CONCEPTS GUIDE

# SL 4J

USAREUR-AF

# CONCEPTS GUIDE — SL 4J COMPANION — DATA PROGRAM MANAGER · MAVEN SMART SYSTEM (MSS)

*Specialist Course Manual*

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



**26 MARCH 2026**

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

# CONCEPTS GUIDE — SL 4J COMPANION — DATA PROGRAM MANAGER · MAVEN SMART SYSTEM (MSS)

**Forward:** This guide is for data program managers accountable for a portfolio of data products on MSS. It does not teach Foundry mechanics. It develops the judgment to manage them. **Purpose:** Develops the mental models required to manage data pipelines, programs, and portfolio health on MSS effectively. Read before beginning SL 4J task instruction. *HQ USAREUR-AF · v1.0 · 2026 · DISTRIB: USG only*

## PREFACE

**BLUF:** This guide is for data program managers accountable for a portfolio of data products on MSS. It does not teach Foundry mechanics. It develops the judgment to manage them.

Most failures on MSS data programs are not technical — they are organizational and conceptual: a PM who accepted a delivery timeline before confirming data access; a project shipped without governance sign-off that immediately generated downstream data quality complaints; a dashboard in production for eight months with no certain owner. These failures share a common root: the PM lacked a coherent mental model of what they were managing.

Read this guide linearly before beginning SL 4J task instruction. Return to individual sections as needed when specific challenges arise.

## SECTION 1 — THE DATA PM'S ROLE ON MSS

### 1-1. What a Data PM Owns

A data PM does not own code, pipelines, Object Types, or Foundry Workshop applications in the technical sense. The PM owns **accountability for outcomes**: a portfolio of data products that are working, governed, and meeting operational requirements.

| PM Primary Output | Description |
|---|---|
| Delivery milestones | Scoped, resourced, and tracked to completion |

| PM Primary Output | Description |
|---|---|
| Portfolio health visibility | Aggregate view of what is working, what is degraded, what is at risk |
| Stakeholder alignment | Operational users and technical producers working toward the same definition of done |
| Governance compliance | Data products built and promoted through proper C2DAO channels |
| Debt awareness | Identification and prioritization of technical and governance debt accumulating in the portfolio |

The PM manages across people, pipelines, and products — not just projects. A project has a start and end date. A data product has a lifecycle requiring ongoing ownership, maintenance decisions, and eventual retirement. A PM who thinks only in projects will consistently under-resource the sustain phase.

**NOTE — DDOF Roles and PM Oversight (TM-40J Section 1-2a):** TM-40J Section 1-2a defines how the Data and Digital Operations Framework (DDOF) assigns roles across the data lifecycle and where PM oversight responsibilities begin and end. Review before scoping any new data program.

## 1-2. What the PM Does Not Own

- **Technical architecture decisions.** The PM ensures design was reviewed by the right people, meets governance requirements, and has a realistic timeline — not whether the design is technically optimal.
- **Data quality.** Data quality is owned by data stewards and enforced through governance. The PM creates conditions for quality (adequate review time, proper sign-off, testing windows).
- **Access decisions.** MSS access control belongs to the USAREUR-AF C2DAO. The PM coordinates access requests and ensures they are processed on time — does not grant access unilaterally.

## 1-3. The PM and C2DAO Governance

The C2DAO (Command and Control Data Architecture Office) sets naming conventions, owns the promotion gate criteria, maintains the master Ontology design, and adjudicates data stewardship disputes. Every data product the PM delivers must pass through C2DAO governance — every project schedule must account for C2DAO review cycles.

The productive relationship is collaborative, not adversarial. The PM who engages C2DAO early, keeps them informed of upcoming builds, and incorporates governance checkpoints into project plans will move faster, not slower.

# SECTION 2 — THINKING IN PIPELINES AND DEPENDENCIES

## 2-1. The Dependency Chain

No data product on MSS is a standalone artifact. Every product sits at the end of a dependency chain. Understanding that chain — and being able to trace it — is a core PM survival skill.

