CONCEPTS GUIDE

SL 4H

CONCEPTS GUIDE — SL 4H COMPANION — AI ENGINEER · MAVEN SMART SYSTEM (MSS)

Specialist Course Manual

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

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

CONCEPTS GUIDE — SL 4H COMPANION — AI ENGINEER · MAVEN SMART SYSTEM (MSS)

Forward: The AI engineer's job is to turn MSS data into AI-augmented workflows that increase decision velocity for operational users. This is not the same job as the ML engineer, the software engineer, or the ORSA. **Purpose:** Develops the mental models required to design, build, and govern AI workflows on MSS effectively. Read before beginning SL 4H task instruction. *HQ USAREUR-AF · v1.0 · 2026 · DISTRIB: USG only*

SECTION 1 — THE AI ENGINEER'S ROLE ON MSS

BLUF: The AI engineer's job is to turn MSS data into AI-augmented workflows that increase decision velocity for operational users. This is not the same job as the ML engineer, the software engineer, or the ORSA.

1-1. Role Distinctions on MSS

Role	Primary Output	Core Platform Tools	Not Responsible For
AI Engineer (SL 4H)	AI-augmented workflows: LLM chains, Agents, grounded reasoning products	AIP Logic, Agent Studio, Code Workspaces	Model training, statistical inference, OSDK app architecture
ML Engineer (SL 4M)	Trained models, validation pipelines, model-backed Object properties	Code Workspaces, inference transforms, Foundry datasets	Prompt design, Agent orchestration, operational workflow UX
Software Engineer (SL 4L)	Custom applications, OSDK integrations, TypeScript UI components	OSDK, TypeScript SDK, Code Workspaces	LLM workflow design, model development, data pipeline authorship

These roles overlap at the edges. On a USAREUR-AF MSS project: the ML engineer builds a readiness prediction model; the AI engineer wraps that prediction in an AIP Logic workflow that synthesizes it with current LOGSTAT data and produces a commander's assessment draft; the software engineer surfaces that draft in a Workshop application with an approval widget. Each role has a lane.

The AI engineer's lane is the **reasoning layer**: data retrieval from the Ontology, routing through an LLM, and producing a human-legible product for review.

1-2. What the AI Engineer Produces

The primary deliverable is a **workflow** — a defined sequence of data retrieval, context assembly, LLM inference, and output handling that produces a useful product for an operational consumer. On MSS, that workflow lives in AIP Logic or Agent Studio.

The measure of effectiveness for an AI workflow is not "does the LLM return something?" — it is "does this increase decision velocity without increasing decision error?"

1-3. Boundary: AI Engineering vs. ML Engineering

ML engineers teach the model. AI engineers use the model.

Task	Belongs To
Selecting features, splitting datasets, tuning hyperparameters, validating model performance	ML Engineering
Deciding what data to put in a prompt, structuring a chain, specifying output format, designing human review	AI Engineering
Model-backed Object property feeding an AIP Logic workflow	Both coordinate at the interface; AI engineer consumes outputs, ML engineer owns model quality

NOTE — ADP 3-13 AI/ML Doctrine: TM-40H Section 1-3 now includes an ADP 3-13 (Information) doctrine note placing AI/ML capabilities within the Army's information advantage framework. Review before designing any AI workflow that processes or generates information products.

NOTE — PED-to-AI/ML Pipeline Mapping (TM-40H Section 1-4): SL 4H maps the Processing, Exploitation, and Dissemination (PED) cycle to AIP Logic pipeline stages. Reference Section 1-4 for the standard PED-to-AI/ML workflow mapping when designing intelligence-domain AI workflows.

1-4. AIP Logic vs. Agent Studio vs. Traditional ML

Tool	Use Case	Nature
AIP Logic	Structured, repeatable LLM workflow chains: inputs → prompt → inference → output handlers	Deterministic pipeline
Agent Studio	AIP Logic extended with memory, multi-step reasoning, and tool access; appropriate when the answer depends on what the first query returns	Dynamic, LLM-directed

Tool	Use Case	Nature
Traditional ML	Classification, regression, forecasting — model artifacts	Not LLM-based; AI engineer may consume outputs but does not build them

SECTION 2 — HOW LLMS ACTUALLY WORK — WHAT YOU MUST UNDERSTAND

BLUF: You do not need to understand backpropagation to be an effective AI engineer. You must understand tokens, context windows, temperature, and the fundamental nature of LLM output. Gaps in this understanding produce unsafe workflows.

