False | Default: False |
| `lessonExtractionNotes` | Text | Conditional | | Required if lessonExtractionFlag = True |
| `status` | Enum | Yes | Draft, Submitted, Reviewed, Archived | Default: Draft |
| `submittedBy` | ExpertiseProfile Link | Auto | | Auto-populated from user session |
| `submittedDate` | DateTime | Auto | | Auto-populated on submission |
| `reviewedBy` | ExpertiseProfile Link | No | | KM reviewer |
| `reviewedDate` | DateTime | No | | Auto-populated on review action |

DRAFT

## B-2. Link Types for AAR

| Link Type | Source | Target | Created By |
|---|---|---|---|
| `generatedByExercise` | AAR | Exercise | Submission pipeline (auto-match by event name/date) |
| `submittedBy` | AAR | ExpertiseProfile | Submission pipeline (auto from user session) |
| `extractedLesson` | AAR | Lesson | Lessons pipeline (when lessonExtractionFlag = True) |

## B-3. AAR Validation Rules Summary

| Rule | Field | Condition | Action |
|---|---|---|---|
| V-001 | dateOfEvent | Future date | Block submission |
| V-002 | dateOfEvent | Older than 90 days | Warn; require KMO override |
| V-003 | unit | Not in unit registry | Block submission |
| V-004 | sustainActions | Length < 10 chars | Block submission |
| V-005 | improveActions | Length < 10 chars | Block submission |
| V-006 | correctiveActions | Missing owner for any improve action | Block submission |
| V-007 | classification | Not selected | Block submission |
| V-008 | lessonExtractionNotes | Missing when flag = True | Block submission |
| V-009 | All | Classification = SECRET | Route to classified workflow; block normal submission |

DRAFT

# APPENDIX C — AIP KNOWLEDGE WORKFLOW PATTERNS

## C-1. Standard Prompt Patterns for Knowledge Management

The following prompt templates are approved starting points for common KM AIP Logic workflows. Each template must be customized with unit-specific context and tested in staging before production deployment.

### Pattern C-1.1 — Document Summarization and Lesson Extraction

```
# Version: 1.0 | Workflow: Document-Summarizer | Updated: [DATE]
You are a knowledge management assistant for a U.S. Army unit in USAREUR-AF.
Your task is to analyze the following document and extract discrete lessons learned.

For each lesson you identify:
1. Write a concise title (10 words or fewer)
2. State the observation: what happened, factually and specifically
3. Write the discussion: why it happened, what factors contributed
4. Write the recommendation: what should be done differently, or confirmed as a
sustain practice

Rules:
- Extract only what is explicitly stated or directly implied in the document
- If a claim is unclear or ambiguous, flag the lesson as "Requires KM Review" in the
title
- Do not invent details not present in the document
- Military-specific terms: TTP = Tactic, Technique, Procedure; AAR = After-Action
Review; OPORD = Operations Order; FRAGO = Fragmentary Order; MEDEVAC = Medical
Evacuation
- Format output as a JSON array with fields: title, observation, discussion,
recommendation, requiresReview (boolean)

Document text:
{document_text}
```

### Pattern C-1.2 — Content Tagging

```
# Version: 1.0 | Workflow: Tag-Suggester | Updated: [DATE]
You are a tagging assistant for an Army knowledge management system.
Suggest tags for the following lesson from the available values only.
Do not create new tag values. If uncertain, leave the field as null.

Available operationalDomain values: {domain_list}
Available relevantMOS values: {mos_list}
Available tacticalLevel values: {level_list}
```

```
Available operationalEnvironment values: {environment_list}
Available lessonType values: {lesson_type_list}

Lesson title: {title}
Lesson observation: {observation}
Lesson recommendation: {recommendation}

Return output as JSON with fields: operationalDomain (string), relevantMOS (array),
tacticalLevel (string), operationalEnvironment (array), lessonType (string).
```