**Standard dependency chain on MSS:**

```
Source System → Ingestion Pipeline → Raw Dataset → Transformed Dataset
→ Ontology Object Type → Workshop Application → Operational User
```

A readiness reporting dashboard visible to a V Corps staff officer has a source system, ingestion pipeline, raw dataset, transformed dataset, Ontology Object Type, and Workshop application. If the source system stops sending data, the dashboard goes stale. If the transform breaks, the dashboard may display corrupt data. If the Ontology Object Type is modified without versioning, every dependent application may fail silently.

**The PM who cannot trace the dependency chain cannot triage.** When a user reports "the dashboard is wrong," the PM with dependency awareness asks: Is the source feed arriving? Is the ingestion pipeline running? Did any transforms fail in the last 24 hours? Has anyone modified the Ontology Object Type recently?

## 2-2. Dependency Mapping

Every significant data product should have a dependency map. A simple table is sufficient:

| Layer | Asset Name | Owner | Last Validated |
|---|---|---|---|
| Source | GCSS-A feed (LOGSA pull) | 21st TSC G4 data steward | 2026-02-15 |
| Ingestion | `gcss_a_ingest_transform` | Builder, SSG Petrov | 2026-02-01 |
| Raw Dataset | `gcss_a_raw` | Data Steward, CW2 Flores | 2026-02-15 |
| Transformed | `equipment_readiness_v3` | Builder, SSG Petrov | 2026-01-30 |
| Object Type | EquipmentReadinessRecord | C2DAO Ontology team | 2026-01-15 |
| Application | LOGSTAT Dashboard v2 | PM, MAJ Reyes | 2026-02-10 |

Maintain this table for every tier-1 product. Update it after every significant change. A dependency map six months stale is almost as dangerous as no map at all.

**NOTE — DDOF Friction Matrix (TM-40J Section 2-6):** TM-40J Section 2-6 provides a DDOF friction matrix identifying common organizational friction points between DDOF-defined roles during pipeline delivery. Use this matrix during project planning to anticipate coordination bottlenecks.

### 2-3. Types of Dependency Breaks

| Break Type | Description | Defense |
|---|---|---|
| **Noisy** | Pipeline fails with an error; Foundry logs the failure; builder gets alerted | Ensure resolution time is within SLA; communicate status to affected users |
| **Silent** | Pipeline runs without error but produces incorrect output (schema changed; transform no longer maps correctly) | Embed data quality checks in transforms; cultivate user reporting culture |
| **Slow** | Data that was timely yesterday is now 18 hours stale; a pipeline that ran in 4 minutes now takes 40 | Monitor for degradation, not only for failures; slow breaks become crises if undetected |

## SECTION 3 — HEALTH VS. STATUS — THE PM'S MONITORING FRAMEWORK

### 3-1. The Distinction

**Status answers: is it running? Health answers: is it doing what it is supposed to do?**

A pipeline can have a green status in Foundry's monitoring interface while producing stale, low-quality data that no one is using. That pipeline has a health problem, not a status problem.

### 3-2. Three Health Dimensions

**Availability** (Is it running?) - Are pipelines executing on schedule? - Are datasets refreshing within SLA? - Are Workshop applications accessible and loading within acceptable response time? - Are ingestion feeds arriving from source systems?

**Quality** (Is the data correct?) - Are data quality checks passing? - Is data completeness within accepted thresholds? - Are there anomalies or outliers suggesting upstream corruption? - Has the data been validated against a known-good source recently? - Do users report that the data matches their operational reality?

**Utility** (Is it being used for its intended purpose?) - Are users accessing the application at the expected frequency? - Are the operational questions the product was designed to answer actually being answered with it? - Has the user base shrunk since deployment? Why? - Has the operational requirement the product was built to support changed since it was built?

