TECHNICAL MANUAL

SL 5J

USAREUR-AF

TM-50J — ADVANCED PROGRAM MANAGER (TECHNICAL)

Specialist Course Manual

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

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

TM-50J — ADVANCED PROGRAM MANAGER (TECHNICAL)

Forward: SL 5J qualifies Technical Program Managers to lead the USAREUR-AF MSS data capability at scale — managing multiple concurrent product teams, directing portfolio investment, and advising senior leaders on data as an operational asset. **Prereqs:** SL 4J, Program Manager (Technical) (required). Graduates of SL 4J who have managed at least one full-lifecycle MSS data or AI project from initiation through production release, with demonstrated proficiency in Agile delivery, stakeholder management, and MSS platform governance. Familiarity with SL 4G (ORSA), SL 4H (AI Engineer), SL 4M (ML Engineer), SL 4K (Knowledge Manager), and SL 4L (Software Engineer) track content strongly recommended; CONCEPTS_GUIDE_TM50J_PROGRAM_MANAGER_ADVANCED (read before this manual). *HQ USAREUR-AF · v1.0 · 2026 · DISTRIB: USG only · AUTH: C2DAO/UDRA v1.1*

WARNING: SL 5J program managers are accountable for the health of the USAREUR-AF data capability portfolio. Decisions made at this level — team structures, vendor deliverable acceptance, product retirement, and GO/SES briefings — have direct operational consequences for USAREUR-AF and EUCOM mission readiness. Apply the same standard of care as an operational mission brief: confirm facts, coordinate with stakeholders, and communicate clearly before execution. CAUTION: Multi-team Agile coordination (SAFe or equivalent) requires explicit dependency management across teams. An undocumented inter-team dependency is a hidden risk. Failure to surface and manage cross-team dependencies during planning creates blocked sprints, delayed releases, and degraded data product quality. NOTE: This manual assumes full mastery of SL 4J content. Chapter references to SL 4G through SL 4O indicate the technical tracks managed by SL 5J program managers — not prerequisites for the PM role itself. Cross-references are provided to help SL 5J graduates understand the work they are overseeing.

CHAPTER 1 — INTRODUCTION AND SCOPE

1-1. Advanced Program Manager Manual

BLUF: SL 5J qualifies Technical Program Managers to lead the USAREUR-AF MSS data capability at scale — managing multiple concurrent product teams, directing portfolio investment, and advising senior leaders on data as an operational asset.

SL 4J qualified you to manage a project. SL 5J qualifies you to manage a program. The distinction is not cosmetic. A project has a defined scope, timeline, and team. A program is a persistent organizational capability — a collection of projects, products, teams, and people operating simultaneously toward a theater-level mission. Managing a program requires different skills, different tools, and a different mental model.

At the SL 5J level, your job is not to execute sprints — your team leads do that. Your job is to create the conditions in which multiple teams can execute sprints simultaneously, without stepping on each other, without competing for the same resources, and without delivering products that contradict each other. You set standards. You resolve conflicts SL 4J PMs cannot resolve themselves. You manage the interfaces that connect data products to each other and to the command. You maintain the health of the portfolio so that when a commander asks "what can MSS tell me about force readiness?" the answer reflects coherent, validated, current data — not six disconnected dashboards built by six teams who never talked to each other.

1-2. Scope of SL 5J

SL 5J covers the following capability areas:

Table 1-1. SL 5J Capability Areas

Area	Description
Scaled Agile	Coordinating multiple Agile product teams at program level using PI Planning and dependency management
Portfolio Tracking	Designing and operating theater-level MSS dashboards for program-wide visibility
Advanced Stakeholder Management	Managing relationships and communications at GO/SES/SES-equivalent level; building data culture
Quantitative Delivery Management	Using velocity, cycle time, throughput, and predictability data to manage team performance
Technical Debt Strategy	Assessing, prioritizing, and communicating technical debt across the portfolio
Build/Buy/Configure	Decision frameworks for when to write code vs. use platform features vs. procure capability
Vendor Management	Managing Palantir task order delivery, evaluating contractor deliverables, accepting/rejecting work
Team Design	Structuring ORSA, MLE, SWE, KM, and PM roles for program-scale delivery
PM Leadership	Training junior PMs, establishing program standards, conducting PM health assessments

Area	Description
Product Lifecycle	Version management, deprecation, and retirement of MSS products

1-3. Relationship to SL 4J and the Technical Tracks

SL 5J program managers oversee the work of SL 4J project managers and the technical practitioners in tracks SL 4G through SL 4O. Understanding what each track produces — and what failure looks like in each — is essential for program-level oversight.

Table 1-2. Technical Tracks Managed by SL 5J

Track	Role	What They Build	Common Failure Mode
SL 4G	ORSA	Analytical models, campaign analysis, optimization	Over-engineered models that outrun user comprehension
SL 4H	AI Engineer	AI pipeline architecture, model deployment, AIP	Technically correct pipelines that solve the wrong problem
SL 4M	ML Engineer	ML model development, training, evaluation	Models that degrade silently in production
SL 4J	Tech PM	Project delivery, stakeholder management, product release	On-time delivery of the wrong product
SL 4K	Knowledge Manager	Ontology, data governance, KM architecture	Perfect data models no one uses
SL 4L	Software Engineer	Custom transforms, integrations, API layers	High-quality code solving problems that could be configured
SL 4N	UI/UX Designer	Workshop applications, user interfaces, dashboards	Beautiful interfaces that don't solve the operational problem
SL 4O	Platform Engineer	Platform configuration, DevSecOps, infrastructure	Over-automated environments that operators cannot troubleshoot

SL 5J graduates must recognize failure modes across all tracks and understand how to intervene at the program level — not by doing the technical work, but by restructuring teams, realigning priorities, removing blockers, or escalating to C2DAO leadership.

1-3A. Relationship to Other SL 5 Publications

Publication	Track	Key Overlap with SL 5J
SL 5G	ORSA Advanced	Portfolio-level analytical program governance; OR product review

Publication	Track	Key Overlap with SL 5J
SL 5H	AI Engineer Advanced	AI program lifecycle management; governance acquisition (Chapter 8)
SL 5M	ML Engineer Advanced	ML program portfolio management; automated pipeline governance
SL 5J	Program Manager Advanced	THIS DOCUMENT
SL 5K	Knowledge Manager Advanced	KM system program oversight; lessons learned program management
SL 5L	Software Engineer Advanced	Platform engineering program coordination; SWE team structure
SL 5N	UI/UX Designer Advanced	Design team coordination; UX requirements management
SL 5O	Platform Engineer Advanced	Platform capacity planning; infrastructure program management

WFF Operational Consumer Note. The six Warfighting Function (WFF) tracks — Intelligence (SL 4A), Fires (SL 4B), Movement and Maneuver (SL 4C), Sustainment (SL 4D), Protection (SL 4E), and Mission Command (SL 4F) — represent the primary operational stakeholder base for MSS programs. SL 5J program managers must understand WFF requirements when scoping data products, managing stakeholder relationships, and communicating program value to GO/SES audiences. WFF staff sections define the demand signal; the data program exists to serve their mission decisions.

1-4. Operating Environment

The USAREUR-AF MSS operates in an environment characterized by:

- **High tempo and competing priorities.** Operational staff have limited tolerance for long feedback cycles. Program managers must balance delivery speed against quality.
- **Personnel turbulence.** Military personnel rotate on 2-3 year cycles. Program continuity depends on documentation, standards, and onboarding processes — not on any individual.
- **Command climate sensitivity.** Delivering accurate but unwelcome data to senior leaders requires both technical credibility and interpersonal skill.
- **Vendor partnership complexity.** Palantir delivers capability through task orders executed by contractors and government personnel simultaneously. The PM must manage both.
- **Evolving policy environment.** Army CIO data policy (April 2024) and UDRA v1.1 (February 2025) establish the framework, but platform capabilities evolve faster than policy. Program managers must interpret intent, not just follow procedure.

1-4A. The SL 5J Program Manager Profile

The SL 5J program manager is not the most technically skilled person on the team. The ORSA (SL 5G), AI Engineer (SL 5H), or ML Engineer (SL 5M) will typically have deeper technical expertise in their domains. The SL 5J's value is integrative: the ability to hold a coherent picture of the entire program simultaneously, to make decisions under uncertainty with incomplete information, and to maintain the trust of both senior leaders and technical practitioners.

Table 1-3. SL 5J Competency Profile

Competency Domain	Key Behaviors	Development Path
Program-level thinking	Sees portfolio risk, not just project risk; understands how individual decisions affect the whole	Managing 3+ concurrent SL 4J-level projects; PI Planning facilitation experience
Technical fluency	Understands enough of each technical track to ask the right questions and recognize when answers are implausible; reads code and data schemas	Completion of SL 4J; familiarity with SL 4G through SL 4O content; regular sprint reviews
Senior stakeholder management	Comfortable briefing GO/SES; can deliver bad news professionally; maintains trust under pressure	Direct GO/SES briefing experience; mentorship from senior military or SES leaders
Quantitative reasoning	Uses delivery data analytically; interprets statistical signals; communicates metrics in operational terms	Statistical literacy; data product familiarity; Chapter 5 of this manual
Change leadership	Reads organizational resistance; designs change for the military context; sustains change through personnel turbulence	Field experience with technology adoption in military formations
Vendor management	Understands task order mechanics; can evaluate deliverables technically; manages the Palantir relationship as a partner, not a dependency	Hands-on task order administration experience; Chapter 7 of this manual

SL 5J is not a certification that can be earned in a classroom. It is a capability that develops through repeated application — through PI Planning cycles, through program health crises navigated, through GO briefings delivered and difficult conversations had. This manual provides the framework. The development happens in the work.

1-5. How to Use This Manual

This manual is organized by capability area, not by chronological workflow. Program managers should read Chapters 1 through 3 sequentially to establish the program management foundation. Chapters 4 through 8 may be addressed in priority order based on current program needs.

Each chapter contains: - Background and framework (conceptual grounding) - Task procedures with Conditions, Standards, Equipment, and Steps - Tables for reference use - WARNING, CAUTION, and NOTE callouts

Appendix A (Program Health Assessment) and Appendix B (Product Lifecycle Guide) are operational references — use them regularly, not just when things go wrong.

CHAPTER 2 — SCALED AGILE: MANAGING MULTIPLE CONCURRENT TEAMS

2-1. From Project to Program

BLUF: Managing one Agile team is project management. Managing five simultaneously is program management. The mechanics differ at every level — planning, execution, dependency management, metrics, and communication.

A single Agile team manages its own backlog, holds its own ceremonies, and tracks its own velocity. When you add a second team, you introduce interface risk: what happens when Team A's pipeline is a prerequisite for Team B's dashboard? Who resolves the conflict when both teams want the same Foundry dataset, but Team A wants to restructure it for their model and Team B needs the existing schema for their production dashboard?

These are program-level problems. No SL 4J PM can resolve them alone — because the answer may require trading off one team's sprint plan against another's. Only the program manager, with visibility across all teams, can make that call.

2-2. Program Increment (PI) Planning

PI Planning is the scaled Agile ceremony that aligns multiple teams to a common set of program-level objectives for a fixed interval (typically 8-12 weeks on MSS). PI Planning is not sprint planning at scale — it is a structured coordination event where teams expose their dependencies to each other and commit to a shared set of Program Increment Objectives (PIOs).

TASK 2-2A: CONDUCT PI PLANNING FOR MSS PROGRAM

CONDITIONS: Program manager has assembled team leads for all active MSS product teams. A draft list of program-level priorities and stakeholder requests has been collected for the upcoming PI interval. MSS Workshop is configured with a program-level tracking ontology (see Chapter 3).

STANDARDS: PI Planning is complete when: all teams have documented their sprint objectives for the PI; all inter-team dependencies are identified, assigned an owner, and logged in the program tracker; program-level risks are documented with mitigations; and GO/SES-visible Program Increment Objectives are drafted and approved by C2DAO leadership.

EQUIPMENT: MSS Workshop (program tracking dashboard), MSS Pipeline Builder (dependency visualization), shared video conferencing if distributed, physical or virtual dependency board.

PROCEDURE:

1. **Prepare the PI Planning brief (T-2 weeks).** Compile stakeholder requests received since the last PI Review. Prioritize by operational impact and strategic alignment with USAREUR-AF Commander's data priorities. Draft 5-8 Program Increment Objectives (PIOs) — measurable outcomes the program will deliver by end of PI. Circulate draft PIOs to C2DAO leadership for alignment before the planning event.
2. **Distribute team capacity information (T-1 week).** Collect each team's available capacity for the upcoming PI: number of developers, leave schedules, TDY, known platform maintenance windows, and contractor availability. Normalize capacity to story points or ideal engineering days per team. Share across all team leads.
3. **Open PI Planning with a program briefing (Day 1, Morning).** Brief the full assembled team on: USAREUR-AF Commander's priorities for the PI, current program health status, lessons learned from the previous PI, and draft PIOs. Allow team leads to ask questions about priority and scope. The program briefing should take no longer than 90 minutes.
4. **Conduct team breakouts (Day 1, Afternoon).** Teams plan their sprint objectives for each sprint in the PI. Each team identifies: features they commit to deliver, features they will attempt but cannot guarantee (stretch objectives), and — critically — their dependencies on other teams or on Palantir/contractor deliverables.
5. **Run the dependency board session (Day 2, Morning).** All teams assemble. Each team calls out dependencies on other teams. Record each dependency with: the requesting team, the providing team, the deliverable, the sprint in which it is needed, and the risk if the dependency is not met. Visually map dependencies — a simple matrix or string diagram is sufficient. Identify circular dependencies immediately and resolve before PI Planning closes.
6. **Negotiate and resolve conflicts (Day 2, Morning, continued).** For each dependency conflict — two teams competing for the same resource, a team asking for a deliverable another team cannot commit to — the program manager facilitates resolution. Options include: re-sequencing work, reducing

scope, pulling in a contractor resource, or elevating to C2DAO for priority decision. Document all resolutions.

7. **Finalize PIO commitments (Day 2, Afternoon).** Teams present their committed sprint objectives. Program manager validates that team commitments are coherent, that dependencies are resolved, and that the aggregate of team commitments supports the program PIOs. Adjust PIOs if warranted — it is better to set accurate expectations than to over-commit to leadership.
8. **Conduct PI Planning retrospective (Day 2, Closing).** 15 minutes. What went well in planning? What needs to change for the next PI? Document action items.

NOTE: PI Planning for a distributed military team often requires asynchronous pre-work due to time zone differences across USAREUR-AF. Build in one week of asynchronous dependency identification before the synchronous planning event to reduce Day 2 conflict resolution time.

CAUTION: Do not allow individual teams to use PI Planning as an opportunity to surface pet projects or low-priority features. The program manager must maintain priority discipline. Any feature not traceable to a PIO or a documented stakeholder request should be deferred to the program backlog.

2-3. Managing Dependencies Across Teams

Inter-team dependencies are the primary source of program-level risk. A single unresolved dependency can block multiple teams simultaneously — cascading from one stalled sprint into a delayed release for the entire program.

Table 2-1. Dependency Risk Classification

Dependency Type	Description	Risk Level	Management Approach
Data dependency	Team B requires a dataset or ontology object that Team A is building	High	Resolve during PI Planning; establish mock data interface for parallel development
Platform resource	Two teams require the same Foundry resource simultaneously (compute, storage quota, admin access)	Medium	Schedule access windows; coordinate with C2DAO for resource allocation
Personnel dependency	Team A's tech lead is the only person who can unblock Team B's architecture question	Medium	Document knowledge; establish a program-level SME registry
Contractor deliverable	Team A requires a Palantir-delivered component before they can begin a sprint	High	Establish acceptance criteria in advance; build 2-sprint buffer into planning

Dependency Type	Description	Risk Level	Management Approach
Approval gate	A compliance review, C2DAO sign-off, or stakeholder approval blocks Team B	Medium	Map all approval gates at PI Planning; initiate early where possible
External feed	A pipeline depends on a data feed from outside the MSS (S2, G4, partner nation systems)	Very High	Establish SLA with feed owner; build resilience into pipeline design (SL 4L)

TASK 2-3A: MAINTAIN THE PROGRAM DEPENDENCY REGISTER

CONDITIONS: PI Planning is complete. A program tracking workspace exists on MSS Workshop.

STANDARDS: The dependency register is current within 48 hours of any status change. Each dependency has a single owner who is accountable for its resolution. All blocked dependencies are escalated to the program manager within one sprint of blocking.

EQUIPMENT: MSS Workshop (program tracking workspace), dependency register object type in program ontology, access for all team leads to update dependency status.

PROCEDURE:

1. Export the dependency map from PI Planning into the MSS program tracker. Each dependency becomes a tracked object in the program ontology (see Chapter 3 for ontology design).
2. Assign each dependency an owner — typically the lead of the team with the greater exposure to blocking.
3. At each sprint review (typically bi-weekly), require each dependency owner to provide a one-sentence status: Green (on track), Amber (at risk, mitigation in place), Red (blocked, escalation needed).
4. For any Red dependency, the program manager conducts a direct conversation with both team leads within 24 hours. Options: re-sequence, reduce scope, pull in additional resource, or formally defer the dependent feature to the next PI.
5. Close dependencies when both teams confirm the deliverable was received and accepted. Do not close on promise — close on delivery.

NOTE: A dependency register that is not actively maintained provides false assurance. Program managers who discover blocked dependencies through sprint failures — rather than through the register — have lost the early warning function. Protect the register's accuracy.

2-3A. Conflict Resolution Between Teams

Inter-team conflict is inevitable when multiple teams share resources, depend on each other's outputs, or compete for the same stakeholder attention. The program manager is the court of last resort for conflicts that team leads cannot resolve themselves. Waiting for the program manager to resolve every

disagreement creates a bottleneck. Preventing conflicts from escalating requires a clear protocol.