2-1. The Conceptual Minimum

A large language model predicts the most probable next token given a sequence of preceding tokens. A token is roughly a word fragment. The model generates a response by sampling from a probability distribution of next tokens until it reaches a stop condition.

Three non-obvious implications for operational AI engineering:

Implication	Practical Consequence
The same prompt can produce different outputs	Use low temperature for operational workflows; even at temperature 0, some models are not fully deterministic
The model is not a database	The model does not "know" current unit readiness. It knows statistical patterns from training data. Without grounding in retrieved data, it generates plausible-sounding answers that are frequently wrong
More context tokens ≠ better output	LLMs lose focus on information in the middle of long context windows. Relevance filtering before context assembly is required practice

2-2. Why LLMs Hallucinate and What That Means for Operational Use

"Hallucination" — confidently stated but factually incorrect output — happens because the model's objective is to generate fluent, plausible text, not to verify facts.

The operational risk is not that a hallucinated output will fool an expert analyst. The risk is that it will be accepted by a consumer who does not know the data — a junior planner reviewing a pre-populated SITREP draft, a logistics officer seeing a generated demand forecast.

The structural mitigation is **grounding**: every factual claim in an AI-generated product must be traceable to a specific retrieved data value from MSS. Better prompts reduce hallucination frequency; they do not eliminate it.

2-3. Knowing vs. Being Right

LLMs do not flag uncertainty the way a human expert would. A human analyst says "I'm not sure, let me check." An LLM generates text with the same syntactic confidence regardless of whether it draws on dense training data or thin analogy.

Never evaluate LLM output based on how confident it sounds. Evaluate it based on whether factual claims are grounded in retrieved data you can verify. Ungrounded claims are hypotheses, not findings — and should be marked accordingly in the output format.

For USAREUR-AF workflows: AI-generated products must structurally distinguish between **retrieved facts** (verifiable, source-cited) and **synthesized assessments** (LLM-generated, require human review). Merging these categories into undifferentiated text is a workflow design failure.

SECTION 3 — RAG AND GROUNDING — THE OPERATIONAL AI PATTERN

BLUF: Retrieval-Augmented Generation (RAG) is the primary pattern for operational AI on MSS. Ground LLMs in Ontology data. Do not rely on model knowledge for operational facts.

3-1. What RAG Is

RAG is a workflow pattern: before calling the LLM, retrieve relevant data from a structured source and include it in the prompt context. The LLM generates its response based on retrieved data, not (or not primarily) on training knowledge.

On MSS, the retrieval source is the Foundry Ontology — the authoritative, continuously-updated model of USAREUR-AF operational data.

Simple RAG workflow for a readiness assessment: 1. Identify the unit Object(s) relevant to the query 2. Retrieve relevant Object properties: equipment readiness rates, personnel fill, maintenance backlog, supply status 3. Assemble properties into a structured context block 4. Pass the context block and user's question to the LLM with instructions to answer based on the provided data 5. Return the response for human review

Without steps 2–3, the LLM answers from training data — which contains no information about current unit status.

3-2. The Context Window as a Design Decision

The context window is not a dump zone. Every token costs inference time, potentially costs money, and may degrade response quality by diluting relevance.

Context assembly requires answering: - **What does the LLM need to answer this question?** Not everything on the Object — the specific properties relevant to the query. - **What format communicates the data most efficiently?** Structured tables are generally more token-efficient than prose for multiple data points. - **What can be filtered out?** Historical data not relevant to the current assessment, unrelated properties, metadata fields the LLM will not use.