### 3-3. Applying the Framework: A V Corps Example

A readiness reporting pipeline built to support V Corps daily battle rhythm, running for seven months:

| Dimension | Question | Finding |
|---|---|---|
| Availability | Is the pipeline running? | Yes, daily at 0300Z |
| Availability | Are datasets refreshing on schedule? | Yes, within 15 minutes |
| Quality | Are data quality checks passing? | Partially — completeness check fails for 3 subordinate units |
| Quality | Do users trust the data? | Unknown — no user feedback collected since month 2 |
| Utility | Is it being used in battle rhythm? | Unclear — application access dropped 60% in month 5 |
| Utility | Is it still meeting operational requirements? | Unknown — S3 changed readiness reporting format in month 4; product not updated |

**Status: green. Health: degraded.** The PM who only checks status misses four significant findings.

### 3-4. Health Review Cadence

- Full three-dimension health review for all tier-1 products: monthly
- For products in active development or recent production release: availability and quality checks weekly
- Utility checks: quarterly at minimum, with direct user engagement — automated usage metrics alone do not capture whether the product is meeting its operational purpose

## SECTION 4 — MILESTONE AND MILESTONE RISK THINKING

### 4-1. How Data Projects Actually Fail

Data projects fail in predictable ways. The top five, in order of frequency in USAREUR-AF data programs:

1. **Data access lead time underestimated.** Assumed two weeks; took eight. Schedule never adjusted.

2. **Ontology design changed late.** Object Type modified by another team mid-build. Builder had to rework transforms. Three weeks lost.
3. **Governance approval took longer than planned.** C2DAO review scheduled for the last two weeks. Review identified naming violations requiring rework. Delivery slipped one month.
4. **User acceptance failed at delivery.** Product built against month-one requirements. By delivery in month four, the operational requirement had changed. Users rejected the product.
5. **Key personnel departed.** Sole builder PCS'd in month three. Knowledge transfer incomplete. Replacement spent six weeks reconstructing context.

These are not unpredictable risks. They are expected conditions for data programs in a theater environment. Build them into every project plan.

## 4-2. Milestones vs. Checkpoints

A milestone marks delivery. A checkpoint verifies that a delivery is working as intended.

| Milestone | Paired Checkpoint |
|---|---|
| Data ingestion pipeline deployed | Data completeness check passes; latency within SLA; three business days of clean execution confirmed |
| Ontology Object Type created | Object Type reviewed and approved by C2DAO data steward; no naming violations; link types validated |
| Workshop application delivered | User acceptance testing complete with actual operational users; access permissions confirmed; usage baseline established |
| Project close-out | Product owner identified; maintenance responsibility documented; dependency map filed in project record |

A milestone without a checkpoint is a declaration. A milestone with a checkpoint is a confirmation. Operational programs require confirmation.

## 4-3. Building Risk into Schedules

| Risk Category | Buffer | Notes |
|---|---|---|
| Data access lead time | Add 3–6 weeks | Especially for feeds originating outside USAREUR-AF (LOGSA, DCSA, external theater partners). Do not begin pipeline builds against data not yet confirmed accessible. |
| C2DAO review cycles | Add 2–3 weeks per promotion gate | Engage C2DAO before development begins to preview product design and surface objections early. |

| Risk Category | Buffer | Notes |
|---|---|---|
| Ontology coordination | Add 1–2 weeks | For new or modified Object Types requiring coordination with other teams who may have dependencies on the same types. |
| User acceptance | Add 1–2 weeks after delivery | With actual operational personnel. PMO sign-off ≠ user acceptance. |
| Personnel continuity | Document bus factor for projects > 3 months | Maintain knowledge transfer artifacts for any single-point-of-failure personnel. |

## SECTION 5 — MANAGING TECHNICAL DEBT IN A DATA PORTFOLIO

### 5-1. What Technical Debt Looks Like in Data Programs

Debt accumulates in places that do not generate immediate alerts: - Pipelines built quickly during DEFENDER or Spring Storm, never refactored or documented, now running in production as semi-permanent products - Object Types created by a builder who has since PCS'd, with unclear ownership and no documentation, yet downstream applications depend on them - Dashboards deployed for a specific operation six months ago, receiving near-zero traffic, still refreshing daily - Transforms written to accommodate a source system quirk that has since been fixed, but the workaround was never removed and now produces incorrect results - Governance documentation that exists but is six months out of date

Unlike code debt, data portfolio debt can directly affect operational users: the stale dashboard still referenced in a unit SOP; the Object Type whose schema no longer matches reality.