Table 2-3. Inter-Team Conflict Resolution Protocol

Conflict Type	First Resolution Attempt	Escalation Path	Timeline
Technical disagreement (architecture, naming, schema)	Team leads + Platform Lead (SL 5K/SL 5L) resolve jointly	Program Manager facilitates if unresolved in 48 hours	Resolve within 1 sprint
Resource contention (shared dataset, tool access)	C2DAO resource coordinator	Program Manager reallocates or escalates	Resolve within 1 sprint
Priority conflict (two teams competing for same backlog slot with PM)	Program Manager makes priority decision based on operational impact	C2DAO Chief if GO-level stakeholder interests conflict	Resolve before next sprint planning
Dependency dispute (Team A says they delivered; Team B says it's not usable)	Joint walkthrough with both team leads and the relevant technical practitioner	Program Manager reviews documented acceptance criteria	Resolve within 48 hours of dispute surfacing
Scope dispute (who owns a given capability or data domain)	Program Manager determines ownership per product roadmap	C2DAO if domain involves policy-level decisions	Resolve before PI Planning of the affected PI

NOTE: Most inter-team conflicts are symptoms of insufficient clarity at PI Planning — unclear ownership, undocumented interfaces, or uncommitted dependencies. The best conflict resolution strategy is prevention: invest heavily in clarity during PI Planning and the program will generate fewer conflicts during execution.

2-4. Cross-Team Sprint Ceremonies

Scaled Agile requires additional ceremonies beyond the single-team standard. SL 5J program managers must design a ceremony cadence that provides cross-team coordination without ceremony overload.

Table 2-2. Program-Level Ceremony Cadence

Ceremony	Frequency	Duration	Participants	Purpose
Program Sync	Weekly	30-45 min	All team leads + PM	Surface cross-team blockers; dependency status; critical path updates
Sprint Review (Program)	Every 2 sprints	90 min	All teams + C2DAO rep + key stakeholders	Demo delivered features; update program dashboard; adjust PI backlog

Ceremony	Frequency	Duration	Participants	Purpose
PI Planning	Every PI (8-12 weeks)	2 days	All teams + C2DAO + leadership	Plan, align, and commit to next PI objectives
PI Review / Retrospective	End of PI	Half day	All teams + C2DAO	PI objective assessment; lessons learned; adjust team structures
Program Retrospective	Quarterly	2 hours	Program manager + all team leads	Process improvement; standards updates; PM capability development

CAUTION: Ceremony overload is a real risk in scaled Agile environments. If teams are spending more than 20% of their capacity in meetings, the ceremony cadence is too heavy. Audit the meeting load quarterly and cut any ceremony that is not producing actionable outputs.

CHAPTER 3 — PORTFOLIO-LEVEL TRACKING AND THEATER VISIBILITY

3-1. Why Portfolio Tracking Matters

BLUF: Senior leaders cannot manage what they cannot see. The program manager's responsibility is to maintain a single, authoritative source of truth for MSS program health — one that is always current, always accurate, and always accessible to the right people.

At the SL 4J level, project tracking is focused on one team's sprint board, backlog, and burn-down. At the SL 5J level, the audience changes. You are building dashboards for the USAREUR-AF G6, the C2DAO Chief, and potentially the DCG or CG. These leaders do not need sprint-level detail. They need answers to four questions:

1. Are we delivering capability on schedule?
2. Are there risks that require leadership attention?
3. Are we using resources efficiently?
4. What has been delivered since the last briefing?

The program tracking architecture must answer all four questions without requiring the reader to dig through raw data.

3-2. Program Tracking Ontology Design

The MSS program tracker lives in Foundry. Its data model — the ontology — determines what you can and cannot measure. Designing the ontology correctly at the start of the program is far cheaper than restructuring it after six months of data have been collected.

Table 3-1. Core Program Tracking Object Types

Object Type	Key Properties	Links
Program	Name, C2DAO owner, PI current, status	→ Projects
Project	Name, PM, sprint current, status, team	→ Program, Features, Dependencies
Feature	Title, team, sprint target, sprint actual, status, story points estimated, story points delivered	→ Project, Dependencies
Dependency	Description, requesting team, providing team, due sprint, status, risk level	→ Feature (from), Feature (to)
Risk	Title, category, probability, impact, mitigation, owner, status	→ Project
Team	Name, team lead, capacity (points/sprint), track (ORSA/MLE/SWE/KM/PM)	→ Project
Milestone	Title, target date, actual date, status	→ Project, Program
Stakeholder	Name, rank/grade, unit, communication preference	→ Project

NOTE: Build the ontology to the minimum viable schema first. Add properties only when you can demonstrate they will be populated and queried. A sparse, complete ontology is more valuable than a rich, incomplete one.

TASK 3-2A: DESIGN AND BUILD THE PROGRAM TRACKING WORKSPACE

CONDITIONS: C2DAO has approved the program tracking workspace. The program manager has a list of active projects, teams, and current PI objectives. SL 4J PMs on each project team are responsible for populating project-level data.

STANDARDS: The workspace is operational when: all active projects are represented with current status; all teams and team leads are registered; all PI features are tracked with estimated and actual story points; program health roll-up is visible in a single-view dashboard accessible to C2DAO leadership.

EQUIPMENT: MSS Foundry (Ontology Manager, Pipeline Builder, Workshop), program ontology schema document, C2DAO access list.

PROCEDURE:

1. **Define the ontology schema.** Using Table 3-1 as a baseline, finalize the object types and properties for your specific program. Add any program-specific attributes (e.g., classification level, AOR of primary user unit, operational system dependency). Document the schema before building — changes to the ontology after data population require migration.
2. **Build object types in Ontology Manager.** Create each object type. Define properties with correct data types: string, integer, boolean, date, enum. For status fields, use controlled vocabularies (Green/Amber/Red) rather than free text — free text status fields produce unquantifiable data.
3. **Configure object links.** Establish the relationships in Table 3-1. Link direction matters for query performance and Workshop widget behavior. Test each link by creating a sample object and verifying the link resolves correctly.
4. **Build ingestion pipelines.** For any data that can be automated — sprint velocity from a connected tool, feature completion dates from a project management system, capacity data from a personnel tracker — build Pipeline Builder transforms to ingest and normalize the data. For data that requires manual entry, build Workshop input forms.
5. **Build the program health dashboard in Workshop.** Create three views: (a) Executive Summary — one-page health rollup for GO/SES audience; (b) Program Operations — PI progress, dependency status, risk register for PM and C2DAO daily use; (c) Team View — sprint board and backlog for individual team leads.
6. **Configure access.** Apply least-privilege principles. GO/SES leaders receive read-only access to the Executive Summary view. Team leads receive write access to their own project objects. The program manager receives full write access.
7. **Validate with stakeholders before going live.** Brief the C2DAO chief on the dashboard before the first formal program review where it will be used. Confirm the labels, metrics, and visualizations match their expectations. A dashboard that confuses its senior audience on first use loses credibility it may never regain.

WARNING: Do not populate the program tracker with aspirational data. Status fields must reflect actual current state, not planned state. A dashboard that shows all green because PMs are reporting planned status — not actual delivery — is worse than no dashboard. It misinforms leadership and delays corrective action.

3-3. Designing for GO/SES Readability

Senior leaders scan before they read. The executive summary view must communicate program health in under 30 seconds of reading time. Design principles:

Table 3-2. Executive Dashboard Design Principles

Principle	Application
One page, one screen	The executive view must fit on a single Workshop page without scrolling
Color-coded status	Red/Amber/Green for each program objective; no status without a color
Trend, not just status	Show direction: is it getting better or worse? Include a sparkline or delta indicator
Exception-oriented	Highlight what requires leadership attention; do not bury problems in a wall of green
Time-stamped	All data must display the time of last update; stale data is worse than no data
Action-linked	Each red item should link to a risk register entry with a documented mitigation owner

3-3A. Data Quality in the Program Dashboard

A program dashboard is only as trustworthy as the data feeding it. Program managers who present dashboards to senior leaders without validating the underlying data quality are creating a liability. When a GO challenges a number in the dashboard — and they will — the program manager must be able to say: "That data is sourced from [X], updated [Y], and has a known quality characteristic of [Z]." The inability to answer that question destroys the dashboard's credibility immediately.

Table 3-4. Dashboard Data Quality Standards

Data Element	Required Documentation	Update Frequency	Quality Indicator
Sprint velocity (per team)	Source: sprint review records; PM-entered	Per sprint	Green if entered within 24 hours of sprint review
PI objective status	Source: PM assessment; PM-entered	Weekly	Green if updated within 7 days
Dependency status	Source: team lead assessment; team lead-entered	Every Program Sync	Amber if any dependency not updated in 10+ days
Risk register	Source: PM/team lead-entered	As risks change; reviewed weekly	Red if any risk has no mitigation owner
Escaped defects	Source: production incident log	Per incident	Green if all incidents logged within 24 hours
Team capacity	Source: PM-entered per PI	Per PI	Green if current PI capacity is confirmed

NOTE: Build data freshness indicators into every dashboard widget. A timestamp showing "Last updated: 14 days ago" is more honest — and more useful — than a stale green indicator with no date. Senior leaders who see stale data will question whether the dashboard reflects reality. Make staleness visible; it creates accountability for data entry.

3-4. Program Review Briefing Package

The program review brief is not a Workshop dashboard printout. It is a commander's brief format adapted for a data program context: BLUF, current status, key risks, decisions required. The dashboard supports the brief — it does not replace it.

Table 3-3. Program Review Brief Structure

Slide/Section	Content	Audience
BLUF	One paragraph: PI objectives status, overall program health, top risk	CG/DCG/G6
PI Objective Status	Table: each PIO with status (Green/Amber/Red), % complete, delta from last review	C2DAO, G6
Key Deliveries	What was delivered since last review: features, capabilities, user groups reached	CG/G6
Top Risks	Top 3 risks requiring leadership attention; mitigation status; decision requested (if any)	CG/DCG
Decisions Required	Explicit: what decision is needed from this audience, and by when	CG/DCG
Looking Ahead	Next PI top 3 capability investments; major milestones	CG/G6

CHAPTER 4 — ADVANCED STAKEHOLDER MANAGEMENT AND DATA CULTURE

4-1. Managing Up: Briefing GO/SES Leadership

BLUF: Senior leaders invest in what they understand. Your ability to make data capability tangible — and credible — at the GO/SES level determines the program's long-term resource and political viability.

SL 4J stakeholder management focused on working-level relationships: requirements gathering from G-staff analysts, user story validation with battalion S2/S3 shops, and change management with end users adopting new dashboards. SL 5J stakeholder management operates at a different level. You are now communicating with Generals, Senior Executive Service Civilians, and senior NATO partners who set budget priorities, sign task orders, and determine whether the MSS program grows or contracts.

The rules change at this level:

- **Depth on demand, summary by default.** Senior leaders want BLUF. They will ask for depth when they want it. Do not lead with methodology.
- **Know your audience's decision horizon.** A GO is making decisions for the next 6-24 months. Frame everything in terms of capability and mission impact — not sprints, story points, or pipeline architecture.
- **Bring options, not problems.** When you escalate a risk to a GO, arrive with a recommended course of action and two alternatives. "We have a problem" without options wastes senior leader time.
- **Own the bad news.** Senior leaders who are surprised by program failures — especially failures the PM knew about and did not communicate — lose trust permanently. Deliver bad news early, with context and a mitigation plan.

4-2. Building Organizational Buy-In for Data Culture Change

NOTE — Palantir Developers reference: *Building Enterprise Autonomy with Shyam Sankar, CTO* — A strategic framing of enterprise-scale AI deployment from Palantir's CTO perspective, covering the organizational and leadership conditions required for autonomous data capabilities to take root. Relevant context for PMs building organizational buy-in for data culture change in this section. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

BLUF: Technology adoption in a military organization is a command climate problem, not a technology problem. Your job is to change behavior, not just deploy software.

MSS is not a self-adopting platform. Units that have operated with SharePoint spreadsheets and slide decks for years will not abandon those workflows because a new dashboard exists. Data culture change requires sustained, deliberate effort over 12-24 months — not a deployment event.

TASK 4-2A: DEVELOP AND EXECUTE A DATA CULTURE CHANGE PLAN

CONDITIONS: A new MSS capability has been developed and is ready for operational deployment to a formation that has not previously used MSS as part of their workflow.

STANDARDS: Change plan is complete when: all affected user roles are identified; a communication plan is defined (who, what, when, by what channel); a training plan is defined (who delivers, what format, what outcomes); success metrics are defined (adoption rate, active users, workflow change indicators); a 90-day post-deployment review is scheduled.

EQUIPMENT: MSS Workshop (user analytics, if configured), stakeholder registry, program communication channels.

PROCEDURE:

1. **Map the current state workflow.** Before designing the change, understand what you are replacing. Interview two to four users in the affected unit. Document: what data sources they use today, how they compile and format their products, what decisions they make with those products, and what their

current pain points are. You cannot sell a solution until you understand the problem from the user's perspective.

2. **Identify the change champions.** In every formation, there is a person — usually a mid-grade NCO or warrant officer — who is trusted by peers and who adopts new technology faster than average. Find that person. Bring them into the development process early. A grassroots champion is worth ten top-down mandates.
3. **Design the communication cadence.** Use the rule of seven: users need to encounter a new capability approximately seven times before they act. Plan communications across multiple channels: email, unit brief slide, TC/TM distribution, in-person demo at a staff meeting, a video walkthrough posted to the unit shared drive, a helpdesk tip-of-the-week. Do not rely on a single all-hands email.
4. **Stage the rollout.** Do not deploy to an entire theater formation simultaneously. Identify a pilot unit — ideally one with a data champion and a motivated S6 or G6. Deploy there first. Collect feedback. Fix issues. Then expand.
5. **Train to standard, not to exposure.** Training that shows users what the dashboard looks like is not training. Training that requires users to answer three mission-relevant questions using the dashboard — and grade them on speed and accuracy — builds actual capability. Design training tasks that mirror real operational use cases.
6. **Measure adoption, not deployment.** Deployment is when you make the capability available. Adoption is when users incorporate it into their workflow. Measure: weekly active users, query volume, share of products citing MSS data. If deployment happened and adoption is not following, diagnose the gap — it is either a usability problem, a data quality problem, or a command climate problem.
7. **Sustain the change.** Schedule a 30-, 60-, and 90-day check-in with the pilot unit commander or data champion. Confirm the workflow change has held. Identify any regression toward old methods and address it directly. Rotation of personnel is the most common cause of adoption regression — ensure your onboarding documentation captures the workflow so new arrivals do not revert to old patterns.

CAUTION: Do not interpret unit resistance to MSS adoption as evidence that the product is wrong. Resistance to change is normal and expected in high-tempo military environments. Distinguish between substantive feedback (the dashboard doesn't show what I need) and adoption resistance (I don't have time to learn a new tool). The first requires product changes; the second requires change management.

4-3. Navigating Command Climate

BLUF: In a military organization, data tells truths that some leaders do not want to hear. The program manager must be able to deliver accurate information — including unfavorable information — without losing the trust of the command.

This is not a technical skill. It is a leadership skill. Techniques:

Separate the data from the interpretation. Present what the data shows before offering interpretation. "The readiness dashboard shows that 23% of organic vehicles across theater are non-mission-capable. This is an increase from 19% last month." Then, after a pause: "One factor that may explain this trend is..." This prevents the audience from conflating the data with your analysis — and prevents your analysis from being dismissed as bias.

Cite the source, surface the limitation. If the data has a known quality issue, say so explicitly. "This figure draws from GCSS-A feeds; units with delayed report submission may be underrepresented." This builds credibility — leaders trust data providers who are honest about limitations more than those who present everything as perfect.

Anticipate the challenge. Before briefing unfavorable data to a GO, ask yourself: "What is the most likely pushback?" Prepare a one-paragraph response. If the data shows a unit performing poorly, the command will likely challenge the methodology, the comparison baseline, or the data source. Know your answers before you walk into the room.

Document the data, not the politics. After a briefing where data was challenged or suppressed by leadership direction, document the facts in writing — what data was presented, what the response was, and what decision was made. This protects the program manager and creates an accurate record if the issue surfaces later.

4-3A. Vignette: Managing a Commander Who Wants the Wrong Dashboard

Situation: The USAREUR-AF Deputy Commanding General (DCG) has requested a new MSS dashboard that shows "real-time unit locations and movement" for all theater formations. The request comes through the G6 as a direct tasking to the MSS program manager. The program manager recognizes immediately that: (1) real-time unit location data at the resolution requested is operationally sensitive and outside the approved data feeds in the current MSS data governance framework; (2) the requirement as stated would require six months of development and a new external data integration; and (3) what the DCG probably wants — theater force posture visibility for readiness planning — can likely be addressed with a two-week enhancement to an existing dashboard.

The wrong response: Start building toward the stated requirement immediately. Accept the tasking verbatim, begin scoping the six-month effort, and present a timeline to the G6.

The right response:

1. Acknowledge the requirement and request a 15-minute clarification meeting with the G6 or DCG's action officer. "I want to make sure I'm building exactly what the DCG needs. Can I get 15 minutes to walk through the use case before we scope the effort?"
2. In the clarification meeting: ask "What decision is the DCG making with this dashboard? What would they look at it to answer?" Listen carefully. Document the underlying operational need.

3. If the underlying need is force posture visibility for readiness planning: present the existing dashboard, what it currently shows, and what a two-week enhancement would add. Ask: "If we could show this data in two weeks versus six months for the fuller capability, which better meets your timeline?"
4. If the underlying need genuinely requires the full capability: document the requirement clearly, brief the data governance implications to C2DAO, and present a realistic timeline. Do not promise a shorter timeline to please the DCG.
5. Document the outcome of the clarification and send a written confirmation to the G6: "Per our discussion, the DCG's primary need is [X]. We will deliver [Y] by [date]. The more extensive capability [Z] is planned for PI [N]."