---

### Pattern C-1.3 — Knowledge Q&A with Citation

```
# Version: 1.0 | Workflow: Knowledge-QA | Updated: [DATE]
You are a knowledge management assistant for a U.S. Army unit in USAREUR-AF.
Answer the user's question using ONLY the knowledge objects provided below.

Rules:
- If the answer is not contained in the provided objects, respond exactly: "I do not
have a lesson or document that addresses this question. Consider submitting a
knowledge gap report to your unit KMO."
- Do not use your general training knowledge to answer. If provided objects do not
contain the answer, say so.
- Cite the object ID (e.g., "Lesson LSN-2024-0047") for every factual claim you make.
- Military context: this is U.S. Army USAREUR-AF. Acronyms follow Army standards.

User question: {user_question}

Knowledge objects:
{retrieved_objects}
```

---

### Pattern C-1.4 — Duplicate Detection

```
# Version: 1.0 | Workflow: Duplicate-Detector | Updated: [DATE]
You are a knowledge management assistant. Assess whether the following two lessons are
duplicates, near-duplicates, or distinct.

Definitions:
- Duplicate: substantively the same observation, drawn from the same or similar event
- Near-duplicate: similar observation, possibly different event or slightly different
context — flag for KM review
- Distinct: different observations that should remain separate records

Lesson A:
Title: {lesson_a_title}
Observation: {lesson_a_observation}

Lesson B:
Title: {lesson_b_title}
Observation: {lesson_b_observation}
```

```
Return JSON with fields: assessment (Duplicate / Near-Duplicate / Distinct),
confidence (High / Medium / Low), reasoning (one sentence explanation).
```

### C-2. AIP Workflow Governance Requirements

| Requirement | Standard |
|---|---|
| All production AIP prompts must be version-controlled | Include version number and date in comment at top of prompt |
| Staging test required before production deployment | Minimum 10 test cases with human-validated expected outputs |
| Prompt changes to production workflows require KMO approval | Document change rationale and test results |
| AIP outputs routing to published knowledge objects must pass human review | No AIP output auto-publishes without KM approval |
| Feedback datasets must be reviewed quarterly | KMO reviews negative feedback patterns and adjusts prompts or knowledge gaps accordingly |

## APPENDIX D — PROFESSIONAL READING LIST

Curated articles from Army professional journals and military publications. These supplement doctrinal references with contemporary operational perspectives.

| Source | Title | Date | Relevance |
|---|---|---|---|
| NCO Journal | "Knowledge Management and The Old Guard" | Aug 2025 | KM in practice |
| Green Notebook | "How To Be a Data Literate Leader" | Mar 2024 | Data literacy for KM leaders |
| Green Notebook | "Harnessing the Power of Knowledge Management" | Apr 2024 | KM fundamentals |
| Small Wars Journal | "Elevating Information as a Core WfF for MDO" | Apr 2025 | Information as warfighting function |

# GLOSSARY

**AAR (After-Action Review).** A structured review of a training event or operation that answers four questions: what was supposed to happen, what actually happened, why was there a difference, and what will we sustain or improve. Defined in FM 7-0.

**Action (Foundry).** A configured workflow in the Foundry Ontology that executes a defined operation — creating, modifying, or linking objects — triggered by a user or an automated condition. Used in KM workflows for routing, status changes, and notification.

**AIP Logic.** An AIP (AI Platform) component in MSS that allows KMs to configure natural language AI workflows connected to Foundry Ontology data. Enables summarization, extraction, tagging, and Q&A without code.

**Archive.** The lifecycle state of a knowledge object that is no longer current but is retained for historical reference. Archived objects are not deleted; they are suppressed from default search results.

**C2DAO (Command and Control Data Architecture Office).** USAREUR-AF authority for theater-level data architecture standards on MSS. All knowledge ontology designs and enterprise deployments require C2DAO coordination.

**CALL (Center for Army Lessons Learned).** U.S. Army organization at Fort Leavenworth, Kansas, responsible for collecting, analyzing, and distributing lessons learned from Army operations and training. A primary external source for USAREUR-AF KM ingestion pipelines.

**Classification.** The security designation of a knowledge object — UNCLASSIFIED, CUI, SECRET, or releasability-qualified. Every knowledge object must carry a classification property; access control enforces it at the dataset and application level.

**Combined Resolve.** A semi-annual USAREUR-AF-hosted multinational exercise series that generates a large volume of AAR data, lessons learned, and TTPs. A primary driver of knowledge capture operations for USAREUR-AF KMs.

**Core Object Views.** A platform-managed standard view (GA February 2026) for browsing and exploring Ontology objects. Provides consistent object detail display, linked object navigation, and inline action execution across all Foundry surfaces. See Section 6-2a.