### 5-2. Identifying and Quantifying Debt

Conduct a portfolio debt audit twice per year. For each data product, answer:

| Question | Debt Indicator |
|---|---|
| Does it have a named, reachable owner? | No owner = governance debt |
| Is its documentation current (within 90 days of last change)? | Stale docs = documentation debt |
| Is it being actively used? | <10% of expected usage = potential retirement candidate |
| Were its governance artifacts completed? | Missing artifacts = governance debt |

| Question | Debt Indicator |
|---|---|
| Does it have any known quality issues not remediated? | Open quality issues = quality debt |

Assign each product a debt level: Low (0–1 flags), Medium (2–3 flags), High (4–5 flags). High-debt products require remediation plans or retirement decisions.

### 5-3. Making the Case to Leadership

Frame the debt case in operational risk language, not technical language:

"The readiness reporting pipeline for 21st TSC has no documented owner, no rollback procedure, and a known data quality issue affecting completeness for three subordinate units. If it fails during DEFENDER, we cannot reconstitute it without four to six days of rebuild time. The risk is an operational reporting gap during peak exercise tempo."

That argument is more persuasive than "we have technical debt that needs to be addressed."

Propose a debt-to-new-capability ratio for sprint allocation: 20% of builder capacity dedicated to debt remediation is a defensible baseline. When the portfolio audit surfaces high-debt products, negotiate a temporary increase.

## SECTION 6 — STAKEHOLDER MANAGEMENT FOR DATA PMS

### 6-1. Two Stakeholder Classes

**Operational users** (commanders, staff officers, NCOs who consume data products to make decisions) need to know: does the product work? Does it answer their operational question? Can they trust it?

**Technical producers** (builders, engineers, data stewards, C2DAO personnel who build the products) need to know: are requirements stable? Is the architecture sound? Is there adequate time for quality governance?

**The PM's translation function runs in both directions:**

| Direction | Translation |
|---|---|
| Operational → Technical | "The S3 needs a readiness view that shows equipment status by subordinate unit within V Corps. The access population is field-grade officers and above. The refresh requirement is daily by 0600Z." |
| Technical → Operational | "The dashboard you need requires data from three source systems, two of which are outside USAREUR-AF. Initial delivery will take six weeks. You will see a prototype by week three." |

The PM who fails to translate in both directions creates a gap that manifests as missed requirements, schedule surprises, and post-delivery rejection.

### 6-2. Managing the Delivery Tension

Operational users want delivery fast. Technical producers need time to build quality products that will not fail in production. Both positions are legitimate. The PM manages the tension through:

| Tool | Description |
|---|---|
| **Staged delivery** | A working prototype with partial data, delivered on the user's timeline, allows early feedback and reduces late-stage requirement changes. Frame it explicitly as a prototype — do not let a prototype drift into production without completing governance. |
| **Transparent constraints** | "Data access from LOGSA requires a formal request through NETCOM taking three to four weeks" is a concrete constraint users can accommodate. "It takes time" is not. |
| **Scope negotiation** | When a full-scope delivery cannot meet the user's timeline, negotiate scope reduction. What is the minimum viable product meeting 80% of the operational requirement? Deliver that on time; scope the remainder into a follow-on release. |

### 6-3. Managing a Multi-Team Ontology Coordination Scenario

**Scenario:** V Corps G4 requests a readiness dashboard requiring a new Object Type that will also be used by 21st TSC for their own logistics pipeline and by 10th AAMDC for IAMD equipment tracking. The 21st TSC data steward and 10th AAMDC data officer each have existing views on how that Object Type should be structured.

Without stakeholder management, this becomes a conflict that stalls both projects. With stakeholder management, the PM:

1. Identifies the shared dependency early — before either team begins building
2. Convenes a coordination session with V Corps G4, 21st TSC data steward, 10th AAMDC data officer, and C2DAO Ontology team
3. Facilitates agreement on the Object Type schema that meets both teams' needs, or negotiates a clear owner/consumer relationship
4. Documents the outcome and incorporates the coordination timeline into both project schedules
5. Follows up with C2DAO to ensure the agreed schema receives formal governance approval before any build begins

The PM does not design the Object Type. The PM creates the conditions for the right people to make the right decision on time.