Lesson: Requirement clarification is not pushback — it is professional competence. A program manager who builds what the commander asks for without confirming the underlying need will eventually deliver the right capability too late, or the wrong capability on time.

4-4. Stakeholder Mapping at Program Level

At program level, stakeholder management requires a formal map — not informal relationship knowledge held in the PM's head.

Table 4-1. Program Stakeholder Register Template

Stakeholder	Role/Grade	Unit	Influence	Interest	Engagement Strategy	Engagement Frequency
USAREUR-AF G6	O-8 equivalent	HQ USAREUR-AF	Decision authority	High (platform owner)	Bi-monthly program review brief	Bi-monthly
C2DAO Chief	O-6/GS-15	USAREUR-AF	Technical authority	High	Weekly working-level sync	Weekly
EUCOM J6	O-8 equivalent	USAREUR-AF	Resource influence	Medium	Quarterly capability update	Quarterly
G2/G3 Data Leads	O-4/O-5 or GS-13	HQ USAREUR-AF	Requirements authority	High	Sprint review invitees; monthly requirements sync	Monthly
Palantir Delivery Lead	Contractor	Task Order	Delivery accountability	High	Weekly task order status meeting	Weekly

Stakeholder	Role/Grade	Unit	Influence	Interest	Engagement Strategy	Engagement Frequency
Unit S6/Data Leads	O-3/CW2-CW4	Division/Brigade	Adoption influence	Medium	Quarterly field engagement	Quarterly

CHAPTER 5 — QUANTITATIVE DELIVERY MANAGEMENT

5-1. Using Data to Manage the Data Team

BLUF: A data program that does not measure its own delivery performance is practicing intuition management. Use the same analytical discipline on your program that your ORSA team applies to operational problems.

SL 4J introduced delivery planning at the project level — estimating story points, running retrospectives, and adjusting sprint capacity based on velocity. SL 5J extends this to the program level: aggregating metrics across multiple teams, identifying systemic patterns (not just individual sprint anomalies), and using quantitative data to make investment and prioritization decisions.

The goal is not to micromanage teams through metrics. The goal is to understand the delivery system well enough to predict its behavior — and to change the system when predictions are consistently wrong.

5-2. Core Delivery Metrics

Table 5-1. Program-Level Delivery Metrics

Metric	Definition	Measurement	Use
Velocity	Story points delivered per sprint, per team	Collected at sprint review	Capacity planning; team health indicator
Velocity Trend	Direction and rate of velocity change over 6+ sprints	Calculated from sprint history	Early warning of team degradation or improvement
Throughput	Features (not points) delivered per PI per team	Counted at PI Review	Cross-team comparison; planning denominator
Cycle Time	Elapsed time from feature start to feature delivery	Measured in the tracker	Process efficiency; identifies bottlenecks
Predictability	Ratio of PI committed features to PI delivered features	Calculated at PI Review	Commitment reliability; planning accuracy

Metric	Definition	Measurement	Use
Escaped Defects	Defects found in production that were not caught pre-release	Tracked in defect log	Quality trend; test coverage adequacy
Blocked Time	Fraction of sprint capacity consumed by blockages	Reported at sprint review	Dependency management effectiveness
Team Stability	Average tenure of team members over rolling 6 months	Calculated from roster	Context for velocity interpretation

TASK 5-2A: ESTABLISH PROGRAM METRICS BASELINE

CONDITIONS: At least one PI of delivery history exists. Sprint review data has been collected for all active teams.

STANDARDS: Baseline is established when: all eight metrics in Table 5-1 are calculated for the past two PIs; data is stored in the program tracker; a trend chart is available for velocity, cycle time, and predictability.

EQUIPMENT: MSS program tracker (sprint review records, feature completion data), MSS Workshop (charting and visualization), program ontology (metrics storage).

PROCEDURE:

1. Collect sprint review data for all teams for the past two PIs. If sprint data was not recorded, reconstruct from feature completion dates in the tracker.
2. Calculate each metric per team and for the program aggregate. Store results in the program tracker ontology.
3. For velocity, cycle time, and predictability, generate control charts or run charts showing the distribution over time. Mark the mean and acceptable variation bands (typically ±1 standard deviation).
4. Identify statistical anomalies: sprints or teams whose metrics fall outside normal variation. Flag these for qualitative investigation — a statistical anomaly is a question, not an answer.
5. Document the baseline in the program tracker. All future PI metrics will be measured against this baseline to show trend.

5-3. Interpreting Metrics: What They Tell You and What They Don't

Delivery metrics are correlational, not causal. A drop in velocity tells you a team delivered fewer points — it does not tell you why. Before acting on a metric signal, investigate the cause.

Table 5-2. Metric Signals and Investigation Questions

Metric Signal	Possible Causes	Investigation Questions
Sustained velocity drop (3+ sprints)	Team instability, scope creep, technical debt accumulation, morale issues, undocumented dependencies	Who left or joined the team? What changed in the sprint composition? Is the backlog growing faster than delivery?
High cycle time	Excessive work in progress, unclear acceptance criteria, blocked dependencies, insufficient code review bandwidth	What features are sitting in a "in progress" state for more than one sprint? What is the WIP limit?
Low predictability (<75%)	Inaccurate estimation, late scope changes, dependency surprises, overly ambitious PI commitment	Are scope changes coming from the team or the stakeholder? Are committed features the same features that were delivered?
High escaped defects	Insufficient testing, rushed release gates, unclear Definition of Done	Was the Definition of Done checklist completed for each release? What is the test coverage level?
High blocked time	Unresolved dependencies, waiting for approvals, unclear requirements	What category of blockage is most common? Is it internal or external?

WARNING: Do not use delivery metrics to evaluate individual personnel performance. Velocity is a team metric, not an individual metric. Using sprint velocity as a proxy for individual productivity is a well-documented antipattern that destroys team trust and incentivizes gaming the metric. SL 5J program managers who use metrics punitively will degrade the teams they are supposed to enable.

5-3A. Vignette: Reading a Velocity Drop Correctly

Situation: Team Bravo's velocity has dropped from an average of 42 points per sprint to 24 points per sprint over the past three sprints. The program manager notices this in the weekly metrics review and pulls the team lead aside.

The wrong diagnosis: "Team Bravo is underperforming. They need to work harder and commit to more points."

The correct diagnostic process:

1. **Ask, don't tell.** The program manager asks Team Bravo's SL 4J PM: "I'm seeing a velocity drop over the last three sprints. Walk me through what's been different."
2. **Look at what changed.** The PM reports: Sprint N-2 was normal; Sprint N-1 began after the senior ML engineer (SL 4M) rotated out; Sprint N began after a major Foundry platform update that required the team to adapt three of their pipelines mid-sprint.

3. **Interpret the data in context.** The velocity drop is explained by two compounding factors: loss of a senior technical contributor mid-PI, and an unplanned platform migration burden. Neither is evidence of underperformance — both are evidence of a program-level risk that was not adequately mitigated.
4. **Correct program-level, not team-level.** The corrective actions are: (a) initiate a knowledge transfer review for the rotated ML engineer — have they transferred their model documentation? (b) add platform update impact to the dependency register as a standing item; (c) notify C2DAO that platform update timing should be coordinated with active sprint schedules.
5. **Adjust PI commitments if needed.** If the velocity reduction will persist into the next sprint, re-forecast Team Bravo's PI commitment. Do not hold the team to a commitment made under different capacity conditions.

Lesson: Velocity is a trailing indicator that reflects past conditions. The program manager's job is to understand the conditions, not react to the number.

5-4. Program-Level Predictability Targets

Table 5-3. Program Predictability Benchmarks

Predictability Level	Interpretation	Program Manager Action
>85%	High predictability; planning process is working	Maintain current planning rigor; explore capacity for stretch objectives
70-85%	Acceptable; normal variation and scope change	Review PI commitment process; ensure change control is working
55-70%	Low; systemic estimation or scope management issue	Conduct root cause analysis; adjust estimation techniques; tighten change control
<55%	Very low; planning process is broken	Pause new commitments; conduct PI Planning retrospective; consider team restructure

5-5. Communicating Metrics to Leadership

BLUF: Metrics presented without narrative mislead senior audiences. Always pair a metric with an interpretation and, if relevant, an action.

When briefing delivery metrics to GO/SES leadership:

- Report trend, not snapshot. "Team Alpha's velocity has increased 22% over three PIs" is more meaningful than "Team Alpha's velocity this sprint was 34 points."
- Quantify predictability. "This PI, the program delivered 87% of committed features" is the single most important delivery metric for a senior audience.

- Flag anomalies with mitigations. Do not present a metric anomaly without also presenting the investigation result and corrective action.
- Do not brief raw story points to non-technical audiences. Translate to capability language: "This PI we delivered 11 new features, including the theater-wide readiness dashboard and the updated force tracking pipeline."

CHAPTER 6 — TECHNICAL DEBT STRATEGY AND BUILD/BUY/CONFIGURE DECISIONS

6-1. Technical Debt at Program Scale

BLUF: Technical debt is the accumulated cost of earlier shortcuts. In a data capability program, unmanaged debt does not merely slow delivery — it accumulates interest, compounds across products, and eventually makes the portfolio brittle enough that a single pipeline failure can cascade into a theater-level data outage.

SL 4J introduced technical debt as a project-level concern — tracking debt items in the backlog and allocating sprint capacity to address them. SL 5J raises the stakes: you are now managing debt across a portfolio of 5-10 MSS products simultaneously, some of which are years old and were built by teams that have since rotated out. Your job is to assess the aggregate debt level of the program, prioritize which debt to address first, and communicate the risk to leadership in operational terms.

6-2. Debt Classification Framework

Not all technical debt is equal. Classify debt before prioritizing it.

Table 6-1. Technical Debt Classification

Category	Description	Examples in MSS Context	Risk if Unaddressed
Architectural debt	Foundational design decisions that now constrain all downstream development	Pipeline architecture built for one dataset that now must support ten; ontology schema that cannot accommodate new object types	Exponential cost to change; limits all future capability growth
Data quality debt	Known data issues that have been accepted rather than fixed	Missing field validation, inconsistent unit normalization, unresolved duplicate records	Inaccurate dashboards; analytical models trained on dirty data
Documentation debt	Insufficient documentation of pipelines, transforms,	Undocumented pipeline steps; no data dictionary for an ontology; no	Knowledge loss on personnel rotation;

Category	Description	Examples in MSS Context	Risk if Unaddressed
	and business logic	runbook for a production alert	unmaintainable code
Test coverage debt	Insufficient automated testing for production pipelines and models	No CI checks on transform logic; no data quality assertions; no regression tests for ML model outputs	Silent failures in production; undetected data corruption
Dependency debt	Reliance on deprecated platform features or unsupported library versions	Pipelines using deprecated Foundry APIs; Python transforms relying on pinned library versions no longer supported	Breaking failures during platform upgrades
Configuration debt	Hardcoded values, duplicated logic, inconsistent naming	Magic numbers in transform code; duplicated business logic across three pipelines; inconsistent dataset naming	Costly maintenance; errors when values change

TASK 6-2A: CONDUCT A PORTFOLIO-LEVEL DEBT ASSESSMENT

CONDITIONS: The program has at least three active MSS products in production. Team leads have been briefed on the debt classification framework.

STANDARDS: Assessment is complete when: all active products have been assessed against the debt classification framework; each identified debt item is categorized, sized (story points or days), and assigned a risk rating; the aggregate debt backlog is documented in the program tracker; a prioritized debt reduction roadmap covers the next two PIs.

EQUIPMENT: MSS program tracker (product inventory, backlog), debt classification framework (Table 6-1), access to product documentation for each active MSS product, technical leads (SL 4G through SL 4O) for each product.

PROCEDURE:

1. **Assemble the assessment team.** The debt assessment requires input from the technical practitioners who built and maintain each product. Schedule a half-day workshop with SL 4G through SL 4O leads for each active product. The PM does not lead the technical assessment — they facilitate it and document the outputs.
2. **Assess each product against the six debt categories.** For each category, rate the severity: None, Low, Medium, High. For any Medium or High rating, document the specific debt item, the estimated remediation cost, and the operational risk if the debt is not addressed.
3. **Calculate aggregate debt per product.** A product with four High debt items is a program risk. Flag any product with two or more High ratings as a debt priority candidate.

4. **Prioritize by risk × cost efficiency.** Rank debt items by: (operational risk of the debt) × (cost efficiency of remediation). High-risk, low-cost debt items are the first priority. High-risk, high-cost items require a business case and leadership visibility. Low-risk debt items are addressed opportunistically during sprint slack time.
5. **Build the debt reduction roadmap.** Allocate 20% of program capacity in each PI explicitly to debt reduction. This is not optional. Teams that do not allocate debt reduction capacity will deliver faster in the near term and break catastrophically in the long term. Identify which debt items will be addressed in each PI.
6. **Communicate debt risk to leadership.** For any debt item that poses production outage risk or analytical accuracy risk, brief C2DAO leadership. Frame in operational terms: "The readiness pipeline has undocumented normalization logic that will break if the G4 GCSS-A schema changes. This is a medium-probability event that would produce incorrect NMC rates in the theater readiness dashboard. We are allocating four points in the next sprint to document and validate this logic."

CAUTION: Debt assessment workshops sometimes surface problems that team leads have been hiding — either out of embarrassment or because they feared the PM would use the information to pressure them. Create a safe environment for honest reporting. The goal is to quantify and address the debt, not to assign blame.

6-2A. Communicating Technical Debt to Non-Technical Leaders

Technical debt is an abstract concept for most military leaders. Telling a GO that "we have architectural debt in our pipeline layer" is the equivalent of telling them "we have some software complexity issues." It is not actionable. The program manager's job is to translate debt risk into operational terms that drive appropriate investment decisions.

Table 6-3. Technical Debt Communication Translation Guide

Technical Debt Description	Operational Translation
"Undocumented business logic in the readiness pipeline"	"If the two engineers who built this pipeline rotate out, the next team will spend 6-8 weeks reverse-engineering the logic before they can make changes. Any bug during that period will take 3x longer to fix."
"Deprecated Foundry API in 5 production pipelines"	"Palantir will stop supporting the API we use in [month]. After that date, any platform upgrade may break 5 of our production dashboards simultaneously. We need 3 sprint-weeks to migrate before that deadline."
"No automated data quality checks on the G4 feed"	"If the G4 changes their data format — which has happened twice in the past year — our supply chain dashboard will silently display wrong data until a user catches the error manually. We have no automated detection."
"Ontology schema fragmentation across	"Team Alpha and Team Bravo both have 'Unit' objects in their ontologies, but they define them differently. A commander looking at both dashboards simultaneously

Technical Debt Description	Operational Translation
product teams"	sees inconsistent readiness numbers for the same unit. This is a data trust problem."
"Single-threaded pipeline with no retry logic"	"If the GCSS-A feed fails for any reason during the nightly pipeline run, the readiness dashboard does not update. This has happened three times in six months. Users have started calling the helpdesk at 0600 when the dashboard is stale."

NOTE: When briefing debt risk to senior leaders, always pair the risk with the remediation cost and timeline. "This debt item costs 4 story points to fix and prevents a high-probability production outage" is a complete brief. "We have technical debt" is not.

6-3. Build vs. Buy vs. Configure

BLUF: Every capability request can be met in three ways: write custom code, configure existing MSS platform features, or procure a new capability. The correct answer depends on context — and the wrong answer is costly.

This decision recurs constantly at the program level. A new stakeholder requirement arrives. The team asks: "Do we build this in Python, or can Workshop handle it?" The wrong answer can mean 60 days of development work for something a Workshop widget could have delivered in a day — or, equally, a fragile, undocumented configuration that breaks the next time Palantir upgrades the platform.

Table 6-2. Build vs. Buy vs. Configure Decision Framework

Factor	Build (Custom Code)	Configure (Platform Feature)	Buy/Procure (New Capability)
Use when	Requirement is unique, complex, and not addressed by platform; will be maintained long-term by the team	Requirement is well-addressed by a standard platform feature; no custom logic needed	Capability is well-established commercially or through Army programs; building is not a core competency
Advantages	Full control; optimized for requirement; no dependency on platform vendor decisions	Fast to deploy; maintained by Palantir; platform-supported; low technical debt	Leverages proven capability; reduces team capacity burden
Risks	Highest technical debt risk; requires ongoing maintenance; knowledge loss on rotation	May not exactly fit requirement; can break during platform upgrades; limited auditability of platform logic	Procurement time; vendor lock-in; may not integrate cleanly with MSS
MSS exampl	Custom Python transforms for complex business logic;	Workshop dashboards for standard reporting; Pipeline	Palantir AIP for AI-assisted workflows; Army-wide data

Factor	Build (Custom Code)	Configure (Platform Feature)	Buy/Procure (New Capability)
es	bespoke ML model architectures; API integrations with non-standard data feeds	Builder for ETL on standard formats; Ontology object types for standard data models	tools procured through Army CIO
Decision gate	Can the platform feature meet 80% of the requirement? If yes, configure first.	Is the custom logic reusable across multiple products? If yes, building may justify the cost.	Is this capability already available through an Army-managed procurement? Check with C2DAO before building.

NOTE: The most common mistake at program scale is building what can be configured. When a requirement arrives, the SL 5J program manager's first question is always: "What can MSS do out of the box that addresses this?" Only after a clear negative answer — documented and validated by the technical leads — should a build decision proceed.

TASK 6-3A: CONDUCT A BUILD/BUY/CONFIGURE DECISION REVIEW

CONDITIONS: A new stakeholder requirement has been received that will require significant development investment (estimated >40 story points or >3 weeks of team capacity).

STANDARDS: Decision is documented when: all three options have been evaluated against the framework in Table 6-2; a recommendation is made with supporting rationale; the decision is recorded in the program backlog and communicated to the requesting stakeholder with an expected delivery timeline.