**Contour.** Foundry's multi-dimensional analysis tool. Used by KMs for lesson pattern analysis, coverage gap identification, and cross-exercise trend analysis.

**Controlled Vocabulary.** A defined set of allowed values for a tag or classification field. Controlled vocabularies enforce consistent tagging, which is the foundation of reliable knowledge search.

**Defender Europe.** A U.S. Army Europe-led exercise series involving large-scale logistics and force projection across the European AOR. Generates exercise-specific lessons in sustainment, transportation, and C2.

**Deduplication.** The process of identifying and managing duplicate or near-duplicate knowledge objects. Duplicates are suppressed, not deleted; near-duplicates are flagged for KM review.

**Document Intelligence (AIP).** A managed Foundry service (GA Q1 2026) that extracts text from uploaded documents, chunks it into semantically coherent passages, embeds passages using a platform-managed model, and provides similarity-based retrieval for AIP Logic and Agent Studio workflows. Enables KMs to ingest document collections without custom code. See Section 5-6a.

**Domain Owner.** The individual accountable for the quality, currency, and accuracy of knowledge in a defined knowledge domain. Domain ownership is a governance requirement under UDRA v1.1.

**ExpertiseProfile.** A Foundry Object Type that represents an individual Soldier's skills, qualifications, language proficiencies, and subject matter expertise. Subject to Privacy Act requirements.

**Foundry.** The Palantir Foundry platform on which MSS is built. Provides the data, ontology, application, and AI infrastructure for USAREUR-AF knowledge management operations.

**Grounding (AIP).** The configuration of an AIP Logic workflow to answer from retrieved knowledge objects only, rather than from the AI's general training knowledge. Grounding is mandatory for all KM Q&A workflows.

**Hallucination (AIP).** The generation by an AI system of factually incorrect or fabricated content stated as fact. A primary risk in KM workflows — mitigated by grounding, citation enforcement, and mandatory human review before publishing.

**Institutional Knowledge.** The collective expertise, experience, lessons learned, and procedural understanding held by an organization's members. Institutional knowledge is at risk when personnel transition; KM systems exist to externalize and preserve it.

**JLLIS (Joint Lessons Learned Information System).** The DoD-wide system for lessons learned collection and sharing. USAREUR-AF KMs coordinate with J7 for access to JLLIS data as an ingestion source.

**Key Person Dependency.** A condition where critical knowledge or capability is concentrated in one or very few individuals, creating organizational vulnerability when those individuals transition.

**KM (Knowledge Management).** The systematic process of capturing, organizing, sharing, and applying organizational knowledge to improve performance. Army KM is governed by doctrine and guided by the KMO.

**KMO (Knowledge Management Officer).** The officer or senior NCO assigned as the primary knowledge manager for a unit. At division and corps level, typically a 37F MOS. At brigade and below, may be a collateral duty for S3 or S6 personnel.

**Knowledge Gap.** A documented area where required knowledge does not exist in the repository, cannot be found, or is held by too few personnel. Represented as a `KnowledgeGap` Object Type in MSS.

**Knowledge Transfer Package.** The structured set of SOPs, lessons, TTP entries, and documentation a departing Soldier prepares before PCS or ETS. Required before departure in units with MSS-based KM systems.

**Lesson (Knowledge Object).** A structured knowledge object representing a single extracted observation, discussion, and recommendation from operational experience. The primary unit of knowledge in the lessons learned system.

**Lessons Learned Pipeline.** The automated workflow that ingests, validates, deduplicates, tags, and routes lessons from multiple sources into the knowledge repository.

**Link Type.** A defined relationship between two Object Types in the Foundry Ontology. Link Types make knowledge navigable — the Exercise-to-AAR-to-Lesson-to-TTP link chain is the core navigation structure of the KM knowledge graph.

**MSS (Maven Smart System).** The USAREUR-AF enterprise AI/data platform, built on Palantir Foundry. The deployment environment for all SL 4K knowledge management systems.

**Object Type.** A defined entity class in the Foundry Ontology. Each Object Type has a schema (defined properties) and participates in Link Types. `Lesson`, `AAR`, `SOP`, and `ExpertiseProfile` are Object Types.