3-3. Context Breadth vs. Response Quality

Scenario	Context Breadth	Effect on Response
Narrow, targeted query	5–10 highly relevant properties	High precision, low hallucination risk, fast inference
Broad assessment query	30–50 properties, mixed relevance	More comprehensive but higher noise, increased hallucination risk, slower
Full record dump	Entire Object serialized	LLM may miss critical items buried in middle of context; no quality benefit over targeted approach

For a broad assessment requirement: decompose into sub-queries, each with a targeted context assembly, and synthesize sub-results in a final step. This is Agent territory (Section 5).

3-4. Grounding as a Quality Guarantee

A workflow is "grounded" when every factual claim in its output can be directly traced to a retrieved value from the Ontology. Grounding is a structural property you either design in or leave out.

Grounding mechanisms: - Require the LLM to cite source Object and property for each factual claim (output format enforcement) - Post-process the LLM output to verify cited values against retrieved data (validation transform) - Display retrieved source values alongside LLM-generated narrative so the reviewer can spot discrepancies

An ungrounded workflow produces a product that looks authoritative but cannot be verified. In a USAREUR-AF operational context, that product is a liability.

SECTION 4 — DECOMPOSING AN OPERATIONAL AI USE CASE

BLUF: Before writing a single prompt, fully decompose the use case. Answer all five questions in writing before development begins.

4-1. The Five-Question Framework

Question	Purpose	Example
Q1: What is the user trying to do?	State the operational task in user terms — not AI engineering terms	"The S4 needs to identify units with critical supply shortfalls before a major exercise"
Q2: What information do they need?	List specific data elements required — be precise	"Supply status by class, by unit, for units assigned to exercise IRON RESOLVE, vs. the 30-day sustainment standard"
Q3: Which information can be retrieved from MSS?	Map each required data element to a specific Ontology Object Type and property; identify gaps	If a required element does not exist in the Ontology, the workflow cannot provide it — tell the user explicitly
Q4: What does the LLM add on top of retrieved data?	Identify the specific AI value-add	Legitimate: synthesizes multiple data streams into narrative prose; drafts formatted products; identifies patterns across many Objects. Illegitimate: "fills in information we don't have" (hallucination risk); "makes the assessment so the analyst doesn't have to" (unauthorized removal of human judgment)
Q5: What is the human review/approval step?	Document who reviews, what they review against, what action approves/rejects/modifies, and what prevents output from flowing downstream without that review	This question is answered in the workflow design before any other design decision

4-2. Vignette: LOGSTAT Synthesis for Exercise Planning

A USAREUR-AF G4 planner needs to produce a pre-exercise logistics assessment for a brigade-level exercise involving V Corps, 21st TSC support elements, and 10th AAMDC air defense coverage. Current process: manually review LOGSTAT data across 8 subordinate battalions and 3 AAMDC batteries, compare against sustainment standards, draft a summary paragraph for the EXORD annex. Takes 4–6 hours.

Question	Answer
Q1: What is the user doing?	Producing a logistics readiness assessment paragraph for EXORD annex
Q2: What information needed?	Supply status by class (III, V, IX) by battalion; maintenance readiness; days-of-supply on hand vs. standard
Q3: What is in MSS?	Supply Objects with class/battalion/quantity; equipment Objects with maintenance status — all required data present
Q4: What does LLM add?	Synthesizes 8 battalion data sets into coherent prose paragraph; flags units below standard; formats to EXORD template
Q5: Human review step?	G4 planner reviews draft against source LOGSTAT data; approves by clicking "Accept" in Workshop; output held in draft state until approval recorded

This decomposition reveals: the workflow is feasible with full MSS grounding; the LLM's role is synthesis and formatting, not assessment; the human review step is the G4 planner, not an automated gate.

SECTION 5 — AGENT DESIGN MENTAL MODEL

BLUF: An Agent is an LLM with access to tools and memory. Use Agents when a task requires sequential decisions based on intermediate results. Do not use Agents when a single LLM call is sufficient. Over-agentic design produces unpredictable, hard-to-audit workflows.

5-1. What an Agent Is

An Agent is a control loop: the LLM is given a task, decides what tool to call, calls it, receives the result, decides what to do next, and continues until it determines the task is complete or requires human input.