EQUIPMENT: Build/Buy/Configure decision framework (Table 6-2), MSS program backlog in program tracker, access to relevant technical leads (SL 4H, SL 4M, SL 4K, SL 4L), C2DAO program inventory for existing procurement options.

PROCEDURE:

1. Brief the requirement to the relevant technical leads (SL 4H, SL 4M, SL 4K, SL 4L as appropriate). Ask each to assess: can this requirement be met by an existing platform feature? Rate: Yes (full fit), Partial (80% fit with acceptable gaps), No (platform feature inadequate).
2. If consensus is "Yes" or "Partial": proceed with configure approach. Document the gap between the platform feature and the requirement. If the gap is acceptable to the stakeholder, close the analysis. If the gap is not acceptable, document why the platform feature is insufficient.
3. If consensus is "No": assess the build option. Estimate development cost (story points, team capacity, PI allocation). Assess maintenance burden: who maintains this after the PI team rotates? Is the logic reusable across other products?
4. Simultaneously, assess the buy option. Query C2DAO: is there an Army-managed program or existing Palantir capability that addresses this requirement? Check with the Palantir delivery lead: does the task order include capability that covers this?

5. Document the decision: selected approach, rationale, cost estimate, delivery timeline, and maintenance owner. Circulate to C2DAO and the requesting stakeholder for concurrence.
6. Log the decision in the program tracker as a program decision record. Over time, this log becomes a reference for future similar decisions and supports program retrospectives.

CHAPTER 7 — VENDOR AND CONTRACTOR MANAGEMENT

7-1. The Palantir Partnership Model

NOTE — Palantir Developers reference: *Code in Production: Process Orchestration x Eaton | DevCon 4* — Documents a large-scale process automation deployment through Palantir's platform, covering the enterprise delivery model and partnership structure that underpins production-scale rollouts. Reinforces the Palantir partnership and task order management content in this chapter. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

BLUF: Palantir operates as a strategic platform partner on MSS, not simply a software vendor. Managing this relationship well requires understanding what Palantir is accountable for, what the government is accountable for, and where those boundaries create gaps.

The MSS operating model combines government staff (military and Civilian), Palantir forward-deployed engineers (FDEs) and deployment strategists (DSs), and other contractors operating under task orders. Understanding who does what — and who is accountable for what — is essential for the SL 5J program manager.

Table 7-1. MSS Delivery Roles

Role	Who	Accountable For	Not Accountable For
Palantir FDE	Palantir employee	Platform capabilities, technical architecture guidance, platform bug escalation	Operational requirements translation; government data quality; military workflow understanding
Palantir DS	Palantir employee	User adoption, change management, training delivery, use case development	Code delivery under government task orders; production pipeline maintenance
Government PM (SL 5J)	Military/Civilian	Program delivery, task order acceptance, requirements validation, team health	Platform architecture decisions made by Palantir; platform bugs or outages

Role	Who	Accountable For	Not Accountable For
Government tech leads (SL 4 series)	Military/Civilian/GS	Technical delivery of government-owned products	Palantir-delivered features on task order
Task order contractors	Contractor (non-Palantir)	Deliverables specified in the task order; testing and documentation per acceptance criteria	Scope outside the task order; Palantir platform decisions

7-2. Task Order Management

A Palantir task order defines a set of deliverables, a schedule, and a price. The government PM is responsible for: verifying deliverables meet acceptance criteria, accepting or rejecting work, and documenting performance.

TASK 7-2A: ESTABLISH TASK ORDER ACCEPTANCE CRITERIA

CONDITIONS: A new task order or task order modification has been executed. Deliverables are listed in the performance work statement (PWS).

STANDARDS: Acceptance criteria are established when: each deliverable in the PWS has written acceptance criteria that are specific, measurable, and testable; criteria have been reviewed by the relevant SL 4 technical lead; criteria are shared with the Palantir delivery lead before work begins.

EQUIPMENT: Performance Work Statement (PWS) for the task order, MSS program tracker (task order deliverable object), relevant SL 4 technical leads, UDRA v1.1 data governance standards, MSS naming and governance standards.

PROCEDURE:

1. Read the PWS deliverable descriptions. For each deliverable, identify: (a) what the deliverable is (document, pipeline, dashboard, trained model, API), (b) what "done" means (specific functional requirements, performance thresholds, data format specifications), and (c) what quality standards apply (documentation coverage, test coverage, adherence to MSS naming conventions and data governance standards per UDRA v1.1).
2. Draft acceptance criteria in plain language. Use the format: "The deliverable is accepted when [specific observable condition is met]." Avoid vague language: "The pipeline runs correctly" is not an acceptance criterion. "The pipeline processes 100% of records from the G4 GCSS-A feed within 30 minutes of feed receipt, with zero records lost and all output fields populated according to the data dictionary in Appendix [X]" is an acceptance criterion.
3. Route draft criteria to the relevant SL 4 technical leads for technical validation. The PM writes criteria in operational terms; the technical lead validates that the criteria are technically sound and testable.

4. Share finalized criteria with the Palantir delivery lead before work begins. Confirm mutual understanding. Any disagreement about what the criteria mean must be resolved before the sprint starts — not during acceptance review.
5. Document the finalized criteria in the program tracker as a linked attribute of the relevant task order deliverable object.

WARNING: Accepting a task order deliverable without documented acceptance criteria creates a permanent contractual record that the deliverable met the government's requirements — regardless of its actual quality. Once accepted, returning a deliverable for rework requires a contract modification and is significantly more costly than rejecting it correctly the first time. Never accept a deliverable based on verbal assurance that "it works."

TASK 7-2B: CONDUCT DELIVERABLE ACCEPTANCE REVIEW

CONDITIONS: Palantir or contractor has submitted a task order deliverable for government acceptance. Acceptance criteria from Task 7-2A are documented.

STANDARDS: Acceptance review is complete when: all acceptance criteria have been tested; findings are documented; the deliverable is either formally accepted (with documented rationale) or rejected (with specific deficiencies listed and a remediation timeline agreed).

EQUIPMENT: Documented acceptance criteria (from Task 7-2A), the submitted deliverable (access to Foundry workspace, document, or system as applicable), relevant SL 4 technical lead for evaluation, MSS program tracker (acceptance event documentation).

PROCEDURE:

1. **Notify the technical review team.** Assign the relevant SL 4 technical lead as the technical evaluator for the deliverable. Provide the deliverable and the acceptance criteria simultaneously. Establish a review timeline — typically 5 business days for a standard deliverable, 10 for a complex pipeline or model.
2. **Execute acceptance testing.** The technical evaluator tests each acceptance criterion. For a pipeline deliverable: run the pipeline against the specified test data and verify output fields. For a dashboard deliverable: verify each required widget displays the correct data from the correct source with the correct business logic applied. For a documentation deliverable: verify completeness against the required sections and accuracy against the actual system.
3. **Document findings.** For each criterion: Pass (criterion is fully met), Conditional Pass (criterion is mostly met, minor deficiency noted — deficiency is documented but does not block acceptance), or Fail (criterion is not met — specific failure documented).
4. **Conduct the acceptance review meeting.** Brief findings to the Palantir delivery lead. For any Fail, present the specific deficiency with evidence. Do not negotiate on failures — the criterion either is or is not met.

5. **Issue formal acceptance or rejection.** If all criteria Pass or Conditional Pass (with minor deficiencies documented): issue formal acceptance in writing. If any criterion Fails: issue a written rejection with specific deficiencies and required remediation actions. Establish a remediation timeline in writing.
6. **Document performance.** Record the acceptance outcome, date, and any deficiencies in the program tracker. This documentation supports past performance assessments and future task order negotiations.

7-3. Managing the Palantir Relationship Day-to-Day

The Palantir FDE is a technical resource, not a government employee. Their priorities are influenced by their employer's business interests, their utilization targets, and their personal technical preferences. Managing this productively requires clarity about roles and assertive, professional communication when boundaries are crossed.

Table 7-2. Common Palantir Relationship Friction Points and Responses

Friction Point	Description	PM Response
Scope creep from FDE	Palantir FDE begins building features beyond the task order scope without direction	Formally acknowledge the work; document in the program tracker; confirm whether it will be accepted as in-scope or out-of-scope for the task order. Do not allow informal scope expansion.
Platform advocacy	FDE advocates strongly for a platform feature solution when the requirement warrants a custom approach	Conduct the Build/Buy/Configure review (Task 6-3A) with government technical leads present. Decision authority rests with the government.
Deliverable quality disputes	Palantir disputes a government rejection of a deliverable	Reference the documented acceptance criteria established before work began. Escalate to contracting officer if criteria dispute cannot be resolved at the PM level.
FDE turnover	Palantir replaces an FDE mid-task order with a less experienced resource	Document the impact. If delivery is at risk, formally notify the contracting officer and request Palantir mitigation plan.
Platform outage attribution	A production outage is attributed to the platform; Palantir attributes it to government configuration	Document the technical facts. Engage C2DAO to determine root cause. Do not publicly attribute blame before root cause is confirmed.

7-3A. Working with Palantir FDEs: A Practical Guide

Palantir Forward Deployed Engineers are typically technically excellent and highly motivated. They are also operating under pressures and incentives that differ from the government's. Understanding these dynamics makes the relationship more productive.

FDE incentives to understand:

- Palantir FDEs are incentivized to demonstrate platform capability. They will often propose solutions that use advanced Foundry features — AIP, complex ontology architectures, custom applications — even when simpler approaches would meet the requirement. This is not malicious; it reflects their training and their instinct to showcase the platform. The PM must evaluate FDE proposals against the build/buy/configure framework (Chapter 6), not accept them uncritically.
- Palantir FDEs rotate. The FDE assigned to USAREUR-AF today may not be the FDE assigned six months from now. Do not allow critical program knowledge to reside only in the FDE's head. Require documentation of all FDE contributions in government-accessible Foundry workspaces.
- Palantir FDEs are not government employees and are not subject to government task management. You cannot assign work to an FDE outside of what is defined in the task order or a documented advisory relationship. If you need a specific FDE to perform specific work, ensure it is covered by the task order scope.

Effective FDE engagement practices:

- Hold a weekly standing meeting with the Palantir delivery lead and the primary FDE. Agenda: task order delivery status, upcoming dependencies, platform updates affecting production, and any technical questions the government team needs input on.
- Invite the FDE to sprint reviews as an observer, not a participant. They should see what the government team is building but not have a vote in sprint planning.
- When the FDE makes an architectural recommendation, require a brief document: what they are recommending, why, and what alternatives they considered. This creates accountability for the recommendation and ensures the government team has enough information to evaluate it.
- When disagreements arise between the government technical lead and the FDE, the government position prevails. The FDE is an advisor; the government owns the product.

7-4. Vendor Performance Documentation

The SL 5J program manager is responsible for maintaining performance documentation that informs future task order awards and modifications.

Table 7-3. Vendor Performance Documentation Requirements

Event	Documentation Required	Timeline
Task order deliverable accepted	Formal acceptance memo; deficiencies noted	Within 5 business days of acceptance
Task order deliverable rejected	Rejection memo with specific deficiencies; remediation plan	Within 5 business days of rejection
Deliverable accepted with significant rework	Supplemental performance note documenting initial deficiencies and remediation quality	At final acceptance
Task order completed	Summary performance assessment: schedule performance, quality performance, responsiveness, overall rating	Within 30 days of task order completion
Significant vendor performance issue	Incident memo documenting the issue, impact, and resolution	Within 10 business days of resolution

CHAPTER 8 — TEAM DESIGN AND PM LEADERSHIP

8-1. Designing the Data Program Team Structure

BLUF: There is no universal team structure for a data capability program. The right structure depends on program size, portfolio composition, available personnel, and operational tempo. What does not vary is the principle: structure should serve the work, not the org chart.

At program scale, the SL 5J manager makes structural decisions that SL 4J PMs cannot: how many teams, how large, what skill mix, what reporting relationships, how to handle shared services (data engineering, ontology governance, security). These decisions have long-term consequences — restructuring mid-program is costly and disruptive.

8-2. Team Topology Options

Table 8-1. Team Topology Options for MSS Programs

Topology	Description	When to Use	Risk
Feature Teams	Each team is cross-functional (ORSA, SWE, KM, PM) and owns a specific product or product area end-to-end	Programs with distinct product domains; stakeholders want a single team to call	Duplicates skill sets; cross-product consistency requires explicit coordination

Topology	Description	When to Use	Risk
Component Teams	Teams are organized by technical layer (data engineering team, analytics team, UI team)	Programs where reuse across products is high and technology layers are well-defined	Creates dependencies between teams; feature delivery requires coordination across components
Stream-Aligned Teams	Teams are aligned to a user workflow or operational stream (readiness stream, logistics stream, intelligence stream)	Programs where operational stakeholders are the organizing principle	Requires clear stream ownership; hard to balance streams of unequal scope
Platform Team + Feature Teams	A shared platform/infrastructure team supports multiple product teams building on top	Programs with significant shared infrastructure (ontology, data feeds, pipelines)	Platform team becomes a bottleneck if not appropriately resourced

NOTE: For most USAREUR-AF MSS programs at current scale, the recommended structure is Platform Team + Stream-Aligned Feature Teams. The platform team (typically SL 4K KM and SL 4L SWE) owns ontology governance, shared pipelines, and data quality standards. Stream-aligned feature teams own product delivery within their operational domain.

8-3. Skill Mix Planning

Table 8-2. Recommended Skill Mix per MSS Feature Team

Role	Track	Count per Team	Notes
Technical PM	SL 4J	1	Required; team lead for delivery
ORSA	SL 4G	1-2	Required if team is producing analytical products
ML Engineer	SL 4M	1	Required if team is building or maintaining ML models
AI Engineer	SL 4H	0-1	Required if team is deploying production AI pipelines or AIP integrations
Software Engineer	SL 4L	1-2	Required for custom transforms; optional if team is primarily configure-mode
Knowledge Manager	SL 4K	0-1	Required if team owns ontology objects; may be shared from platform team

Table 8-3. Platform Team Composition

Role	Track	Count	Notes
Platform Lead	SL 5L or SL 5K	1	Oversees ontology, shared pipelines, governance
Knowledge Manager (Senior)	SL 5K	1-2	Owns ontology governance; data standards
Software Engineer (Senior)	SL 5L	1-2	Owns shared pipeline infrastructure; code standards
Data Quality Lead	SL 4K or SL 4G	1	Owns data quality standards and monitoring

8-4. Personnel Turbulence Planning

Military organizations rotate personnel. A program that is not designed to absorb turnover will degrade in capability each rotation cycle — losing institutional knowledge, breaking established processes, and re-fighting resolved architectural debates with each new cohort.

TASK 8-4A: ESTABLISH PROGRAM KNOWLEDGE CONTINUITY STANDARDS

CONDITIONS: Program has been operating for at least one PI with a stable team structure. Rotation of military personnel is anticipated within the next 6-12 months.

STANDARDS: Continuity is established when: all critical knowledge (pipelines, business logic, ontology decisions, stakeholder relationships) is documented; an onboarding package exists for each role; a knowledge transfer protocol is defined for departing personnel.

EQUIPMENT: MSS program tracker (program decision record, documentation workspace), MSS Foundry (product documentation workspaces for all active products), role-specific onboarding template, current team roster.

PROCEDURE:

1. **Conduct a knowledge audit.** For each person on the program, ask: "If this person left tomorrow, what would we not be able to do?" Document the answer. Any capability that lives only in one person's head is a continuity risk.
2. **Build role-specific onboarding packages.** Each package includes: a role overview (what the person does, who they work with, what tools they use), links to key documentation, access to the relevant MSS workspaces, a list of recurring meetings and their purpose, and a reading list (relevant TMs, UDRA sections, key architecture documents).
3. **Document critical decisions.** Maintain a program decision record in the program tracker. Each record includes: the decision made, the options considered, the rationale, the date, and the approving authority. This record is invaluable for new arrivals who encounter existing systems and ask "why is it done this way?"

4. **Implement a structured handover protocol.** Require a minimum 30-day overlap for any critical role transition (team lead, senior ORSA, platform lead). During overlap: conduct joint meetings, transfer relationship contacts, review all in-progress work, and conduct a formal knowledge handover checklist review.
5. **Cross-train within teams.** No critical capability should be held by one person. For every pipeline, model, or dashboard that is in production, require that at least two team members can support it. Build cross-training into sprint capacity allocation.

NOTE: Personnel turbulence is the largest long-term risk to a military data program. The technology does not forget — but the people do. A program manager who does not actively manage knowledge continuity will rebuild the same capabilities repeatedly as each generation of personnel rediscovers what the previous generation already knew.

8-5. PM Leadership: Developing Junior PMs

BLUF: SL 5J program managers are responsible for the professional development of SL 4J PMs they supervise. This is a leadership function, not just a management function.

TASK 8-5A: CONDUCT A PM CAPABILITY ASSESSMENT

CONDITIONS: A SL 4J PM is under program manager supervision for at least one PI.

STANDARDS: Assessment is complete when: the PM's capability profile has been evaluated against SL 4J competencies; development priorities are identified; a 90-day development plan is documented and agreed.

EQUIPMENT: SL 4J competency framework (Table 8-4), the PM's sprint delivery record for the past PI (from program tracker), development plan template, program tracker (to document and store the development plan).

PROCEDURE:

1. **Evaluate against SL 4J competencies.** Review the PM's delivery record for the past PI. Assess: backlog management quality (are user stories well-written, estimated, and linked to stakeholder requirements?); ceremony effectiveness (do sprints start and end on time, are retrospectives producing actionable improvements?); stakeholder management (are requirements clear, is communication timely, are issues surfaced early?); risk management (were risks identified before they became blockers, or discovered at impact?); metrics literacy (does the PM understand and use delivery metrics?).
2. **Conduct a 1:1 assessment conversation.** Share your observations. Ask the PM to self-assess against the same competencies. Listen for gaps between self-assessment and your observation — discrepancy is itself informative.
3. **Identify 2-3 development priorities.** Avoid trying to improve everything at once. Select the competencies where improvement will have the greatest program impact.

