TECHNICAL MANUAL

SL 5O

USAREUR-AF

TM-50O — MAVEN SMART SYSTEM (MSS)

Specialist Course Manual

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

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

TM-50O — MAVEN SMART SYSTEM (MSS)

Forward: SL 5O qualifies advanced platform engineers to manage multi-cluster fleets, implement platform reliability engineering (SRE practices), automate RMF/ATO compliance, and design developer experience systems that accelerate application delivery across the MSS portfolio. This manual extends SL 4O from single-cluster operations to enterprise platform leadership. **Prereqs:** SL 4O, Platform Engineer (required, Go evaluation on file); demonstrated operational experience managing MSS infrastructure from SL 4O practical exercise or operational assignment; CONCEPTS_GUIDE_TM50O_PLATFORM_ENGINEER_ADVANCED (read before this manual) *HQ USAREUR-AF · v1.0 · 2026 · DISTRIB: USG only · AUTH: C2DAO/UDRA v1.1*

WARNING: Fleet-level platform decisions at SL 5O level affect every cluster, every application, and every user across the MSS ecosystem simultaneously. A misconfigured fleet-wide policy, a botched cluster upgrade, or a broken cross-domain replication has theater-wide blast radius. Validate changes on canary clusters before fleet-wide rollout. Maintain rollback capability at every level. CAUTION: Fleet management credentials and cross-domain replication keys are the highest-value secrets in the MSS infrastructure. Compromise enables access across classification boundaries. Store, rotate, and audit these credentials with extreme rigor. Report any suspected compromise immediately to ISSM, S6/G6, and C2DAO.

TABLE OF CONTENTS

CHAPTER 1 — INTRODUCTION: THE ADVANCED PLATFORM ENGINEER

ROLE IN MSS

1-1. Advanced Platform Engineer Manual

BLUF: SL 5O extends SL 4O from building and operating individual clusters to managing the fleet — the collection of clusters, environments, and infrastructure services that constitute the MSS platform across the USAREUR-AF AOR.

This manual provides task-based instruction for advanced platform engineers operating on the Maven Smart System (MSS). SL 5O personnel architect multi-cluster topologies, define service level objectives, automate compliance, and engineer the developer experience that enables application teams to deliver at speed.

SL 5O covers multi-cluster fleet management (Cluster API, fleet topology, lifecycle management, upgrade strategies); platform reliability engineering (SLOs, SLIs, error budgets, incident management, capacity planning); RMF/ATO automation (continuous compliance monitoring, automated evidence generation, STIG automation); developer experience engineering (golden paths, self-service portals, internal tooling, developer productivity measurement); cross-domain infrastructure (cross-domain solutions, data diode integration, multi-classification cluster management); and platform observability at scale (distributed tracing, log aggregation, metric federation, alerting strategy).

SL 5O does NOT cover single-cluster Kubernetes fundamentals — see SL 4O; CI/CD pipeline basics — see SL 4O; application-level architecture — see SL 5L; or individual application design — see SL 4N, SL 5N.

1-2. Curriculum Position

Prerequisite: SL 4O (Platform Engineer) is REQUIRED with Go evaluation on file. Demonstrated operational experience managing MSS infrastructure (from SL 4O practical exercise or operational assignment).

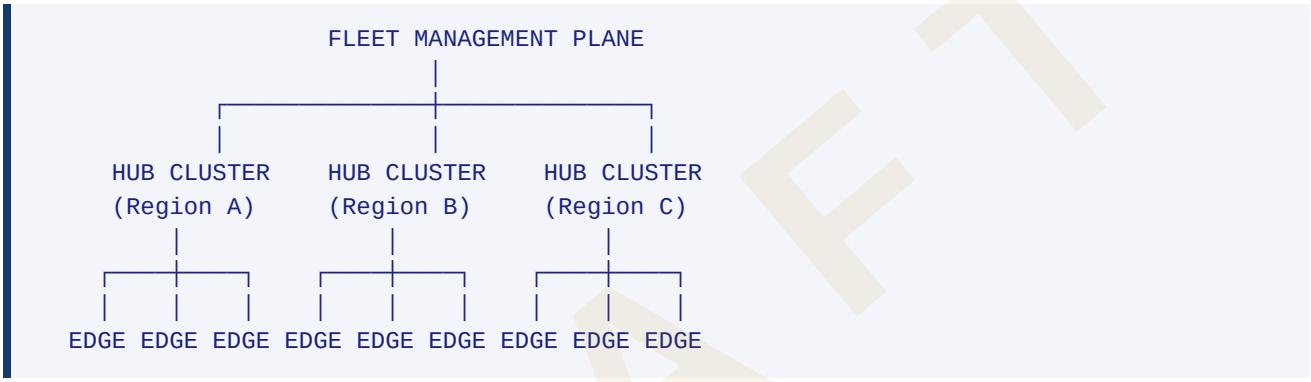
Peer advanced tracks: Coordinate with SL 5L (Advanced SWE) on the platform-application boundary — where does the platform's responsibility end and the application's begin? Coordinate with SL 5J (Advanced PM) on platform roadmap prioritization. Coordinate with SL 5N (Advanced UI/UX) on platform-wide performance budgets and design system hosting.