Available tools on MSS: Ontology search, property lookup, Action execution (write back to Ontology — with governance gates), document retrieval, custom Python functions registered as tools.

The critical point: the Agent decides which tools to call and in what order. You, the AI engineer, control the available tools, the system prompt, and the termination conditions — not the exact execution path.

5-2. Single Call vs. Multi-Step Agent

Use Case	Single Call	Agent
Format structured data as prose	Yes	No
Answer a question about known Objects	Yes (with RAG)	No

Use Case	Single Call	Agent
Synthesize a SITREP from a defined data set	Yes	No
Investigate an anomaly requiring sequential lookups	No	Yes
Complete a task where the next step depends on the previous result	No	Yes
Automate a multi-step workflow with decision points	No	Yes

Default is a single LLM call. Upgrade to Agent only when the task genuinely requires it. Every Agent is more complex to test, audit, and explain to a commander than every single-call workflow.

5-3. Agent Failure Characteristics

An Agent workflow is a decision tree where you do not know in advance which branches will be taken. This has implications:

- You cannot enumerate all possible execution paths for testing — only common paths and adversarial inputs
- The Agent can get stuck in loops without explicit step limits and loop detection
- The Agent's reasoning is not transparent — step-by-step decisions emerge from LLM inference and are not introspectable in the traditional engineering sense

Design implications: Set hard step limits (maximum N tool calls); log every tool call and result; define explicit termination conditions; test failure cases first.

5-4. Over-Agentic Design — The Primary Failure Mode

Over-agentic design manifests as: - Using an Agent for a task accomplishable with a single call plus RAG - Building Agents with access to more tools than the task requires - Allowing an Agent to make write operations without a human checkpoint at each write - Treating Agent completion as task completion — without reviewing what tools were called and why - Building Agent complexity because it is impressive in a demo, not because the task requires it

In a USAREUR-AF operational context, over-agentic design is a safety issue. An Agent with unnecessary tool access and no step logging can modify operational data in ways that are difficult to trace, explain, or reverse.

Design heuristic: If you cannot explain to the commander what the Agent will do on every input type, the Agent is not ready for operational deployment.

SECTION 6 — HUMAN-IN-THE-LOOP AS A DESIGN CONSTRAINT, NOT AN ADD-ON

BLUF: In USAREUR-AF, human review before action is a doctrinal and legal requirement. Design the human review step first, then design the AI workflow around it.

6-1. The Doctrinal Requirement

Army CIO Policy (April 2024) requires human oversight of AI-assisted operational decisions. USAREUR-AF C2DAO governance requirements apply to all AIP Logic workflows that process operational data or produce products informing command decisions.

The requirement is not "a human sees the output." It is: **a qualified human with the information needed to evaluate the output reviews it and makes an affirmative decision to accept, reject, or modify it before it flows into an operational product or triggers an action.**

6-2. Meaningful Review vs. Rubber-Stamping

Meaningful Review	Rubber-Stamping
Output is clearly structured: assessments vs. retrieved facts are visually distinct	Output is undifferentiated LLM-generated prose with no source citations
Source data for retrieved facts is displayed alongside LLM output	Review interface is a single "Submit" button with no option to modify
Reviewer has enough domain knowledge to evaluate whether the LLM assessment is plausible	Reviewer is under time pressure making careful evaluation impractical
Review interface provides explicit accept/reject/modify options	Output sounds technically correct enough that a non-expert has no basis to question it
Time allotted is sufficient to actually read and evaluate the output	Workflow is designed to be reviewed in 30 seconds; output requires 5 minutes to evaluate properly

Rubber-stamping is worse than no human review. It creates a false accountability record — a log that says "human approved" when no meaningful evaluation occurred.