4. **Build a development plan.** For each priority: identify the specific behavior to change, the learning activity (shadow a senior PM, read a specific chapter of this manual, facilitate a ceremony with feedback, attend a C2DAO stakeholder meeting with the program manager), and the success measure. Assign a timeline and a check-in date.
5. **Check in monthly.** Development plans not reviewed regularly are not executed. Schedule a monthly 30-minute developmental conversation. Review progress. Adjust the plan if priorities have shifted.

Table 8-4. SL 4J to SL 5J Development Milestones

Competency	SL 4J Standard	SL 5J Ready Standard
Backlog management	Manages own project backlog with clean user stories and estimates	Can review another PM's backlog and identify gaps; coaches user story quality
Stakeholder management	Manages working-level stakeholders; escalates to program manager when needed	Independently manages O-6/GS-14 equivalent stakeholders; identifies escalation triggers without prompting
Metrics literacy	Reports sprint velocity and burn-down; can interpret a simple trend	Calculates and interprets cycle time, predictability, and blocked time; identifies systemic patterns across sprints
Risk management	Identifies and tracks risks within own project	Identifies cross-project risks; understands how a risk in one project affects the program
Technical fluency	Understands the work at a task level; can write acceptance criteria for technical deliverables	Can conduct a technical debt assessment with a SL 4 lead; understands build/buy/configure tradeoffs
Change management	Executes a change management plan designed by the program manager	Designs a change management plan; identifies change resistance and tailors approach

8-5A. The PM Peer Review

One of the most effective and underused PM development tools is peer review: having one SL 4J PM assess another's sprint artifacts — the backlog, the user stories, the acceptance criteria, the retrospective outputs. Peer review does three things simultaneously: it catches quality issues before they affect delivery, it develops the reviewer's critical thinking, and it reduces the isolation that PMs often feel managing their own project in a silo.

Table 8-6. PM Peer Review Rotation

Sprint	Reviewer	Reviewee	Focus Area
Sprint N	PM A	PM B	User story quality and acceptance criteria completeness
Sprint N+1	PM B	PM C	Risk register currency and risk mitigation quality

Sprint	Reviewer	Reviewee	Focus Area
Sprint N+2	PM C	PM D	Stakeholder communication artifacts
Sprint N+3	PM D	PM A	Retrospective outputs and action item follow-through

The program manager should facilitate the first two peer review cycles — structuring what to look for, modeling the feedback conversation, and debriefing the pair afterward. After two cycles, most PM pairs can conduct peer review independently.

NOTE: PM peer review is not a performance evaluation. It is a learning exercise. The program manager should reinforce this framing explicitly. If PMs feel they are being graded on their peer review artifacts, they will produce defensively written artifacts rather than honest ones.

8-6. Establishing Program PM Standards

SL 5J program managers set the standards that all SL 4J PMs on the program are expected to meet. These standards should be written down, communicated at program onboarding, and enforced consistently.

Table 8-5. Program PM Standards Template

Standard	Requirement
Sprint documentation	Sprint goal, committed features, and retrospective outputs documented in program tracker within 24 hours of ceremony completion
Risk reporting	All new risks entered in the risk register within 48 hours of identification; risk status updated at each sprint review
Stakeholder communication	All stakeholder commitments documented in writing within 24 hours of verbal agreement
Definition of Done	No feature marked complete without full Definition of Done checklist signed off (see SL 4J Appendix B)
Dependency flagging	All dependencies on other teams or contractors logged in the dependency register at PI Planning; status updated at each Program Sync
Metrics reporting	Velocity and predictability reported at each sprint review; no estimate revisions accepted after sprint start without program manager notification
Escalation	Any risk that will prevent PI commitment from being met is escalated to the program manager within the sprint it is identified — not at PI Review

CHAPTER 9 — PROGRAM GOVERNANCE AND POLICY COMPLIANCE

9-0. Why Governance is Not Bureaucracy

A common failure mode in Agile programs is to treat governance as the enemy of delivery — a layer of process and approval that slows teams down and exists primarily to satisfy auditors. This is wrong, and a SL 5J program manager who holds this view will build a program that is fast, fragile, and ungovernable.

Governance exists because programs at scale make interdependent decisions. Without explicit governance, every team makes local decisions that seem rational in isolation but are incoherent in aggregate. One team modifies the ontology for their product and breaks three dashboards owned by other teams. Another team accepts a Palantir deliverable without testing because the team lead trusts the FDE personally. A third team retires a dataset without notifying the downstream model that depends on it. None of these failures require malice — they require only the absence of governance.

Good governance is not a gate — it is a clarification of who decides what, communicated in advance, so that teams can move quickly within their authorized lane without creating risk for others.

9-1. Governance at Program Scale

BLUF: Program governance is the set of rules, processes, and decision authorities that determine how MSS capability is built, changed, and retired. Without explicit governance, the program drifts — decisions accumulate informally, standards diverge across teams, and the portfolio fragments into a collection of incompatible products.

SL 4J introduced project-level governance: Definition of Done, access management, sprint ceremonies, and stakeholder sign-off. SL 5J governance operates at the program level: who can approve changes to the ontology? Who authorizes a new production release? Who decides when to retire a product? These decisions affect multiple teams and multiple operational users simultaneously. They require documented authority and consistent process.

9-2. Program Decision Authority Matrix

The Decision Authority Matrix (DAM) defines who can make which decisions at the program level. The DAM prevents the two most common governance failures: decision paralysis (no one knows who can approve) and unauthorized action (someone acts without proper authority).

Table 9-1. Program Decision Authority Matrix

Decision	Program Manager	C2DAO Chief	GO/SES	Team Lead
Approve new PI objectives	Recommends	Approves	Informs	Inputs
Authorize production release (new product)	Recommends	Approves	Informs	Executes
Authorize production release (existing product, minor)	Approves	Informs	—	Executes
Approve ontology schema changes (major)	Recommends	Approves	—	Executes
Approve ontology schema changes (minor)	Approves	Informs	—	Executes
Authorize technical debt reduction sprint allocation	Approves	Informs	—	Executes
Authorize team restructure	Recommends	Approves	Informs	Inputs
Accept/reject task order deliverable	Approves	Informs	—	Technical review
Initiate product retirement	Recommends	Approves	Informs	Executes
Authorize external data integration (new source)	Recommends	Approves	—	Executes
Approve Foundry workspace access (new user)	Approves	Informs	—	Executes
Approve program budget reallocation (within task order)	Recommends	Approves	—	—

NOTE: The DAM is a living document. Update it when authority delegation changes, when the program scales to add new teams, or when a C2DAO policy change shifts approval authority. Publish the DAM in the program tracker and reference it at PI Planning.

9-3. Policy Compliance Framework

The MSS program operates under a layered policy framework. SL 5J program managers are responsible for understanding the requirements at each layer and ensuring program compliance.

Table 9-2. Applicable Policy Framework

Policy	Authority	Key Requirements for MSS Program
Army CIO Memo, Data and Analytics Policy (April 2024)	Army CIO	Data product categorization; AI system governance; data platform authorization; workforce data literacy standards

Policy	Authority	Key Requirements for MSS Program
UDRA v1.1 (February 2025)	Army CIO / DCS G6	Reference architecture for data management; ontology standards; pipeline governance; data product lifecycle management
USAREUR-AF C2DAO Data Governance Policy	C2DAO	MSS-specific access management; ontology change control; production release standards; data classification handling
Foundry Platform Security Requirements	C2DAO / ISSO	User account management; workspace permissions; audit logging; connection security for external data feeds
AI Governance (Army CIO, April 2024)	Army CIO	Requirement to document AI-assisted products; human review gates; model performance monitoring; bias assessment

TASK 9-3A: CONDUCT A QUARTERLY POLICY COMPLIANCE REVIEW

CONDITIONS: Program has been operating for at least one quarter. A list of all active MSS products and their technical characteristics (AI-assisted, external data feeds, data classification levels) is available.

STANDARDS: Review is complete when: all active products have been assessed against applicable policies; compliance gaps are documented with remediation plans; findings are reported to C2DAO leadership.

EQUIPMENT: MSS program tracker (product inventory), Army CIO Memo (April 2024), UDRA v1.1 (February 2025), C2DAO data governance policy, Foundry workspace access list (for access review), compliance status template.

PROCEDURE:

1. **Inventory all active MSS products.** For each product, document: the type (dashboard, pipeline, ML model, AI-assisted workflow), the data classification level of inputs and outputs, whether the product uses AI-assisted features (AIP, ML models), and whether it ingests data from external feeds.
2. **Assess against Army CIO Memo requirements.** For each AI-assisted product: is it documented in the MSS AI product registry? Are human review gates defined? Is model performance being monitored? Are model assumptions and limitations documented?
3. **Assess against UDRA v1.1 requirements.** For each active pipeline: does it comply with the data lineage documentation standard? Are datasets cataloged in the MSS data catalog? Are retention and disposal policies applied?
4. **Assess against C2DAO data governance policy.** For each product workspace: has access been reviewed in the last 90 days? Are all active users still requiring the access they have? Are any accounts for departed personnel still active?
5. **Document findings.** Produce a compliance status table: product name, policy area, status (compliant / gap / not applicable), gap description, remediation owner, remediation timeline.

6. **Report to C2DAO.** Brief findings to the C2DAO Chief at the next program review. For any Critical or High findings, schedule a dedicated briefing within 10 business days of discovering the gap.

9-4. Change Control for Production Systems

Changes to production MSS systems must follow a documented change control process. Unauthorized changes to production pipelines, ontology objects, or dashboard configurations are among the most common causes of data product failures.

Table 9-3. Change Control Classification

Change Type	Examples	Process	Minimum Lead Time
Emergency change	Production outage requiring immediate patch; security vulnerability	Verbal authorization from C2DAO; document within 24 hours post-change	Immediate; document retroactively
Standard change	Pre-approved, low-risk, recurring changes (e.g., adding a user to an access group)	Execute per standard operating procedure; log in change record	Defined in SOP
Minor change	Bug fix, display label correction, non-breaking configuration update	PM approval; notify downstream users; 24-hour notice	24 hours
Major change	New pipeline logic, ontology schema change, breaking interface change, new production model	PM recommends; C2DAO approves; downstream user notification; testing required	1 sprint (2 weeks minimum)
Breaking change	Schema change that requires downstream consumers to modify their code or configuration	PM recommends; C2DAO approves; all downstream teams confirm readiness; parallel period recommended	2 sprints (4 weeks minimum)

TASK 9-4A: EXECUTE A MAJOR CHANGE TO A PRODUCTION MSS PRODUCT

CONDITIONS: A major change to a production pipeline, ontology, or dashboard has been approved by C2DAO. Downstream consumers have been identified.

STANDARDS: Change is complete when: all downstream consumers have been notified and have confirmed readiness; change has been tested in a non-production environment; rollback procedure is documented; change has been executed within the approved maintenance window; change is logged in the program change record.

EQUIPMENT: MSS Foundry (development/staging environment, production environment), program tracker (change record), list of downstream consumer product owners, C2DAO approval documentation, rollback procedure template.

PROCEDURE:

1. **Create the change record.** Document: the change description, the reason, the affected systems, downstream consumers, test plan, rollback plan, and approval. Log in the program tracker.
2. **Notify downstream consumers.** Send written notification to all identified downstream product owners and operational users at least one sprint before the change execution window. Include: what is changing, when it will change, what action (if any) downstream consumers must take, and the rollback window if issues arise.
3. **Test in non-production environment.** Execute the change in a development or staging Foundry environment. Validate that: the change produces the expected behavior, downstream consumers' logic functions correctly against the new schema or output, and no data is corrupted.
4. **Confirm downstream readiness.** Within 48 hours of the change window, contact all downstream consumers. Confirm they have: reviewed the change notification, completed any configuration updates their product requires, and are ready for the change to execute.
5. **Execute in the approved maintenance window.** Apply the change to production during the agreed window. Execute the test cases from step 3 against production to confirm successful change.
6. **Monitor for 48 hours post-change.** Increased monitoring is required for two full business days after a major change. Assign a team member to monitor data quality indicators, pipeline execution logs, and user-reported issues during this window.
7. **Close the change record.** Document the execution outcome: successful, successful with issues (describe), or rolled back (describe). Close the change record in the program tracker.

WARNING: Never execute a major change to a production MSS product without a documented rollback procedure. The rollback procedure must specify: who executes it, what commands or actions are required, how long it will take, and how consumers will be notified of the rollback. A change with no rollback plan is an outage risk.

9-5. Managing the Program Change Log

Every significant program decision must be recorded. The change log is the institutional memory of the program — the record that answers the question every new arrival asks: "Why is it done this way?"

Table 9-4. Program Change Log — Required Entries

Entry Type	Trigger	Required Information
Architecture decision	Any decision about platform architecture, ontology design, or pipeline structure that will persist beyond a single sprint	Decision summary, options considered, rationale, date, authority
Policy change	Any change to program standards, governance policy, or the Decision Authority Matrix	Old policy, new policy, reason, effective date, authority

Entry Type	Trigger	Required Information
Stakeholder change	New key stakeholder, departure of a key stakeholder, or significant change in a stakeholder's role or priorities	Name, role, date of change, impact on program, transition plan
Vendor change	New task order, modification, FDE change, or significant contractor performance event	Event description, date, impact, action taken
Technology change	Palantir platform update, deprecation, or significant capability change affecting production	Change description, affected products, migration status
Program structure change	Team reorganization, new team added, team dissolved	Old structure, new structure, reason, effective date

The change log lives in the program tracking workspace on MSS. It is maintained by the program manager and is readable by all team leads. Entries should be made within 48 hours of the triggering event.

9-6. Advanced Portfolio Governance: Multi-Product Dependency and Friction Analysis

BLUF: SL 4J introduced the DDOF friction matrix, PM oversight roles, and portfolio health metrics for individual products. SL 5J extends these into multi-product dependency mapping, cascading friction analysis, and resource optimization across competing DDOF efforts — the portfolio-level view that prevents local product decisions from creating systemic program risk.

9-6a. Multi-Product Dependency Mapping

DDOF products do not exist in isolation. An ontology object feeds a pipeline, which feeds a dashboard, which feeds a model, which feeds a decision product. The SL 5J program manager maps these dependencies explicitly:

1. **Build the product dependency graph.** For every active DDOF product, document: upstream data sources (what feeds it), downstream consumers (what depends on it), shared ontology objects, and shared pipeline components.
2. **Identify critical path products.** Products with many downstream dependents are single points of failure. Flag any product with three or more downstream consumers as a critical path item requiring enhanced monitoring and change control.
3. **Publish the dependency map.** Make the dependency graph visible to all team leads. Update it at every PI Planning cycle. Use it as the primary artifact for cross-team dependency management.

9-6b. Cascading Friction Analysis

A friction point in one product cascades through its dependents. The SL 5J PM conducts cascading friction analysis by:

1. **Identify friction sources.** Use the DDOF friction matrix categories: data quality friction, pipeline reliability friction, ontology stability friction, and governance friction.
2. **Trace downstream impact.** For each friction source, follow the dependency graph to determine which downstream products are affected. Quantify the impact: degraded data quality, delayed refresh, or complete failure.
3. **Prioritize by blast radius.** Friction with the widest downstream impact gets priority remediation regardless of which team owns the source product. This is a portfolio-level decision, not a team-level one.

9-6c. Resource Optimization Across Competing DDOF Efforts

When multiple DDOF efforts compete for the same resources (ORSA analyst time, FDE support, ontology review capacity, compute allocation), the SL 5J PM allocates based on operational priority and dependency criticality — not on which team lead argues most persuasively. Use the dependency map and cascading friction analysis to justify allocation decisions with evidence.

NOTE (DDOF Playbook v2.2 — Retirement Criteria Enforcement): The program manager is accountable for enforcing DDOF automatic retirement triggers. These are not guidelines — they are program standards:

- **90 days no access → Foundry Data Manager (FDM) review.** If no user has accessed the product in 90 days, initiate a formal review with the product owner. Determine whether the product serves a valid but infrequent need (e.g., annual reporting) or has been abandoned.
- **180 days no access → retirement.** If the FDM review does not produce a documented justification for retention, retire the product. Remove from production, archive data per retention policy, and update the portfolio tracker.
- **Quality score below 70% → remediate or retire.** If a product's data quality score (completeness, accuracy, timeliness, consistency) falls below 70% on the quarterly quality assessment, the product owner has one PI cycle to remediate. If quality remains below threshold after remediation, retire the product.

Failure to enforce retirement criteria results in portfolio bloat: unmaintained products consuming platform resources, creating false signals in operational dashboards, and degrading user trust in the MSS ecosystem.

CHAPTER 10 — PROGRAM COMMUNICATION AND REPORTING

10-1. The Program Manager as Information Integrator

BLUF: The SL 5J program manager is the single point of integration for information flowing between technical teams, operational stakeholders, and senior leadership. Information that is not surfaced clearly and on time becomes a liability.

At SL 4J level, communication is primarily project-scoped: stakeholder updates on one project, sprint review reports for one team, escalations on one blocker. At SL 5J level, the program manager is synthesizing information from five to ten teams simultaneously, integrating technical status with operational impact, and translating between the language of Agile delivery and the language of military readiness reporting.

This is an underappreciated skill. A technically excellent PM who cannot communicate program status clearly to a General will fail at the program level just as surely as one who cannot manage a PI.

10-2. Communication Architecture

A program communication architecture defines the right information to the right audience at the right cadence. Without it, communication defaults to reactive — the PM responds to questions rather than shaping understanding.