CHAPTER 2 — MULTI-CLUSTER FLEET MANAGEMENT

2-1. Fleet Topology

BLUF: MSS is not one cluster — it is a fleet of clusters spanning multiple environments, classification levels, and geographic locations across the USAREUR-AF AOR. Managing the fleet requires thinking in systems, not instances.

MSS fleet topology model:



Fleet management responsibilities:

Responsibility	Single Cluster (SL 4O)	Fleet (SL 5O)
Provisioning	Provision one cluster	Define cluster templates; provision fleets declaratively
Upgrades	Upgrade one cluster	Rolling upgrade strategy across fleet; canary cluster validation
Configuration	Configure one cluster	Fleet-wide policy distribution; drift detection across fleet
Monitoring	Monitor one cluster	Federated monitoring; cross-cluster correlation
Security	Secure one cluster	Fleet-wide security posture; cross-domain boundary management

2-2. Cluster Lifecycle Management

BLUF: Clusters are cattle, not pets — the same principle that applies to pods applies to clusters at fleet scale. Provision declaratively, upgrade systematically, decommission cleanly.

Cluster lifecycle stages:

Stage	Action	Validation
Provision	Cluster API creates cluster from template	Smoke tests pass; monitoring online; GitOps controller synced

Stage	Action	Validation
Configure	GitOps applies workload configuration	All expected namespaces, policies, and base services running
Operate	Normal operations; workloads scheduled	SLOs met; no unresolved alerts
Upgrade	Control plane upgrade, then worker nodes	Canary workloads healthy; no degradation during upgrade
Decommission	Drain workloads; archive logs; destroy cluster	All workloads relocated; data preserved per retention policy

2-3. Fleet-Wide Upgrade Strategy

BLUF: Upgrading one cluster is a deployment. Upgrading a fleet is a campaign — plan it like one.

Upgrade wave strategy:

```
Wave 0: Dev/test clusters (break here, not in prod)
    ↓ validate (24h soak)
Wave 1: Canary production cluster (lowest traffic)
    ↓ validate (48h soak)
Wave 2: Regional hub clusters (one at a time)
    ↓ validate (24h soak per cluster)
Wave 3: Edge clusters (batched by region)
    ↓ validate
Wave 4: High-sensitivity clusters (last, most cautious)
```

2-4. Disaster Recovery and Business Continuity (DR/BC)

BLUF: A fleet that cannot recover from catastrophic failure is a single point of failure at theater scale. DR/BC planning at SL 5O level addresses loss of entire clusters, regions, or data centers — not individual pod failures.

RTO/RPO targets by service tier:

Service Tier	RTO (Recovery Time Objective)	RPO (Recovery Point Objective)	Example Services
Tier 1 — Mission-critical	<1 hour	<15 minutes	C2 dashboards, readiness reporting, cross-domain replication
Tier 2 — Operational	<4 hours	<1 hour	CI/CD pipelines, developer portals, data pipelines
Tier 3 — Supporting	<24 hours	<4 hours	Documentation hosting, training environments, sandbox clusters

Backup strategy across clusters:

Backup Target	Method	Frequency	Storage Location
etcd state	Automated encrypted snapshots	Every 6 hours + before upgrades	Off-cluster storage in separate availability zone
Persistent volumes	CSI volume snapshots + cross-region replication	Daily (Tier 1: continuous replication)	Regional hub storage; Tier 1 replicated to secondary region
GitOps configuration	Git repository (inherently backed up)	Every commit	Git server + mirror at hub cluster
Secrets/credentials	Encrypted vault snapshots	Daily	Offline backup + secondary vault instance
Observability data	Metric/log federation to hub	Continuous	Hub cluster long-term storage