6-3. Design Patterns for Meaningful Review

Pattern	Function	Why It Matters
Source citation in output	Every factual claim links to the Ontology Object/property it came from	Reviewer can verify claims against source in the same interface
Side-by-side display	Retrieved data displayed alongside LLM synthesis	Reviewer sees what the LLM saw; can spot discrepancies
Explicit assessment tagging	LLM assessments labeled as AI-generated	Reviewer knows what requires their judgment vs. what is factual retrieval
Modification field	Reviewer can edit AI output before approving	Promotes active engagement; output reflects reviewer judgment
Audit log	Accept/reject/modify logged with reviewer identity and timestamp	Accountability record is meaningful, not just a click log
Step review for Agents	Each Agent action requires review before next step executes	Prevents Agent from compounding an early error

6-4. The Approval Chain as a Workflow Design Element

For operational products, the human review step is part of the staffing process. A G4 planner reviews a logistics assessment draft; the G4 approves and forwards to the DCofS; the DCofS includes it in the EXORD package. That approval chain must be reflected in the Workshop application workflow.

AI engineering that treats the approval chain as a formality to be streamlined has misunderstood its operational context. The approval chain exists because the product will inform command decisions.

SECTION 7 — PROMPT ENGINEERING AS SYSTEMS ENGINEERING

BLUF: A prompt is system configuration. Version-control it, test it against edge cases, document its design intent, and plan for failure modes.

7-1. Prompts Are System Configuration

In a consumer AI context, prompts are conversational. In an operational AI engineering context, prompts are configuration files governing the behavior of a component in a live operational pipeline. Apply the same standards as any system configuration:

- **Version control.** Every production prompt has a version, a change log, and a review history

- **Testing.** Prompts are tested against representative input sets including edge cases and adversarial inputs before deployment
- **Documentation.** Every prompt has documented design intent: what it produces, what inputs it handles, and what it explicitly does not handle
- **Change management.** Prompt changes go through the same review process as code changes

7-2. Prompt Anatomy for Operational Workflows

Component	Purpose	Example
Role definition	Establishes the LLM's operational context and constraints	"You are a logistics analysis assistant supporting USAREUR-AF G4 planning. You do not make recommendations outside the data provided. You do not speculate about future conditions unless explicitly asked."
Task specification	States precisely what the LLM is to produce — include output format, length, and structure requirements	"Produce a two-paragraph logistics readiness assessment. Paragraph 1: supply status by class, flagging any class below 80% on-hand. Paragraph 2: maintenance readiness summary, flagging any category below 85% operational. Use the USAREUR-AF LOGSTAT paragraph format."
Data block	Retrieved Ontology data, formatted for LLM consumption. Structured tables are more token-efficient than serialized Object JSON. Dynamically populated at runtime — not static.	Table of unit supply status by class and battalion
Constraint block	Explicit statements of what the LLM must not do	"Do not include information not present in the data block. Do not make recommendations regarding personnel actions. If the data block is incomplete for any unit, flag the gap explicitly — do not estimate missing values."

7-3. Required Test Categories

Test Category	What to Test	Why
Nominal inputs	Representative operational data within expected ranges	Establish baseline performance
Missing data	Data block with one or more required fields absent	LLM must flag gaps, not estimate

Test Category	What to Test	Why
Boundary values	Data at threshold values (exactly 80% readiness, exactly 30 days supply)	Ensure thresholds produce correct flags
Contradictory data	Data block with internally inconsistent values	LLM must flag, not resolve arbitrarily
Adversarial inputs	Data block with values designed to elicit hallucination	Test grounding enforcement
Off-task inputs	User query outside the workflow's scope	LLM must decline gracefully, not improvise

Document test results. If a prompt fails a test category, fix and retest — do not accept a known failure mode and deploy.

7-4. The Operational Risk of a 95% Prompt

A SITREP synthesis workflow accurate 95% of the time produces 1 error per week in garrison (20 SITREPs/week) — and 5 errors in 72 hours during a high-tempo exercise processing 100 SITREPs.

Design responses: 1. **Know your failure modes.** Testing reveals which input types produce errors. Document them. 2. **Surface uncertainty.** When the workflow detects a condition historically producing errors (missing data, contradictory values, off-nominal inputs), flag it explicitly rather than generating a normal-looking but unreliable product. 3. **Grade the output.** A well-designed workflow does not produce identical-looking outputs for high-confidence and low-confidence cases. Reviewers should be able to see at a glance whether an output requires cursory review or careful scrutiny.