Table 10-1. Program Communication Architecture

Communication	Audience	Cadence	Format	Owner
Executive Program Brief	CG/DCG/G6/EUCOM J6	Quarterly (or post-PI)	6-slide commander's brief	Program Manager
Program Review	C2DAO Chief, G-staff data leads	Monthly	Dashboard walkthrough + risk brief	Program Manager
PI Review Report	C2DAO, G-staff, all team leads	Post-PI (every 8-12 weeks)	Written report + dashboard	Program Manager
Program Sync Readout	C2DAO Chief (as needed)	Weekly (summary via email)	1-paragraph SITREP	Program Manager
Sprint Review Summary	C2DAO, stakeholders	Bi-weekly (per team)	Delivery summary email	Team Lead
Incident Report	C2DAO, affected stakeholders	Within 24 hours of incident	Written memo	Program Manager

Communication	Audience	Cadence	Format	Owner
New Capability Announcement	All affected user units	2 weeks before release	Email + unit brief	Team Lead + PM

10-3. Writing the Program SITREP

The weekly program SITREP is the primary written communication artifact for the C2DAO Chief and G6. It must be: accurate, brief, and action-oriented.

Format:

```
MSS PROGRAM SITREP — [Date]
PREPARED BY: [Name], Program Manager

1. SUMMARY (2-3 sentences): Overall program status. Any status changes from last week.

2. DELIVERY STATUS (table):
   Team | Sprint | Key Deliveries This Week | Status
   [...]

3. RISKS AND ISSUES (list only Active/Escalated):
   - [Risk]: [Status] — [Mitigation owner] — [Next action by date]

4. DEPENDENCIES (blocked only):
   - [Dependency]: [Blocked reason] — [Owner] — [Resolution timeline]

5. DECISIONS REQUIRED (if any):
   - [Decision needed] — [From whom] — [By date]

6. NEXT WEEK:
   - [Key events, ceremonies, or milestones in next 7 days]
```

NOTE: The program SITREP should take no more than 30 minutes to write. If it is taking longer, the program tracker is not current. The SITREP is a synthesis of data that should already be in the tracker — not a data collection exercise.

10-4. Communicating Failure

BLUF: The program manager who communicates program failure clearly and early, with a mitigation plan, builds more trust than the PM who papers over problems until they are unavoidable. The ability to communicate bad news well is a core SL 5J competency.

Principles for communicating program failure to senior leadership:

State the BLUF first. "The theater readiness dashboard will not be ready for the Q2 operational review as originally planned. We will deliver a partial capability — four of six required metrics — by that date. Full capability delivery is now projected for [date]."

Own the assessment. Do not attribute the failure to external factors until you have exhausted the internal root cause. Leaders can smell deflection. Even if the failure is caused by a late contractor deliverable, the PM is accountable for not having surfaced the risk earlier.

Present the mitigation. Every communication about a failure must include what you are doing about it. "We have accelerated the remaining development by reallocating one SWE from Team Bravo for two sprints and have held a formal acceptance review on the outstanding Palantir deliverable. We anticipate the gap will close by [date]."

Distinguish recoverable from structural. Some failures are sprint-level anomalies. Others indicate a structural problem: estimation is consistently wrong, a team is chronically understaffed, a data feed is unreliable. If the root cause is structural, say so — and present the structural solution, not just the sprint-level patch.

10-5. Managing Program Narrative Over Time

Senior leaders do not always attend every program review. Their understanding of the MSS program is shaped by fragments: a five-minute hallway conversation with the G6, a slide from a brief they half-attended, an incident they heard about secondhand. The program manager does not control all of these touchpoints — but must be aware that they exist and work actively to shape a coherent narrative.

Strategic communication principles:

- **Maintain a program one-pager.** A single page that can be handed to any senior leader and conveys: what MSS is, what it currently does for USAREUR-AF, what is being built next, and who to contact. Update it quarterly. Distribute it to the GO/SES administrative teams so it is available when needed.
- **Brief accomplishments, not activity.** "The team held 40 sprint ceremonies this quarter" is activity. "We delivered the theater readiness dashboard, now used daily by 14 G3 analysts to replace a 4-hour manual reporting process" is an accomplishment. Always translate delivery activity into operational impact.
- **Own the incident narrative.** When a production outage or data quality incident occurs, the program manager must be the first to communicate — not the last. Leaders who hear about incidents from end users before hearing from the PM lose confidence in the program's self-awareness. Communicate the incident, the cause, and the corrective action in a single message, proactively.
- **Document endorsements.** When a GO, DCG, or senior staff officer provides a positive assessment of an MSS product — "this dashboard is exactly what I needed" — document it. Store it in the program tracker. Use it in future investment justifications and in new leadership orientation briefs. Operational endorsements are the program's most persuasive evidence of value.

CHAPTER 11 — ML/AI PORTFOLIO MANAGEMENT

11-1. Managing Research, Build, and Sustainment Simultaneously

NOTE — Palantir Developers reference: *Product Launch: Enterprise Automation* | *DevCon 5* — Covers Palantir's enterprise automation capabilities and the program management model for deploying automation at scale across multiple teams and production systems. Directly reinforces the portfolio-level ML/AI management content in this chapter. Available on the Palantir Developers YouTube channel (@PalantirDevelopers).

BLUF: An ML/AI portfolio at program scale contains products at every stage of the lifecycle simultaneously. Some are in active research (early exploration, feasibility unknown), some are in active development (models being trained and evaluated), and some are in production sustainment (models serving operational users and requiring ongoing monitoring). Each stage requires different management attention and different risk posture.

The SL 4M track covers the technical lifecycle of a single ML project. SL 5J covers the portfolio-level challenge: how do you manage five ML projects simultaneously when they are at different stages, competing for the same data scientist resources, and serving different operational stakeholders with different tolerance for model risk?

Table 11-1. ML Portfolio Stage Management

Stage	Description	PM Focus	Risk	Typical Duration
Research	Feasibility exploration; no committed output	Timebox; define decision criteria; set a no-go trigger	Sunk cost; scope creep into build without decision gate	4-8 weeks
Development	Model being built; training and evaluation ongoing	Requirements stability; data availability; evaluation criteria; Definition of Done	Wrong objective; inadequate evaluation; deployment surprise	2-4 PIs
Production	Model serving users; outputs feeding dashboards or decisions	Performance monitoring; drift detection; retraining schedule; user feedback loop	Silent degradation; over-reliance; undocumented model changes	Indefinite
Sustainment Review	Periodic assessment of whether production model remains fit for purpose	Re-evaluate against current requirements;	Outdated model continues to serve	Quarterly

Stage	Description	PM Focus	Risk	Typical Duration
		assess drift; decide continue/retrain/retire	users without reassessment	
Retirement	Model being phased out; replacement being built	User transition; data archive; documentation preservation	Dependent products lose their input without notice	90 days minimum

11-2. The Research Gate

Research projects are the highest-discretionary-spend items in the portfolio. They have no guaranteed output, and they compete for scarce ORSA and MLE resources with projects that are already committed to operational users. The program manager must govern research investment deliberately.

Every research project must have:

1. **A stated hypothesis.** "We believe that a model trained on X data can predict Y with sufficient accuracy to improve Z operational decision." If you cannot state the hypothesis clearly, the research should not start.
2. **A decision timeline.** Research does not run indefinitely. Set a timebox: at 6 weeks, the team will present findings and recommend proceed, pivot, or stop. This is not a delivery commitment — it is a decision commitment.
3. **Defined go/no-go criteria.** What does "sufficient accuracy" mean? What baseline does the model need to beat? What data quality threshold must be met? If the team cannot define what success looks like before they start, they cannot make a defensible decision when the timebox expires.
4. **A resource ceiling.** How many engineer-weeks is this research worth? A research project that consumes unlimited resources without producing a decision is a portfolio liability.

Table 11-2. Research Gate Decision Criteria Template

Criterion	Threshold	Assessment Method
Model accuracy	Beats baseline by >X% on hold-out test set	Documented evaluation results
Data availability	Required input data available and stable for >Y% of records	Data quality assessment (SL 4M)
Operational feasibility	Model outputs can be integrated into MSS workflow without >Z weeks of additional development	Technical feasibility review with SL 4H
Stakeholder acceptance	Operational stakeholder has reviewed sample outputs and confirmed utility	Stakeholder review session documented

Criterion	Threshold	Assessment Method
Maintenance viability	Team has documented retraining cadence and performance monitoring plan	Sustainment plan document

11-3. AI Governance Responsibilities

Army CIO Memo (April 2024) establishes requirements for AI system governance. The SL 5J program manager is accountable for ensuring compliance across all AI-assisted MSS products.

Table 11-3. AI Governance Checklist — SL 5J Accountability

Requirement	Description	Verification
Product documentation	All AI-assisted products documented with model type, training data, intended use, and limitations	Review product documentation in MSS workspace
Human review gates	All AI-generated outputs that inform operational decisions have a defined human review step	Verify workflow documentation; confirm with operational users
Performance monitoring	Model performance metrics tracked; alert thresholds defined	Review monitoring dashboard (SL 4M product)
Bias assessment	Training data and outputs assessed for systematic bias relevant to the use case	Review bias assessment documentation
Retraining schedule	Schedule for periodic model retraining defined and calendared	Review sustainment plan
Escalation pathway	Defined process for operational users to flag model outputs they believe are incorrect	Verify user guidance and helpdesk capability

WARNING: Deploying an AI-assisted MSS product without documenting the human review gates is a policy violation under Army CIO Memo (April 2024). More critically, it creates operational risk: an AI system that produces incorrect outputs with no review mechanism can corrupt the operational picture before the error is detected. Verify human review gates at every production release review.

11-4. The ML Model Handoff Problem

One of the most common failure modes in ML portfolio management is the model handoff: a research team trains an excellent model, writes a paper-quality evaluation report, hands it to a sustainment team, and walks away. Six months later the model is producing degraded outputs and no one on the sustainment team understands the model well enough to diagnose the problem.

The program manager's job is to ensure this does not happen. The handoff from development to sustainment is not a documentation event — it is a capability transfer that must be verified.

TASK 11-4A: EXECUTE AN ML MODEL HANDOFF REVIEW

CONDITIONS: An ML model has completed development and passed its production readiness gate. A sustainment team has been designated to own the model in production.

STANDARDS: Handoff is complete when: the sustainment team lead can independently describe the model's inputs, outputs, and known limitations; a monitoring dashboard exists and the sustainment team can interpret its alerts; the retraining procedure has been documented and walked through once with the sustainment team; the handoff has been signed off by both the development lead and the sustainment team lead.

EQUIPMENT: MSS Foundry (model workspace, monitoring dashboard), model documentation (training data, architecture, evaluation results, retraining procedure), program tracker (product ownership record, sustainment schedule), development team lead and sustainment team lead.

PROCEDURE:

1. **Schedule a model walkthrough session.** Development lead walks the sustainment team through: training data and preprocessing logic; model architecture and hyperparameter choices; evaluation results and known failure modes; the monitoring metrics being tracked in production; the retraining trigger conditions and procedure.
2. **Test the sustainment team's understanding.** Present three hypothetical scenarios to the sustainment team: (a) The monitoring dashboard shows the model's AUC has dropped from 0.82 to 0.74 over three weeks. What do you do? (b) A G1 analyst reports that the model's 30-day outlook for one brigade is flagging a readiness decline that "doesn't match what we're seeing on the ground." How do you investigate? (c) A new data feed is being added that changes the distribution of one input feature. What is the process?
3. **Document the responses.** If the sustainment team cannot answer these questions confidently, the handoff is not complete. Extend the overlap period and repeat the walkthrough.
4. **Transfer ownership in the program tracker.** Update the product ownership record in the MSS program tracker: from development team to sustainment team. Update the on-call rotation and incident response contacts.
5. **Schedule the first sustainment review.** Per Table 11-1, sustainment reviews are quarterly. Schedule the first review before the handoff is closed. The development team lead attends the first sustainment review as a resource — not as the owner.

NOTE: A model in production with no one who understands it is a liability, not an asset. The production release gate is the point of maximum PM accountability: if you release a model into production and the sustainment team is not prepared to own it, the failure is yours.

CHAPTER 12 — PROGRAM FINANCIAL STEWARDSHIP

12-1. Resource Allocation at Program Level

BLUF: The SL 5J program manager does not manage contracts or obligation rates. The PM does manage the allocation of team capacity — the most finite and non-recoverable resource in the program. A sprint-week of engineering capacity spent on the wrong priority is gone permanently.

Resource stewardship at the program level means: ensuring each team's capacity is applied to the highest-value work, balancing delivery pressure against maintenance and debt obligations, and protecting team capacity from the entropy of unplanned requests that do not go through the backlog.

12-2. Capacity Allocation Policy

Table 12-1. Recommended Program Capacity Allocation

Category	Allocation	Description
New feature delivery (committed PIO)	60-65%	Features committed at PI Planning to operational stakeholders
Technical debt reduction	15-20%	Planned debt work from the debt backlog (mandatory; not optional)
Unplanned work / operational support	10%	Bug fixes, urgent requests, production support — hold this capacity in reserve
Research and exploration	5-10%	Allocated to current research projects with a gate decision pending

CAUTION: Teams that routinely exceed 65% capacity on new feature delivery at the expense of debt reduction are borrowing from the future. After two to three PIs of zero debt reduction allocation, velocity typically begins to decline as accumulated debt creates compounding friction. Protect the debt allocation even under delivery pressure.

TASK 12-2A: CONDUCT A PROGRAM CAPACITY ALLOCATION REVIEW

CONDITIONS: Approaching PI Planning. A list of stakeholder requests for the upcoming PI has been compiled.

STANDARDS: Review is complete when: available capacity per team for the upcoming PI is calculated; requested work has been prioritized against capacity; allocation across the four categories in Table 12-1 is confirmed; the capacity allocation is presented to C2DAO before PI Planning begins.

EQUIPMENT: Team roster and leave/TDY schedules for the upcoming PI, program tracker (backlog, debt backlog, active research items), capacity allocation framework (Table 12-1), stakeholder request list, C2DAO calendar for briefing.

PROCEDURE:

1. **Calculate available capacity.** For each team: starting capacity (team size × sprint days × number of sprints in PI). Subtract: known leave, TDY, training events, platform maintenance windows, expected contractor availability gaps. This is the net available capacity for the PI.
2. **Reserve 10% for unplanned work.** Remove 10% from each team's available capacity immediately. This is the unplanned work buffer. It is not available for new PIO commitment. If it is not consumed, it flows to debt reduction at the end of the PI.
3. **Allocate 15-20% to debt reduction.** Pull the top-priority debt items from the debt backlog. Assign them to the appropriate teams. This allocation is fixed — it is not available for stakeholder requests.
4. **Prioritize stakeholder requests against remaining capacity.** Bring the remaining capacity (approximately 65%) to the prioritization session with C2DAO and operational stakeholders. Work through the request list in priority order until capacity is consumed. Requests that do not fit go to the next PI or are explicitly deferred.
5. **Confirm research allocation.** If any active research projects are ongoing, confirm they have the capacity they need within the 5-10% research allocation. Research that would require more than 10% of program capacity needs explicit C2DAO authorization.
6. **Document and brief.** Produce a one-page capacity allocation summary. Brief to C2DAO Chief before PI Planning. Capacity is not available for re-negotiation during PI Planning — commitments are made against the confirmed available capacity.

12-3. Protecting Team Capacity from Scope Creep

Unplanned work that bypasses the backlog is the most common form of capacity waste at program scale. It typically enters through three vectors:

The urgent stakeholder call. A GO or senior staff officer contacts a team lead directly and asks for "a quick dashboard" or "a small change." The team lead, unwilling to say no to a senior, starts the work without PM visibility.

The informal FDE add-on. A Palantir FDE, trying to be helpful, begins building a feature that was discussed informally but never formally scoped or prioritized.

The developer side project. A team member finds an interesting optimization or feature improvement and begins working on it during sprint time without a backlog item.

All three represent untracked work that consumes capacity committed to PI objectives.

Counter-measures:

- Establish a clear intake rule: all work requires a backlog item before development begins. No exceptions. Team leads are empowered to say "I'll create a ticket and route it through the PM" to any informal request.
- Brief operational stakeholders on the intake process at the beginning of each PI. Reinforce at every sprint review.
- Include "work not in the backlog this sprint" as a standing retrospective question. If teams discover shadow work in retrospectives, the intake process is not working.
- For FDE-initiated work: establish a rule that Palantir FDE activity must be covered by a task order line item or a formally created government backlog item before work begins. Document any FDE activity at the weekly task order status meeting.

12-4. Making the Case for Data Investment

The SL 5J program manager will periodically need to justify the MSS program investment to leadership — especially when budget pressures mount, when a new leader arrives skeptical of data platforms, or when a competing priority threatens to draw resources away from the program.

The case for data investment is not made with technical arguments. It is made with operational outcomes.

Table 12-2. Data Investment Justification Framework

Argument Type	Example	Appropriate For	Avoid
Time savings	"The theater readiness dashboard eliminated 4 hours of daily manual compilation by G3 analysts — equivalent to 1,460 analyst-hours per year reclaimed for analysis."	G6, G3, any operational leader	GO-level briefs (too granular)
Decision quality	"The supply chain risk model surfaced a parts shortage 6 weeks before it became a readiness issue, enabling preemptive action. Manual methods detected it at 2 weeks — too late to mitigate."	GO/SES; DCG	Audiences unfamiliar with the model
Coverage increase	"MSS pipelines now provide real-time readiness visibility for 94% of theater units, up from 40% before the program. The remaining 6% are units with delayed reporting to GCSS-A — a command issue, not a data issue."	G3/G4; any operational leader	Without the caveat about reporting discipline
Risk reduction	"The ontology governance structure has prevented three data integration errors in the past PI that would have produced incorrect readiness metrics in the theater dashboard."	C2DAO Chief; program technical audience	GO-level (too technical)

Argument Type	Example	Appropriate For	Avoid
Comparative	"Peer AORs operating comparable MSS programs have reduced staff reporting burden by an average of 30% and improved their senior leader brief cycle time from 48 hours to 4."	New senior leaders; skeptical stakeholders	Without citing the source

CAUTION: Do not fabricate or embellish data investment arguments. Senior leaders, particularly those with analytical backgrounds, will probe the numbers. One challenged or unsupported claim undermines all the others. Every figure you use in an investment justification must be traceable to a documented source.