# SECTION 7 — GOVERNANCE AS A PM TOOL, NOT A PM OBSTACLE

## 7-1. Why Bypassing Governance Is Borrowing Time

Under schedule pressure, governance looks like friction. A data product promoted to production without governance sign-off: - May have naming violations creating collision problems when another team builds against the same Ontology - Lacks a documented owner, creating an orphaned asset that no one will maintain - May bypass quality checks that would have caught data correctness issues before operational users encountered them - Will require retroactive remediation costing more time than the review would have taken — while the product is live, affecting operational users

The PM who bypasses governance to deliver faster is borrowing time, not saving it.

## 7-2. Using Governance Checkpoints as Milestone Gates

| Governance Event | Natural Project Gate |
|---|---|
| C2DAO Ontology design review | Gate before beginning Ontology build |
| Naming convention validation | Gate before promoting to staging |
| Data steward sign-off on quality checks | Gate before user acceptance testing |
| C2DAO promotion approval | Gate before production release |
| Data product ownership documentation | Gate before project close-out |

This mapping prevents governance events from being discovered late. A governance review that surfaces a naming issue before the staging build is a feature. The same review surfacing the same issue after 40 hours of staging build work is a bug.

**NOTE — Portfolio Health Metrics (TM-40J Section 7-6):** TM-40J Section 7-6 introduces a standardized portfolio health metrics framework with quantitative thresholds for availability, quality, and utility scoring. Reference when building or updating portfolio health tracking artifacts.

## 7-3. Engaging C2DAO as a Partner

The most effective data PMs treat C2DAO as a project partner, not a reviewer: - Brief C2DAO on upcoming projects at the start of planning, not at the start of governance review - Send C2DAO draft Ontology designs for informal feedback before formal submission, reducing the likelihood of formal review

failure - Incorporate C2DAO's institutional knowledge of the MSS architecture into project planning — they often know about existing Object Types or datasets that can reduce build scope - Communicate project timeline changes that affect C2DAO review scheduling as soon as they are known

C2DAO's formal role is reviewer. Their informal role, when the PM engages them correctly, is collaborator.

## SECTION 8 — PORTFOLIO VISIBILITY ON MSS

### 8-1. What the PM Can See in Foundry

| Visibility Surface | What It Shows | PM Use |
|---|---|---|
| Pipeline Lineage | Dependency chain from source to application for any dataset or Object Type | Primary tool for dependency mapping; identifies single points of failure |
| Dataset Health Indicators | Pipeline execution status, last refresh time, build errors at the dataset level | Availability health monitoring; datasets last-updated times significantly older than scheduled refresh cadence signal slow breaks |
| Branch and Promotion History | Branch creation, merge, and promotion events with who/what/when | Audit trail for compliance reviews; understand what was built when |
| Workshop Application Usage Metrics | Application access frequency and active user counts | Utility health monitoring; pull monthly and compare against expected usage baselines |
| Checks and Build Logs | Data quality check results and transform build logs | Quality health assessment; confirm checks are running and failures are routed to the responsible owner |

### 8-2. What the PM Cannot See in Foundry

Foundry tells the PM whether a product is running and how often it is accessed. It does not tell the PM whether the product is operationally valuable. Track these externally:

| Gap | External Tracking Method |
|---|---|
| User satisfaction | Periodic user engagement sessions (quarterly minimum); direct feedback at delivery |

| Gap | External Tracking Method |
|---|---|
| Operational impact | Coordination with unit S6/G6/G2 to understand whether products are influencing decisions |
| Requirement currency | Scheduled requirements validation with product owners (every 90 days for tier-1 products) |
| Stakeholder priorities | Regular engagement with G3/G4/G6 staff to understand shifting operational focus |
| Personnel continuity | Maintained contact list for each product's builder, data steward, and technical owner |