Failover procedures:

1. **Cluster-level failover:** When a cluster is lost, the fleet management plane reschedules workloads to surviving clusters based on pre-defined affinity rules and capacity reserves. Capacity reserves: maintain ≥20% headroom per region to absorb failover load.
2. **Region-level failover:** When a regional hub is lost, a pre-designated secondary hub assumes coordination. DNS and load balancer failover routes traffic to surviving regions. Validate cross-region latency is within SLO bounds before declaring failover complete.
3. **Cross-domain failover:** Cross-domain replication failure requires ISSM notification. Failover to a degraded single-domain mode until the cross-domain solution is restored. Do NOT attempt cross-domain workarounds without ISSM approval.

DR testing and exercises:

Exercise Type	Frequency	Scope	Validation Criteria
Tabletop	Quarterly	Walk through DR runbooks; identify gaps	All runbook steps are current and assigned
Component failover	Monthly	Kill a non-production cluster; verify workload migration	Workloads reschedule within RTO; no data loss beyond RPO
Regional failover	Semi-annual	Simulate loss of one regional hub	Traffic reroutes; applications available within Tier 1 RTO
Full DR exercise	Annual	Simulate loss of primary data center	Full fleet recovers within tiered RTOs; lessons learned documented

After each exercise, update runbooks and DR plans within 5 business days. Track findings as action items with owners and due dates.

CHAPTER 3 — PLATFORM RELIABILITY ENGINEERING

3-1. Service Level Objectives (SLOs)

BLUF: "The system is up" is not a reliability statement. SLOs define what "reliable enough" means in measurable, user-facing terms.

SLO framework for MSS platform:

Service	SLI (what to measure)	SLO (target)	Error Budget
Application availability	% of successful HTTP responses (non-5xx)	99.9% (43 min downtime/month)	0.1%
CI/CD pipeline	% of pipelines completing within time limit	99.5%	0.5%
Deployment success	% of deployments completing without rollback	99%	1%
Data freshness	% of datasets updated within SLA window	99.5%	0.5%
API latency	p99 response time for OSDK queries	<2 seconds	Track 95th and 99th percentile

3-2. Error Budgets and Decision Making

BLUF: Error budgets convert reliability from an aspiration into a decision framework. When the budget is healthy, ship features. When the budget is exhausted, stop shipping and fix reliability.

Error budget policy:

Budget Status	Action
>50% remaining	Normal operations; ship features; accept reasonable risk
25–50% remaining	Increased caution; require additional testing for changes; review recent incidents
<25% remaining	Reliability focus; no non-critical changes; dedicate engineering time to reliability improvements

Budget Status	Action
Exhausted	Change freeze (except reliability fixes); incident review required before resuming deployments

3-3. Incident Management

BLUF: Incidents are not failures — they are data. The incident response process determines whether you learn from them or repeat them.

Platform incident response framework:

Phase	Actions	Owner
Detect	Automated alerting triggers; user report received	Monitoring system / on-call
Triage	Assess severity; determine blast radius; assign incident commander	On-call Platform Engineer
Mitigate	Restore service (rollback, failover, scale); communicate status	Incident commander + team
Resolve	Root cause identified; permanent fix applied	Assigned engineer
Review	Blameless post-incident review; document timeline, contributing factors, action items	Incident commander + all involved
Improve	Action items tracked to completion; monitoring/alerting updated; runbooks updated	Platform team

3-4. FinOps — Cloud and Infrastructure Cost Management

BLUF: Compute is not free, and unchecked infrastructure growth consumes budget that could fund new capabilities. FinOps applies financial accountability to infrastructure decisions — measure cost, allocate it to teams, and optimize continuously.

Cost allocation model:

Allocation Method	Mechanism	Use Case
Direct attribution	Resource tags (team, project, environment) on every provisioned resource	Clusters, persistent storage, dedicated node pools
Proportional share	Split shared infrastructure cost by usage metrics (CPU-hours, request count)	Shared platform services, observability stack, ingress controllers

Allocation Method	Mechanism	Use Case
Fixed overhead	Evenly distributed across all consuming teams	Fleet management plane, security tooling, compliance automation

Tag enforcement: Reject resource provisioning requests that lack required cost-allocation tags (`team`, `project`, `environment`). Enforce via admission controller policy.