12-5. Communicating Capacity Constraints to Stakeholders

The program manager will regularly face requests for new capability that exceed available capacity. Saying no to a GO or senior staff officer is politically difficult — but saying yes without capacity to deliver is worse.

Framework for capacity-constrained conversations:

1. **Confirm the requirement.** "I want to make sure I understand what you need: [restate the requirement]. Is that correct?"
2. **Confirm the priority.** "How does this compare in priority to [the currently committed PI work]?"
3. **Present the trade-off.** "I can accommodate this in the current PI if we defer [specific committed work]. Alternatively, I can commit it as our first priority in the next PI starting [date]. Which do you prefer?"
4. **Never say capacity without substance.** "We don't have capacity" is not a useful answer. "If I pull this into the current PI, I will need to defer the supply chain Phase 2 integration, which [G4 name] has been waiting for since last PI. I recommend we add it to PI 5 to avoid that impact" is a useful answer.
5. **Document the decision.** Whatever the leader decides — pull it in with a trade, defer to next PI, or accept a reduced scope — document it in writing and update the program tracker.

CHAPTER 13 — LEADING THROUGH PROGRAM TRANSITIONS

13-1. The Program Manager's Role in Organizational Change

BLUF: Data programs operate inside organizations that are themselves changing. Personnel rotate, command priorities shift, platform capabilities evolve, and policy frameworks update. The SL 5J program manager must lead the program through transitions without losing the momentum of ongoing delivery.

Transitions at program scale come in several forms:

Leadership transitions. A new G6, C2DAO Chief, or commanding general arrives with different priorities or lower familiarity with the MSS program. The PM must re-establish relationship, re-brief the program, and manage the risk that the new leader deprioritizes data investment.

Team transitions. Military personnel rotate. Key technical practitioners leave. New team members arrive without context. The PM must maintain delivery continuity while onboarding new arrivals.

Platform transitions. Palantir releases a significant platform update, or a policy change requires architectural modifications. The PM must manage the transition without breaking production products.

Scope transitions. The program receives a significant new mission — theater-wide expansion of MSS, integration of a new data domain, or a new operational requirement from EUCOM. The PM must absorb the new scope without disrupting ongoing delivery.

13-2. Managing Leadership Transitions

TASK 13-2A: EXECUTE A PROGRAM LEADERSHIP TRANSITION BRIEF

CONDITIONS: A new GO, SES, or senior leader with program oversight authority has assumed their position or is incoming within 60 days.

STANDARDS: Transition is complete when: the incoming leader has received a program brief; key relationships have been established; the leader's priorities for the program have been documented; a follow-up brief schedule has been established.

EQUIPMENT: Program transition brief (prepared per Appendix D format), program one-pager (Section 13-2A), MSS program tracker (current status, roadmap, risk register), stakeholder register (Table 4-1), communication channel to the incoming leader's executive officer or aide.

PROCEDURE:

1. **Prepare the transition brief.** Structure: (1) MSS program mission and theater-level value — what operational problems MSS solves. (2) Current program health — delivery performance, top risks, key stakeholders. (3) Key products and their operational consumers — who uses what, and what decisions they support. (4) Portfolio roadmap — what is planned for the next PI and the next year. (5) Request: what the program needs from this leader to succeed (sustained investment, stakeholder access, policy guidance).
2. **Request the brief through appropriate channels.** Do not attempt to schedule directly with a GO. Request through the executive officer or aide. Brief the request: "Request 30 minutes to brief the MSS program status and roadmap at the new leader's convenience within the first 30 days of command."
3. **Conduct the brief.** Lead with operational value — what decisions the program supports, what visibility it provides commanders. Follow with status. Close with what you need. Leave time for questions.

4. **Document priorities and guidance.** After the brief, send a follow-up memo or email summarizing: what was briefed, what guidance or direction the leader provided, and the agreed follow-up actions. This creates a record and confirms shared understanding.
5. **Schedule the follow-on cadence.** Establish the regular briefing cadence (see Table 10-1) with the new leader's schedule.

NOTE: A new senior leader who does not receive a program brief in their first 30 days will form their opinion of the MSS program from informal channels — other staff officers, their predecessor's notes, or their prior experience with data programs. Control the narrative by briefing proactively.

13-2A. New Leader Orientation — Program Leave-Behind

When briefing a new senior leader, always leave behind a concise one-page document that they can reference after the brief. A GO who attended your orientation brief six months ago will not remember the details — but they will pull the leave-behind when they need to brief their own boss on the data program.

Program One-Pager Structure:

1. **Mission** (1 sentence): "MSS provides USAREUR-AF and EUCOM commanders with real-time operational data visibility and AI-assisted analysis across readiness, logistics, intelligence, and personnel domains."
2. **What MSS Enables Today** (3-5 bullets): Specific operational capabilities, with the operational user and the decision they support.
3. **Current Portfolio** (small table): Active products, their operational domain, their primary user unit, and their status (Active / In Development / Planned).
4. **Building Next** (3 bullets): The top three capabilities under development in the current PI.
5. **How to Access / Get Help**: Contact information for the C2DAO helpdesk; link to the MSS user training page (learn-data.armydev.com); name of the program manager.
6. **Authority**: "MSS is authorized under Army CIO Data and Analytics Policy (April 2024). Governed by USAREUR-AF C2DAO."

Update this document after every PI Review. Distribute to: the G6 front office, the DCG action officer, the EUCOM J6 rep, and any new unit S6 shops coming online.

13-3. Managing Platform Transitions

Palantir releases platform updates that can affect production MSS products. Some updates are transparent to government-built products. Others — particularly changes to ontology APIs, Pipeline Builder syntax, or Workshop widget behavior — can break production configurations without warning.

Table 13-1. Platform Transition Risk Matrix

Update Type	Typical Risk	Detection	Response
Minor platform release (bug fixes)	Low	Platform release notes	Review notes; monitor production for 48 hours
Major platform release (new features)	Medium	Palantir advance notice (typically 30 days)	Review documentation; assess impact on production products; test in dev environment
Deprecated API or feature	High	Palantir deprecation notice	Inventory all products using deprecated feature; plan migration; execute before deprecation date
Schema-breaking ontology update	Very High	Palantir advance notice + C2DAO coordination	Assess downstream impact; coordinate migration with all affected teams; test thoroughly
Security patch	Variable	Emergency notice from Palantir	Assess impact; apply in coordination with C2DAO ISSO; test immediately

Table 13-1. Platform Transition Preparation Checklist

Action	Timing	Owner
Subscribe to Palantir release notes and deprecation notices	Ongoing	Platform Team Lead (SL 5L)
Maintain a platform dependency inventory: which products use which APIs	Quarterly review	Platform Team Lead (SL 5L)
Reserve 5% of PI capacity for platform adaptation work	Every PI	Program Manager
Coordinate platform update timing with active sprint schedule	Before each Palantir release	Program Manager + FDE
Test all production products in dev environment before major platform updates go live	Before each major release	All team leads

TASK 13-3A: MANAGE A PALANTIR PLATFORM DEPRECATION

CONDITIONS: Palantir has issued a deprecation notice for a platform feature currently used in one or more production MSS products. A deprecation date has been communicated.

STANDARDS: Migration is complete before the deprecation date. All affected products have been assessed, migrated, tested, and re-released. No production product is broken by the deprecation event.

EQUIPMENT: Palantir deprecation notice (with deprecation date and replacement feature documentation), platform dependency inventory (from Platform Team Lead), MSS Foundry (development/staging environment for migration testing), program tracker (migration backlog items),

Palantir FDE (migration guidance and documentation).

PROCEDURE:

1. **Inventory affected products.** Survey all team leads: which products use the deprecated feature? Document each affected product, the specific usage, and the estimated migration effort.
2. **Estimate total migration effort.** Aggregate the per-product estimates. Determine whether the migration can be completed within normal PI capacity or whether a dedicated migration sprint is needed.
3. **Create migration backlog items.** For each affected product, create a backlog item: "Migrate [product] from [deprecated feature] to [replacement]." Assign to the responsible team. Set priority: all deprecation migrations must be completed at least two sprints before the deprecation date to allow for testing.
4. **Coordinate with Palantir.** Confirm the deprecation timeline with the FDE. Ask: Is there any possibility of extension? Are there known migration pitfalls? Does Palantir have a migration guide or script?
5. **Execute and test migrations.** Teams execute migrations per the backlog items. Each migration must be tested in a development environment before being promoted to production.
6. **Monitor after deprecation date.** Even with all known products migrated, monitor production for unexpected failures in the two weeks after the deprecation takes effect. Undocumented usage of deprecated features may surface as production errors.

APPENDIX A — PROGRAM HEALTH ASSESSMENT CHECKLIST

A-1. PURPOSE

The Program Health Assessment (PHA) is a structured review of program status across seven dimensions. Conduct the PHA quarterly — or after any significant program event (major release, leadership change, team restructure, contract modification). The PHA is the program manager's equivalent of a unit readiness review: an honest, evidence-based assessment of what is working and what is not.

A-2. ASSESSMENT DIMENSIONS AND CRITERIA

Dimension 1: Delivery Performance

Criterion	Green	Amber	Red
PI predictability (last 2 PIs)	>85%	70-85%	<70%
Velocity trend (last 6 sprints)	Stable or improving	Minor decline (<15%)	Sustained decline (>15%) or high variance
Escaped defect rate	<2 per PI per team	2-5 per PI per team	>5 per PI per team
Blocked time per sprint	<10% of capacity	10-20% of capacity	>20% of capacity

Dimension 2: Stakeholder Health

Criterion	Green	Amber	Red
GO/SES engagement	Regular access; program is on their radar	Infrequent access; interest declining	No recent GO/SES engagement; program invisible at senior level
Requirements clarity	All active features have clear, validated requirements	Some features have ambiguous requirements	Major active features have no validated requirements
User adoption	Active users growing; adoption metrics positive	Flat adoption; known resistance in one or more units	Declining users; active pushback from operational units
Stakeholder satisfaction	Positive feedback from key stakeholders; complaints rare and addressed	Mixed feedback; unresolved complaints present	Significant negative stakeholder feedback; trust issues with key relationships

Dimension 3: Technical Health

Criterion	Green	Amber	Red
Technical debt level	Debt is tracked; <20% of capacity allocated to debt is not being used	Debt is tracked but growing; allocation insufficient	Debt is untracked, unquantified, or consuming >30% of capacity
Production stability	Fewer than 2 production incidents per month; all resolved within SLA	2-5 incidents per month; SLA occasionally missed	Frequent production incidents; recurring root causes unaddressed
Documentation coverage	All production pipelines and critical business logic documented	Most documentation current; some gaps	Significant undocumented production systems

Criterion	Green	Amber	Red
Test coverage	All production transforms have automated data quality checks	Some gaps in test coverage; no recent test failures from gaps	Known gaps in test coverage; uncaught errors in production

Dimension 4: Team Health

Criterion	Green	Amber	Red
Team stability (rolling 6 months)	<25% team membership change	25-40% team membership change	>40% team membership change; continuity impacted
Morale indicators	Teams meeting, delivering, retrospecting with positive engagement	Some teams showing low engagement; PM has plan	Visible morale issues; attrition risk; retrospectives not being held
Knowledge continuity	All critical knowledge documented; onboarding packages current	Most knowledge documented; some single points of failure	Multiple undocumented single points of failure; rotation risk is high
PM development	All SL 4J PMs have development plans; assessments current	Most PMs assessed; some development plans stale	No PM assessments; no development activity

Dimension 5: Vendor/Contractor Health

Criterion	Green	Amber	Red
Task order delivery	All deliverables on schedule; acceptance criteria met on first submission	1-2 deliverables late or requiring rework	Multiple deliverables overdue or failing acceptance
FDE engagement quality	FDE is proactive, technically competent, aligned to government priorities	FDE engagement is adequate but reactive	FDE is disengaged, technically misaligned, or churning
Performance documentation	All acceptance events documented; performance record current	Some documentation gaps; no major events undocumented	Significant performance events undocumented; record incomplete

Dimension 6: Portfolio Coherence

Criterion	Green	Amber	Red
Cross-product integration	Products share common ontology objects; data is consistent across products	Some inconsistency across products; integration gaps identified and planned	Products are siloed; no common data model; conflicting definitions between dashboards
Dependency management	All inter-team dependencies tracked and current	Dependency register exists but some entries stale	No dependency register; dependencies discovered at impact
Build/buy/configure discipline	All significant investment decisions documented; configure-first principle enforced	Most decisions documented; a few undocumented build decisions	Pattern of building what could be configured; undocumented decision making

Dimension 7: Policy Compliance

Criterion	Green	Amber	Red
Data governance	All products comply with UDRA v1.1 data management requirements	Most products compliant; minor gaps being addressed	Known compliance gaps unaddressed; no remediation plan
Access management	All Foundry project access reviewed in last 90 days; least-privilege enforced	Access reviewed but minor exceptions present	Access not reviewed; over-permissioned accounts present
Army CIO policy	All AI-assisted products comply with Army CIO Memo (April 2024)	Compliance largely maintained; minor gaps	Known AI products out of compliance with Army CIO policy

A-3. PHA SCORING AND ACTION

Table A-1. PHA Overall Health Rating

Overall Rating	Criteria	Action
GREEN	All 7 dimensions Green or Amber	Continue current approach; maintain monitoring
AMBER	1-3 dimensions Red, or majority Amber	Present mitigation plan to C2DAO within 2 weeks; track recovery at next PHA
RED	4+ dimensions Red, or any critical dimension (Delivery, Technical, Team) Red	Escalate to C2DAO leadership immediately; conduct root cause analysis; present recovery plan within 1 week

APPENDIX B — DATA PRODUCT LIFECYCLE AND RETIREMENT GUIDE

B-1. THE DATA PRODUCT LIFECYCLE

Every MSS data product has a lifecycle: it is conceived, built, released, maintained, evolved, and eventually retired. The SL 5J program manager is responsible for managing this lifecycle deliberately — not allowing products to drift into obsolescence while remaining nominally in production.

Table B-1. MSS Data Product Lifecycle Phases

Phase	Description	PM Actions
Concept	Requirement identified; feasibility assessed; build decision made	Document requirement; conduct build/buy/configure review; assign product owner
Build	Active development; feature delivery through Agile sprints	Sprint oversight; stakeholder review; Definition of Done enforcement
Production Release	Product deployed to operational users	Release approval; user communication; training delivery
Active Maintenance	Product in production; bug fixes and minor enhancements	SLA monitoring; defect management; periodic enhancement PI allocation
Enhancement	Major new capability added to existing product	New PI objectives; stakeholder requirements; version management
Sunset Pending	Product identified for retirement; replacement being built or need eliminated	User communication; transition planning; data export/archive decisions
Retired	Product removed from production; no longer supported	Archive decision; documentation preservation; user confirmation

B-2. VERSION MANAGEMENT

MSS data products — like software — should be versioned. Versioning enables rollback, enables communication about what changed, and creates a record for audit and compliance.

Table B-2. MSS Product Version Numbering Convention

Version Change	Convention	Example	When to Use
Major release	Increment first number	1.0 → 2.0	Significant schema change; breaking change to downstream consumers; new product architecture
Minor release	Increment second number	1.3 → 1.4	New features; non-breaking changes; significant bug fixes
Patch	Increment third number	1.4.1 → 1.4.2	Bug fixes; data quality corrections; minor configuration updates

Maintain a changelog in the product's MSS documentation workspace. Each version entry includes: version number, release date, changes summary, breaking changes (if any), and downstream products affected.

NOTE: "Breaking change" for a data product means any change that requires downstream consumers (other pipelines, dashboards, models) to modify their configuration or code to continue functioning. All breaking changes require notification to downstream product owners at least two sprint intervals before release.

B-3. PRODUCT RETIREMENT PROCEDURE

TASK B-3A: RETIRE AN MSS DATA PRODUCT

CONDITIONS: A decision has been made to retire an MSS product from production. The decision has been reviewed by C2DAO and documented. A replacement capability exists or the operational need has been eliminated.

DDOF Playbook v2.2 Automatic Retirement Triggers: Per DDOF Playbook v2.2 (T2COM C2DAO, December 2025), the following conditions automatically trigger retirement review:

Trigger	Action Required
No access in 90 days	FDM review required — determine if product still has operational value
No access in 180 days	Retirement initiated — begin formal retirement procedure below
Source data discontinued	Retirement initiated — product can no longer refresh
VAULTIS-A quality score drops below 70%	Remediate or retire — quality below operational threshold

NOTE

These triggers are governance requirements, not suggestions. Program managers must monitor access metrics and quality scores as part of portfolio health management.

STANDARDS: Retirement is complete when: all active users have been notified and confirmed transition; downstream dependencies have been eliminated or migrated; data has been archived per retention policy; product access has been revoked; retirement is documented in the program record.

EQUIPMENT: MSS Foundry (retiring product workspace, archive workspace), program tracker (dependency map, retirement record), user registry for the retiring product, C2DAO data governance policy (retention schedule), replacement capability access (if applicable).

NOTE: Minimum 90-day retirement process from decision to decommission. Complex products with many downstream dependencies may require 180 days.