## 8-3. Building a Portfolio View Without Additional Tooling

A well-structured spreadsheet maintained weekly is sufficient for most USAREUR-AF data programs:

| Field | Purpose |
|---|---|
| Product Name | Official MSS name per naming convention |
| Tier | 1 (mission-critical), 2 (operationally significant), 3 (administrative) |
| Owner | Named individual, reachable |
| Status | Running / Degraded / Failed |
| Health (Availability) | Green / Amber / Red |
| Health (Quality) | Green / Amber / Red |
| Health (Utility) | Green / Amber / Red |
| Debt Level | Low / Medium / High |
| Last Governance Review | Date |
| Open Issues | Count |
| Notes | Free text |

Update weekly. Brief to leadership monthly. This is the PM's primary portfolio management artifact — it must answer any question a commander or senior leader asks about the MSS data portfolio within 60 seconds.

## SECTION 9 — COMMON DATA PM FAILURE MODES

**BLUF:** Each of the following failure modes shares a common root: the PM allowed urgency to override structure.

| Failure Mode | Pattern | Why It Happens | Prevention |
|---|---|---|---|
| **Requirements Without User Validation** | Product built against requirements captured from the commander or G-staff section; actual users — the analysts, NCOs, staff officers using the tool daily — have different needs | Requesters and users are often different people. Requirements captured from the top do not always reflect how the bottom works | Before accepting any requirements, conduct a user validation session with actual end users: What decisions will you make with this product? What does your current process look like? What would make you trust it? |
| **Committing to Timelines Before Data Access is Confirmed** | PM commits to a delivery date before confirming source system access. Data access requests take longer than expected. Delivery date slips. | Schedule pressure arrives before data access is confirmed. PM believes access will resolve quickly. | Treat data access confirmation as a project prerequisite, not a parallel track. Do not brief a delivery milestone to leadership until data access is either confirmed or a realistic timeline is established. |
| **Managing to Schedule at the Expense of Quality** | Project is behind schedule. Quality checks and user acceptance testing are compressed or skipped to recover timeline. Product ships with defects. Operational users encounter data quality issues. | Schedule pressure is visible and immediate. Data quality failures are delayed and diffuse. | Establish non-negotiable quality gates in the project plan before schedule pressure begins. When schedule slippage occurs, negotiate scope reduction rather than quality reduction. |
| **Losing Track of Production vs. Development** | Portfolio grows. PM loses clarity on which version of a product is in production, what changes are in flight, and what has been promoted vs. what is still in a development branch. | MSS Foundry manages branches and promotions, but portfolio state awareness requires deliberate PM tracking — which lapses in a busy program. | Maintain a production registry — a simple list of every product currently in production, its version, promotion date, and current health status. Review and update at each sprint close. |
| **Failing to Retire** | A data product built for a specific operation is not | Retirement requires a decision. Decisions | Every data product has a documented lifecycle status: Active, Under Review, |

| Failure Mode | Pattern | Why It Happens | Prevention |
|---|---|---|---|
| **Products** | retired when the operational requirement ends. It remains in production, accumulating governance debt. No one knows why it exists. | require ownership. If no one is explicitly responsible, the product persists by default. | or Retiring. When an operational requirement ends, the PM initiates a retirement review: confirm no downstream dependencies, notify affected users, document the retirement in the governance record. |

## SUMMARY: MENTAL MODEL REFERENCE

| Concept | Core Principle |
|---|---|
| PM Role | Own outcomes, not artifacts. Accountable for a working, governed, operational portfolio. |
| Dependency Thinking | Every product is the end of a chain. Trace it. Map it. Know who owns each link. |
| Health vs. Status | Status = running. Health = availability + quality + utility. Check all three. |
| Milestone Risk | Data access, Ontology changes, governance delays, and user acceptance are predictable risks. Build them in. |
| Technical Debt | Identify it. Quantify it. Make it visible to leadership in operational risk terms. |
| Stakeholder Management | Translate between operational users and technical producers. Manage the delivery tension with staged delivery, transparent constraints, and scope negotiation. |
| Governance | Use checkpoints as milestone gates. Engage C2DAO as a partner before formal review begins. |
| Portfolio Visibility | Foundry shows availability and quality. User engagement shows utility. Track both. |
| Failure Modes | Requirements without user validation. Timelines before access confirmed. Quality sacrificed for schedule. Lost production clarity. Unretired products. |