Showback vs. chargeback:

Model	Description	When to Use
Showback	Report costs to teams for visibility; no budget impact	Starting FinOps practice; teams need awareness before accountability
Chargeback	Deduct infrastructure costs from team budgets	Mature FinOps practice; teams have ability to act on cost data

Start with showback. Move to chargeback only when teams have the tooling and authority to optimize their own resource usage.

Resource right-sizing:

1. **Collect utilization data:** Monitor actual CPU, memory, and storage usage per workload over a minimum 14-day window.
2. **Identify waste:** Flag resources where actual usage is <40% of requested allocation for 14+ consecutive days.
3. **Recommend adjustments:** Generate right-sizing recommendations. Apply automatically in dev/staging; require team approval in production.
4. **Track savings:** Report realized savings monthly. Feed savings data back into capacity planning.

Cost-aware autoscaling:

- Set autoscaler maximum bounds based on budget thresholds, not just resource availability. An autoscaler without a ceiling is a blank check.
- Configure scale-down policies aggressively for non-production environments: scale to zero during non-duty hours where workload allows.
- Use spot/preemptible instances for fault-tolerant batch workloads (data pipelines, CI runners). Maintain on-demand instances only for stateful and latency-sensitive services.
- Alert when projected monthly cost exceeds budget by >10%. Trigger review, not automatic shutdown — operational need may justify the spend.

Cost review cadence:

Review	Frequency	Participants	Output
Cost dashboard review	Weekly	Platform lead	Anomaly identification; spot-check tag compliance
Team cost review	Monthly	Platform lead + team leads	Showback reports; right-sizing actions; optimization targets
Fleet cost forecast	Quarterly	Platform lead + resource manager	Budget forecast; capacity plan; procurement input

CHAPTER 4 — RMF/ATO AUTOMATION

4-1. Continuous Compliance

BLUF: Manual compliance evidence generation does not scale. Automate evidence collection, continuously monitor control effectiveness, and generate ATO artifacts from live system data.

Automation targets by RMF step:

RMF Step	Manual Process	Automated Alternative
Categorize	Document data types and impact levels	Automated asset inventory with classification tagging
Select	Choose security controls	Control baseline auto-applied based on categorization
Implement	Configure controls	IaC templates embed controls; drift detection verifies
Assess	Assessor reviews evidence	Automated scan results, compliance dashboards, evidence export
Authorize	AO reviews package	Automated package assembly from live evidence
Monitor	Periodic manual reviews	Continuous monitoring dashboards; automated alerting on control degradation

4-2. STIG Automation

BLUF: STIGs define hundreds of configuration requirements. Checking them manually is a full-time job. Automate the checks; focus human attention on the exceptions.

STIG automation pipeline:

```
STIG benchmark (DISA) → Policy-as-code (OPA/InSpec/SCAP) → Automated scan → Dashboard
                                                                                  |
                                                                          Pass: evidence
logged
                                                                          Fail: ticket
created
                                                                          Exception: waiver
documented
```

CHAPTER 5 — DEVELOPER EXPERIENCE ENGINEERING

5-1. Golden Paths

BLUF: A golden path is the opinionated, paved road for getting a common task done. It is not the only path — but it is the path that has been tested, secured, and documented. Developers who follow it get fast results with fewer mistakes.

Golden paths for MSS:

Path	Starting Point	End State	Time Target
New application	`create-mss-app` template	Deployed app with CI/CD, monitoring, security scanning	<90 minutes
New pipeline	Pipeline template repo	Running data pipeline with scheduling, alerting, retry logic	<60 minutes
New environment	Self-service portal	Isolated namespace with quotas, network policies, service accounts	<15 minutes
Dependency update	Automated PR from dependency bot	Updated, scanned, tested, and deployed	<30 minutes (automated)

5-2. Developer Productivity Measurement

BLUF: If you do not measure developer productivity, you cannot improve it. But measure the right things — output, not activity.

Recommended metrics (DORA-aligned):

Metric	What It Measures	Target
Deployment frequency	How often code reaches production	Multiple times per day

Metric	What It Measures	Target
Lead time for changes	Time from commit to production deployment	<1 day
Change failure rate	% of deployments causing a failure	<5%
Time to restore service	Time from failure detection to service restoration	<1 hour

5-3. Self-Service Portal Design

BLUF: The self-service portal is the Platform Engineer's product UI. It should follow the same design principles (SL 4N/SL 5N) as any MSS application.