PROCEDURE:

1. **Map all dependencies (T-90 days minimum).** Identify every product, pipeline, dashboard, or analytical workflow that consumes data from the retiring product. For each dependency: document the consuming product, the specific data consumed, the operational impact of loss, and the migration path.
2. **Communicate the retirement decision to users (T-90 days).** Issue a formal notification to all identified users. Include: the product being retired, the reason for retirement, the retirement date, the replacement capability (if any) and how to access it, and a point of contact for questions. Communicate through multiple channels: email, unit brief, helpdesk ticket, and the product's Workshop page (add a prominent retirement banner).
3. **Support transition (T-90 to T-30 days).** Conduct transition training for users migrating to a replacement capability. Provide a parallel running period where both the retiring product and the replacement are available simultaneously, if feasible. This reduces the risk of a hard cutover.
4. **Migrate downstream pipelines and products (T-90 to T-30 days).** For each identified downstream dependency, assign a technical lead to execute the migration. Track migration progress in the program tracker. No dependency may be left unresolved at the time of retirement.
5. **Conduct user confirmation (T-30 days).** Contact the data point of contact for each identified user unit. Confirm: they have been briefed on the retirement; they have completed or planned the transition to the replacement; they have no unresolved questions. Document confirmation in writing.
6. **Archive product data (T-14 days).** Determine the data retention requirement for the product's output data. Consult with C2DAO and the unit data leads. For operationally significant historical data, execute an archive to a designated retention workspace per UDRA v1.1 requirements.

7. **Revoke access and decommission (T-0).** Revoke all user access to the retiring product workspace. Disable production pipelines. Archive the Foundry project. Do not delete — decommission by moving to an archive state with read-only access for the program manager.
8. **Document the retirement (T+5 days).** Publish a retirement notice to the program tracker and the product documentation workspace. Include: retirement date, reason, replacement capability, downstream products migrated, data archive location (if applicable), and approving authority.

WARNING: Do not decommission an MSS product before all downstream dependencies have been migrated and confirmed. A pipeline or dashboard that silently loses its input data and continues to serve stale data to operational users is worse than an error message — it creates false confidence in outdated information. Confirm every dependency is resolved before executing the T-0 decommission step.

CAUTION: Data archived at retirement must comply with the applicable records retention schedule and C2DAO data governance policy. Do not archive operationally sensitive data to a workspace with broader access than the original production product. Confirm archive access permissions with C2DAO before executing the archive.

APPENDIX C — SAMPLE PROGRAM INCREMENT OBJECTIVES AND ASSESSMENT

C-1. PURPOSE

This appendix provides example Program Increment Objectives (PIOs) for a USAREUR-AF MSS program, along with an example PI Review assessment. Use these examples to calibrate the quality and specificity expected of PIOs in the USAREUR-AF context.

C-2. EXAMPLE PI OBJECTIVES

PI 4 Program Increment Objectives — MSS USAREUR-AF

PIO 1: Theater Readiness Dashboard — Full Operational Capability Deliver all six readiness metric panels to the USAREUR-AF G3/G4 operational dashboard, enabling daily readiness reporting without manual data compilation by G3 analysts. - Success criteria: All six panels display current data from validated feeds; G3 analyst confirms workflow integration; data latency <4 hours from source. - Owner: Team Alpha (SL 4J: CPT Rodriguez) - Confidence: High (4 of 6 panels complete at PI Planning)

PIO 2: Supply Chain Pipeline — Phase 2 Integration Extend the existing supply chain pipeline to ingest from two additional data sources (forward logistics element feeds), increasing coverage from 60% to 90% of theater units. - Success criteria: Pipeline ingests from both new sources; data quality validation passes for >95% of records; downstream dashboard reflects new coverage. - Owner: Team Bravo (SL 4J: Mr. Harrison) - Confidence: Medium (new data source SLAs not yet confirmed)

PIO 3: Personnel Readiness ML Model — Production Release Promote the personnel readiness prediction model from development to production, enabling G1 staff to generate 30-day readiness outlooks at formation level. - Success criteria: Model passes production readiness gate (SL 4M checklist); G1 chief reviews and accepts model validation results; human review workflow documented and tested with G1 analysts. - Owner: Team Charlie (SL 4J: SFC Washington) - Confidence: Medium (model evaluation 80% complete at PI Planning)

PIO 4: Ontology v2.1 Migration Complete migration of all production products from ontology v2.0 to v2.1, resolving the identified schema fragmentation and enabling cross-product object linking. - Success criteria: All 7 affected pipelines migrated and tested; no production outages during migration window; downstream dashboards validated post-migration. - Owner: Platform Team (SL 5K: Ms. Nakamura) - Confidence: High

PIO 5: PM Capability Development Conduct PM capability assessments for all four SL 4J PMs; develop and begin executing individual development plans. - Success criteria: All four assessments complete; all four development plans documented and agreed; first check-in meetings scheduled. - Owner: Program Manager - Confidence: High

Stretch: Intelligence Analysis Dashboard — Concept and Prototype If capacity remains after PIO 1-5, deliver a concept brief and low-fidelity prototype for the G2 intelligence analysis dashboard. - Success criteria: Concept brief approved by G2 data lead; prototype demonstrates core workflow. - Owner: Team Delta (SL 4J: CW3 Patel) - Confidence: Low (stretch only; no commitment)

C-3. EXAMPLE PI REVIEW ASSESSMENT

PI 4 Review — Assessment of Objectives

PIO	Committed?	Delivered?	% Complete	Assessment
PIO 1: Theater Readiness Dashboard	Yes	Yes	100%	Green — All 6 panels delivered; G3 confirmed workflow integration
PIO 2: Supply Chain Pipeline Phase 2	Yes	Partial	70%	Amber — One of two data sources integrated; second source SLA not signed until PI Week 6; integration in progress

PIO	Committed?	Delivered?	% Complete	Assessment
PIO 3: Personnel Readiness Model — Production	Yes	Yes	100%	Green — Model in production; G1 review gates confirmed
PIO 4: Ontology v2.1 Migration	Yes	Yes	100%	Green — All 7 pipelines migrated; zero production outages
PIO 5: PM Development	Yes	Yes	100%	Green — All four assessments complete; development plans agreed
Stretch: G2 Dashboard Prototype	No (stretch)	No	30%	N/A — Stretch not committed; partial progress noted for next PI

PI 4 Program Predictability: 4 of 5 committed PIOs fully delivered = 80% predictability.

PI 4 Lessons Learned: - External SLA timelines must be confirmed before committing a PIO with external data dependency. PIO 2 amber was foreseeable. Build explicit SLA confirmation milestone into PI Planning for any PIO dependent on an external organization. - Model production release process (PIO 3) worked well. SL 4M checklist and G1 review gate are now standard and should be referenced in next PI's model releases. - Ontology migration (PIO 4) completed ahead of schedule. Credit to Platform Team for early planning and 48-hour pre-migration testing. Adopt platform migration planning process as program standard.

APPENDIX D — PM ONBOARDING GUIDE FOR SL 5J PROGRAMS

D-1. PURPOSE

This appendix provides a structured onboarding guide for a new SL 4J PM joining an active SL 5J-managed program. Use this guide to ensure new PMs have the context, access, and knowledge they need to be productive within their first sprint.

D-2. WEEK 1: ACCESS AND ORIENTATION

Day 1-2: Administrative and Access

- ☐ Confirm CAC-based MSS account creation. Request access to program tracking workspace, team Workshop workspace, and relevant Foundry project spaces through C2DAO.
- ☐ Receive program tracker login and workspace navigation walkthrough from program manager.
- ☐ Review program SITREP archive (last 3 issues) to understand current program status.
- ☐ Receive and read the Program Decision Record (Appendix A of DAM) — understand the key decisions made before you arrived.

Day 3-5: Team and Context

- ☐ Meet with each team lead for a 30-minute context brief: what the team is building, who uses it, current sprint status, top risks.
- ☐ Review the current PI objectives and the dependency register.
- ☐ Review the Program Health Assessment (most recent) to understand overall program health.
- ☐ Attend the weekly Program Sync — observe before participating.

D-3. WEEK 2: STAKEHOLDERS AND PRODUCTS

- ☐ Meet with the C2DAO point of contact for your assigned project area.
- ☐ Review the product documentation for all products your team owns or is building: architecture, ontology, data dictionary, user guide.
- ☐ Attend a sprint review for your assigned team. Review the Definition of Done checklist application.
- ☐ Review the technical debt backlog for your assigned team. Confirm you understand the top three debt items.

D-4. WEEK 3: ACTIVE PARTICIPATION

- ☐ Lead or co-lead your first sprint planning ceremony.
- ☐ Confirm stakeholder contact list for your project area is current.
- ☐ Review the vendor/contractor deliverable schedule for your project area. Confirm acceptance criteria are documented for any pending deliverables.
- ☐ Complete 30-day check-in with program manager. Discuss initial observations and questions.

D-4A. COMMON NEW PM MISTAKES AND HOW TO PREVENT THEM

New SL 4J PMs joining an active program under SL 5J oversight tend to make predictable mistakes. The program manager should brief these proactively — not wait to discover them through sprint failures.

Table D-1. Common New PM Mistakes

Mistake	Why It Happens	Prevention
Accepting scope without checking capacity	Wants to be responsive to stakeholders; not yet comfortable with trade-off conversations	Walk through the capacity allocation policy in week 1; role-play a stakeholder request conversation
Marking features complete without DoD checklist	Under sprint deadline pressure; unclear what the checklist requires	Review DoD checklist together in week 1; require first three releases to be co-reviewed with program manager
Not logging risks until they become blockers	Optimism; not recognizing early warning signs	Review the risk register together weekly for first two sprints; ask "what could prevent each committed feature from being done?"
Letting dependency status go stale	Dependency register feels like overhead; no one is checking it	Program manager reviews the dependency register weekly in Program Sync; hold the PM accountable for currency
Treating metrics as performance scores	Prior experience with organizations where metrics were used punitively	Explicitly brief the metrics-as-tool philosophy in week 1; review a velocity drop together and model the diagnostic process

D-5. 30-DAY COMPLETION CHECKLIST

By the end of 30 days, the incoming SL 4J PM should be able to:

- ☐ Describe the program's current PI objectives and status without referring to notes.
- ☐ Identify the top three cross-team dependencies relevant to their project area.
- ☐ Name the key operational stakeholders for their project area and describe their operational role.
- ☐ Navigate to and interpret the program health dashboard.
- ☐ Lead a sprint planning and sprint review ceremony independently.
- ☐ Locate and understand the technical debt backlog for their team.
- ☐ Explain the change control process for a production change to their product.

CHANGE HISTORY

Version	Date	Description	Authority
1.0	March 2026	Initial publication	USAREUR-AF C2DAO

GOVERNING REFERENCES

Publication	Title	Relevance
Army CIO Memorandum	Data and Analytics Policy (April 2024)	Data governance authority
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

STRATEGIC GUIDANCE

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

Document	Authority	Relevance
UDRA v1.1	Unified Data Reference Architecture (February 2025)	Technical reference architecture
DoD Data Strategy	DoD Data Strategy (2020)	Enterprise data management framework

RELATED PUBLICATIONS

Publication	Title	Relationship
SL 4J	Program Manager (Technical)	Prerequisite; this manual builds directly on SL 4J content
SL 4G	ORSA	Technical track managed by SL 5J; reference for understanding ORSA deliverables

Publication	Title	Relationship
SL 4H	AI Engineer	Technical track managed by SL 5J; reference for AI pipeline delivery standards
SL 4M	ML Engineer	Technical track managed by SL 5J; reference for ML model lifecycle
SL 4K	Knowledge Manager	Technical track managed by SL 5J; reference for ontology governance
SL 4L	Software Engineer	Technical track managed by SL 5J; reference for custom code standards
SL 5G through SL 5O	Advanced technical tracks	Peer publications; reference for advanced capability expectations in each track
Data Literacy Technical Reference	Data Literacy	Doctrinal foundation; establishes organizational data literacy standards
Data Literacy for Senior Leaders	Senior leader data doctrine	Strategic framing for GO/SES briefings on data program value

SL 5J — Advanced Program Manager (Technical) Maven Smart System — USAREUR-AF Operational Data Team Headquarters, United States Army Europe and Africa, Wiesbaden, Germany 2026 This publication implements Army CIO Memorandum, Data and Analytics Policy (April 2024) and aligns to the Unified Data Reference Architecture (UDRA) v1.1 (February 2025).

GLOSSARY

AIP (Artificial Intelligence Platform). Palantir's framework for building AI-assisted workflows on Foundry, including language model integrations and AI-driven application development. SL 5J program managers oversee SL 4H resources who build and maintain AIP-integrated products.

Army CIO Memo (April 2024). Army Chief Information Officer Memorandum on Data and Analytics Policy (April 2024). Establishes Army-wide requirements for data management, AI governance, and data platform standards. All MSS products must comply.

Blocked Time. Sprint capacity consumed by tasks that cannot progress due to unresolved dependencies, missing information, or access issues. Measured as a fraction of total sprint capacity. High blocked time indicates dependency management failure.

Build/Buy/Configure Decision. A structured analysis of how to meet a capability requirement: by writing custom code (Build), using existing platform features with configuration (Configure), or procuring an external capability (Buy). SL 5J program managers govern this decision for all significant investments.

C2DAO (C2 Data and Analytics Office). USAREUR-AF office responsible for MSS governance, data policy, platform oversight, and theater-level data capability management. The C2DAO Chief is the primary internal senior stakeholder for the MSS program.

Cycle Time. The elapsed time from when a feature is started (work begun) to when it is delivered (accepted and released). A key process efficiency metric. Long cycle time indicates work-in-progress accumulation, blockages, or large batch sizes.

Definition of Done (DoD). The minimum quality standards a feature must meet to be considered complete and releasable. Defined in SL 4J Appendix B. SL 5J program managers enforce DoD compliance across all teams.

Dependency Register. The program-level tracking artifact that documents all inter-team, inter-product, and external dependencies — including owner, status, due date, and risk rating. Maintained by the program manager and updated by team leads.

Escaped Defect. A defect (bug, data quality error, incorrect business logic) that reaches production without being caught during development or pre-release testing. Tracked per team per PI. High escaped defect rates indicate insufficient test coverage or Definition of Done compliance.

FDE (Forward Deployed Engineer). A Palantir employee embedded with an operational customer to provide technical expertise, platform architecture guidance, and hands-on development support. Not a government employee; subject to task order terms.

Feature Team. A cross-functional Agile team that owns a product or product area end-to-end, from requirements through delivery and sustainment. The recommended team type for stream-aligned MSS product delivery.

MSS (Maven Smart System). The Palantir Foundry instance deployed for USAREUR-AF and EUCOM operational data management, analytics, and AI capability. The operational platform for all MSS technical tracks.

Ontology. The MSS data model: the structured representation of operational entities (units, personnel, equipment, tasks) as object types with properties and links. Owned by SL 4K Knowledge Managers. SL 5J program managers govern ontology change management at the program level.

PI (Program Increment). A fixed time interval (typically 8-12 weeks) in which a scaled Agile program plans, executes, and reviews a set of committed objectives. The primary planning unit for multi-team MSS programs.

PI Objectives (PIOs). The measurable outcomes a program commits to deliver within a Program Increment. Set at PI Planning; reviewed at PI Review. The primary accountability metric for program-level delivery.

PI Planning. The scaled Agile ceremony where all teams in a program align on objectives, identify dependencies, and commit to sprint plans for the upcoming PI. The most important program management ceremony.

Platform Team. A team that owns shared infrastructure, ontology governance, and data quality standards across multiple feature teams. Enables feature teams to build on a stable foundation without duplicating infrastructure work.

Predictability. The ratio of features committed at PI Planning to features actually delivered by PI Review. A program-level metric for planning accuracy and commitment reliability. Target: >85%.

Program Health Assessment (PHA). A quarterly structured review of program status across seven dimensions: delivery performance, stakeholder health, technical health, team health, vendor/contractor health, portfolio coherence, and policy compliance. Documented in Appendix A.

PWS (Performance Work Statement). The contract document that specifies deliverables, standards, and schedule for contractor performance under a task order. The basis for acceptance criteria development.

SAFe (Scaled Agile Framework). A widely-used framework for applying Agile methods to programs with multiple teams. SL 5J draws on SAFe concepts (PI Planning, team synchronization, ART) without requiring strict SAFe implementation.

Stream-Aligned Team. A feature team aligned to a specific user workflow or operational domain (e.g., readiness, logistics, intelligence). Receives work from a single value stream. Recommended team type for USAREUR-AF MSS programs.

Task Order. A contractual instrument that authorizes and funds a specific set of deliverables from a contractor (e.g., Palantir) against a base contract or vehicle. SL 5J program managers manage task order delivery and conduct formal acceptance reviews.

Technical Debt. The accumulated cost of earlier shortcuts, suboptimal design decisions, or deferred work in a codebase or data product. Classified by type (architectural, data quality, documentation, test coverage, dependency, configuration). Managed as a program-level risk.

Throughput. The number of features (not story points) delivered per PI per team. A counting metric for delivery volume. Useful for cross-team comparisons because it is not affected by estimation calibration differences between teams.

UDRA v1.1 (Unified Data Reference Architecture, version 1.1, February 2025). The Army's technical reference architecture for data management, platform governance, and data product standards. All MSS products must align to UDRA v1.1 requirements for data classification, access management, and lifecycle management.

Velocity. The number of story points a team delivers per sprint, averaged over recent sprints. A planning tool for estimating future capacity. Not a performance metric for individuals. See Table 5-1 and Section 5-3.

Vendor Lock-In. The degree to which a product or capability is dependent on a specific vendor's proprietary features, making future migration costly. Managed through the build/buy/configure decision framework and through designing products against open standards where feasible.

WIP Limit (Work-In-Progress Limit). A constraint on the number of tasks or features that can be actively in progress simultaneously within a sprint or on a team's board. WIP limits reduce context switching and improve cycle time. SL 5J program managers encourage WIP limit discipline across all teams.

DoD and Army Strategic References:

- **DoDI 5000.87, Software Acquisition Pathway (October 2020)** — Establishes the software acquisition pathway for rapid, iterative software delivery
- **Army Directive 2024-02, Agile Software Development (December 2024)** — Army policy for agile software development practices and delivery