## TRANSITION TO SL 4J TASK INSTRUCTION

This guide has developed the conceptual foundation for data program management on MSS. You should now be able to:

- Describe the PM's scope of accountability and how it relates to C2DAO governance

- Trace a data product's dependency chain from source system to operational user
- Distinguish between status and health and apply a three-dimension health framework
- Build milestone risk into a project schedule before schedule pressure begins
- Identify and quantify technical debt in a data portfolio
- Translate between operational users and technical producers
- Use governance checkpoints as milestone gates rather than schedule obstacles
- Build portfolio visibility using native Foundry tooling supplemented by direct stakeholder engagement
- Recognize and prevent the five most common data PM failure modes

Proceed to SL 4J for task-based instruction on executing these responsibilities within the MSS environment.

## GOVERNING REFERENCES

The following publications establish the policy, architectural, and governance framework within which SL 4J program managers operate.

| Publication | Title | Relevance |
|---|---|---|
| Army CIO Memorandum | Data and Analytics Policy (April 2024) | Data governance authority |
| UDRA v1.1 | Unified Data Reference Architecture (February 2025) | Technical reference architecture |
| DoD Data Strategy | DoD Data Strategy (2020) | Enterprise data management framework |
| USAREUR-AF C2DAO Guidance | Command governance for data operations | Operational governance |
| Army DIR 2024-03 | Digital Engineering Policy | Army digital transformation directive |
| AR 25-1 | Army Information Technology | IT governance and data management policy |
| learn-data.armydev.com | CDA Portal | Training platform reference |

## CURRICULUM NOTES

**Prerequisite:** SL 3 (Advanced Builder) is REQUIRED — not recommended. The PM who has not built on MSS cannot accurately estimate effort, identify technical blockers, or write meaningful acceptance criteria for MSS deliverables.

**Advanced track:** SL 4J graduates should pursue **SL 5J (Advanced Program Manager)** for advanced topics including multi-program portfolio management, enterprise data governance leadership, cross-command integration coordination, and MSS capability roadmap ownership.

**Peer specialist cross-references:** The PM coordinates delivery across all specialist tracks: - **SL 4G (ORSA):** ORSA analytical products have specific data quality, methodology review, and product delivery requirements that affect acceptance criteria and sprint scope. - **SL 4H (AI Engineer):** AI workflow projects require C2DAO governance milestones and human-review-gate design as non-negotiable project requirements. - **SL 4M (ML Engineer):** ML projects have model governance checkpoints (Training Data Card, Model Card, C2DAO review) that must be built into every project schedule. - **SL 4K (Knowledge Manager):** KM systems have retention policy, taxonomy governance, and workflow authority requirements that affect project scope and change management. - **SL 4L (Software Engineer):** Production code has CI gate, peer review, and branch-promotion requirements that affect delivery timelines and cannot be compressed without creating operational risk.

**NOTE — New Doctrine Content in SL 4J:** SL 4J now includes DDOF roles and PM oversight responsibilities (section 1-2a), the DDOF Configuration Management Friction Matrix (section 2-6) for identifying execution blockers, and data product portfolio health metrics (section 7-6) covering VAULTIS-A scores, gate rates, and retirement trigger monitoring.

**WFF awareness:** Data products managed by SL 4J PMs are consumed by WFF-qualified users (SL 4A through SL 4F — Intelligence, Fires, Movement and Maneuver, Sustainment, Protection, and Mission Command). Every project schedule decision has a WFF operational consequence. Know which WFF function depends on each product and what the operational impact of degradation or delay is.

*CONCEPTS GUIDE — SL 4J COMPANION | DATA PROGRAM MANAGER | MAVEN SMART SYSTEM HQ USAREUR-AF, Wiesbaden, Germany | 2026 DISTRIBUTION RESTRICTION: Distribution authorized to U.S. Government agencies and their contractors only. Other requests must be referred to Headquarters, C2DAO, Wiesbaden, Germany.*