**Ontology.** The semantic layer of Foundry that defines Object Types, Link Types, and their relationships. The knowledge architecture lives in the Ontology.

**PCS (Permanent Change of Station).** The reassignment of a Soldier to a new duty station. A primary driver of institutional knowledge loss in Army organizations. KM systems are designed in part to mitigate PCS-driven knowledge gaps.

**Pipeline Builder.** Foundry's visual workflow tool for data transformation and routing. Used in KM systems to automate AAR submission processing, lesson routing, and notification workflows.

**Privacy Act.** The Privacy Act of 1974 (5 U.S.C. § 552a). Governs the collection, maintenance, and disclosure of records about individuals by federal agencies. Applies to the ExpertiseProfile Object Type.

**Prompt (AIP).** The instruction text that directs an AIP Logic workflow's AI behavior. Prompt quality directly determines output quality. KMs are responsible for prompt design, versioning, and maintenance.

**Quiver.** Foundry's dashboard and visualization tool. Used by KMs for repository coverage monitoring, SOP review compliance tracking, and knowledge health metrics dashboards.

**Records Management Officer (RMO).** The unit officer responsible for records management compliance under AR 25-400-2. KMs must coordinate with the RMO before deploying any persistent knowledge repository.

**Review Date.** A required property on all knowledge objects that specifies when the object must be reviewed for currency. Objects past their review date are flagged as potentially stale.

**SOP (Standing Operating Procedure).** A unit-level document that defines standard procedures for recurring tasks. SOPs are living documents requiring currency tracking, version control, and periodic review.

**SME (Subject Matter Expert).** A person with recognized expertise in a specific operational or technical domain. SME identification is a primary use case for the ExpertiseProfile system.

**STANAG 4778.** NATO Standardization Agreement defining the format for lessons learned exchange between NATO member nations. Used by USAREUR-AF KMs for NATO LLDB integration and Civil Affairs partner nation knowledge sharing.

**TTP (Tactic, Technique, Procedure).** Operationally derived guidance more specific than doctrine — the how of doing a task at the lowest level. TTPs are derived from lessons learned and validated by SMEs before becoming authoritative.

**UDRA (Unified Data Reference Architecture).** UDRA v1.1 (February 2025). The Army's unified data reference architecture governing federated data governance, domain ownership, and data product standards. All MSS knowledge products must align with UDRA.

**Workflow (AIP/Foundry).** A configured sequence of automated steps triggered by an event or schedule. KM workflows handle notification delivery, status transitions, routing, and AIP processing.

**Workshop.** Foundry's no-code application builder. The primary tool for building AAR capture forms, knowledge browsers, SME directories, and KM dashboard interfaces.

---

*SL 4K — Knowledge Manager Technical Manual Headquarters, United States Army Europe and Africa, Wiesbaden, Germany 2026 Distribution authorized to U.S. Government agencies and their contractors only.*

**DoD and Army Strategic References:**

- **UDRA v1.1 (February 2025)** — Unified Data Reference Architecture; domain ownership and federated governance for knowledge products
- **Army CIO Data Stewardship Memo (April 2024)** — Chain of responsibility for data governance and stewardship

**Supplementary Reading — MCCoE and Army Knowledge Management:**

- **AKMP Data Immersion Course for Knowledge Managers (MCCoE/C2ID, February 2025)** — MCCoE's Command and Control Integration Directorate runs a 32-hour online course for operational knowledge managers at division level and above. Curriculum covers data governance, Army Data Catalog, data workforce roles, data-centric question triage, generative AI capabilities, and Vantage. Prerequisites: 2-hour Excel course and 2-hour Contour course. A September 2024 pilot demonstrated substantial post-course improvement across data governance, analysis, platform use, and product development. SL 4K graduates should be aware of this complementary MCCoE program — it validates the same competencies from the institutional training perspective.

- **Knowledge Management Qualification Course (MCCoE/TRADOC)** — The Army's institutional KM course, administered by MCCoE. Maven instruction is being integrated into the KMQC curriculum as part of the CAC Maven C2 integration initiative (February 2026). SL 4K graduates are directly aligned to this career development pathway.
- **CALL 25-10, Commander and Staff Guide to Data Literacy (April 2025)** — CALL handbook covering data literacy for commanders and staff. Useful reference for KMs building products consumed by senior leaders.

DRAFT