Portal capabilities: - Environment provisioning (create/destroy dev/staging environments) - Pipeline status and logs - Deployment history and rollback controls - Resource usage dashboards (quota consumption, cost allocation) - Documentation and runbook search - Access request workflows - Incident status and communication

CHAPTER 6 — PLATFORM OBSERVABILITY AT SCALE

6-1. Observability Stack Architecture

BLUF: Logs, metrics, and traces are the three pillars of observability. At fleet scale, you need all three, federated across clusters, with correlation capability.

Observability architecture for MSS fleet:

Pillar	Tool Category	Federation Strategy
Metrics	Time-series database (Prometheus-compatible)	Hierarchical federation: edge → hub → fleet aggregator
Logs	Log aggregation (ELK/Loki-compatible)	Ship logs from edge to hub; retain locally for DDIL resilience
Traces	Distributed tracing (OpenTelemetry-compatible)	Trace context propagation across services; central trace store
Alerts	Alert manager with routing	Tiered alerting: edge alerts locally; hub alerts regionally; fleet alerts centrally

6-2. Alerting Strategy

BLUF: Alert on symptoms, not causes. Alert on SLO burn rate, not individual metric thresholds. An alert that fires and is ignored is worse than no alert — it trains the team to ignore alerts.

Alerting principles: 1. **Every alert must be actionable:** If the on-call engineer cannot do something about it, it is not an alert — it is a dashboard metric. 2. **Alert on SLO burn rate:** "Error budget burning 10x faster than normal" is more useful than "CPU at 85%." 3. **Tiered severity:** P1 = pages the on-call. P2 = creates a ticket. P3 = appears on dashboard. No other tiers. 4. **Alert fatigue kills:** Review alert volume monthly. If the team gets >5 non-actionable alerts per week, fix the alerting, not the team.

APPENDIX A — REFERENCES

Reference	Relevance
SL 4O — Platform Engineer	Prerequisite; single-cluster operations
SL 5L — Advanced Software Engineer	Platform-application boundary coordination
SL 5J — Advanced Program Manager	Platform roadmap prioritization
SL 5N — Advanced UI/UX Designer	Platform portal design, performance budgets
NIST SP 800-37 (RMF)	Risk Management Framework
NIST SP 800-53	Security and Privacy Controls
DISA STIGs	Security hardening standards
Google SRE Book	SLO/SLI/error budget framework
DORA Metrics (Accelerate)	Developer productivity measurement
CReATE / Carvel case study	ASF fleet management reference

APPENDIX B — PEER SL 5 CROSS-REFERENCES AND WFF INTEGRATION

Peer SL 5 Publications. Advanced Platform Engineers should coordinate with practitioners in these companion advanced-track publications rather than operating in isolation.

Publication	Track	Coordination Point
SL 5G	Advanced ORSA	Infrastructure for analytical workloads; compute scaling for simulation and optimization pipelines
SL 5H	Advanced AI Engineer	GPU provisioning; AI/ML deployment pipelines; model serving infrastructure; inference endpoint management
SL 5M	Advanced ML Engineer	ML training infrastructure; experiment tracking hosting; model registry; feature store infrastructure
SL 5J	Advanced Program Manager	Platform roadmap prioritization; infrastructure cost reporting for portfolio decisions; capacity planning
SL 5K	Advanced Knowledge Manager	Knowledge system hosting; search infrastructure; content delivery and indexing at scale
SL 5L	Advanced Software Engineer	Platform-application boundary; shared service infrastructure; CI/CD pipeline architecture; deployment tooling
SL 5N	Advanced UI/UX Designer	Platform portal design; performance budgets affecting design patterns; design system hosting and CDN

WFF Operational Consumer Note. Platform infrastructure built by SL 5O engineers supports every application that serves 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). When a cluster goes down, a G2 intelligence dashboard goes dark. When a fleet-wide upgrade introduces a regression, every WFF track is affected simultaneously. When compliance lapses and the ATO is suspended, the entire operational data environment goes offline. The reliability, compliance, and DR/BC questions addressed in SL 5O must be answered in terms of the operational impact on WFF consumers who depend on platform availability to execute their missions.