SECTION 8 — MSS-SPECIFIC AI ENGINEERING MENTAL MODELS

BLUF: AIP Logic and Agent Studio operate within the Foundry Ontology. Understanding the Ontology relationship is the foundation of every design decision.

8-1. AIP Logic and the Foundry Ontology

Inputs come from the Ontology. Object properties, filtered Object sets, linked Objects — these are the data sources. The quality of your workflow's output is bounded by the quality of the Ontology data feeding it. Garbage-in-garbage-out applies to LLM workflows exactly as it applies to dashboards.

Outputs can write back to the Ontology. Through Actions, an AIP Logic workflow can update Object properties, create new Objects, or trigger downstream pipeline events. A misconfigured or malfunctioning workflow that writes to the Ontology does not just produce bad output — it corrupts the authoritative data every other MSS consumer relies on.

The Ontology model constrains what is possible. If the Ontology does not have a property for "AI assessment confidence level," you cannot write confidence metadata back to the Object without adding the property through governance. Never hack around Ontology constraints by stuffing structured data into free-text fields.

8-2. Agent Memory in a Multi-User Operational Context

Agent memory in a multi-user operational environment requires answering:

Question	Implication
Whose memory is this?	User-level (specific to one analyst's session) or workflow-level (shared across all users)?
What goes in memory?	Prior queries, retrieved data, interim assessments, user preferences — each has different sensitivity and retention implications
How long is memory retained?	Session-level, exercise-level, or persistent?
Who can see memory content?	Memory may need to respect the same access restrictions as the source data

Failing to answer these questions before deploying an Agent in a multi-user environment can result in one user's session context influencing another user's outputs — a subtle data spillage problem.

8-3. C2DAO Review as the Quality Gate

C2DAO review for AI workflows is the governance gate preventing a workflow from corrupting operational data or producing unauthorized AI-assisted decisions at scale.

The review evaluates: - Does the workflow write back to the Ontology? What does it write, to which Objects, under what conditions? - What is the human review step? Is it meaningful or rubber-stamping? - Is the workflow grounded? Are all factual claims traceable to retrieved Ontology data? - What are the documented failure modes? Has the workflow been tested against them? - What is the rollback procedure if the workflow produces incorrect outputs at scale?

A workflow that cannot answer these questions clearly is not ready for C2DAO review. Think of it as the pre-deployment checklist every pilot completes — skipping it is not faster, it is the failure mode.

NOTE — UDRA VAULTIS-A Governance for ML Outputs: SL 4H now references the UDRA VAULTIS-A (Visible, Accessible, Understandable, Linked, Trustworthy, Interoperable, Secure, Auditable) governance standard for AI/ML outputs consumed by AIP Logic workflows. Ensure all model-backed properties feeding AI workflows meet VAULTIS-A criteria before production deployment.

8-4. Governance Before Activation — The Required Sequence

1. Decompose the use case (Section 4) and document it
2. Design the human review step (Section 6) and document it
3. Build the workflow in a development environment against synthetic or development-tier Ontology data
4. Test against the required test categories (Section 7-3)
5. Document failure modes, confidence levels, and known limitations
6. Submit for C2DAO review with the workflow documentation
7. Receive C2DAO authorization
8. Deploy to production
9. Monitor output quality post-deployment; report anomalies to C2DAO

Any sequence that moves step 7 later is a path to deploying an ungoverned workflow — not authorized.

SECTION 9 — COMMON AI ENGINEERING FAILURE MODES

BLUF: Most AI engineering failures are design failures. They happen before a line of code is written.

Failure Mode	Description	Prevention
Over-Promising Capability	Scoping the workflow as if the LLM can do everything the user might want, without grounding scope in what MSS data actually supports	Commit to what the workflow does; measure time savings after deployment — never before
Under-Documenting Scope	Building a workflow without documenting what it does not do, leading users to apply it outside its design envelope	Document: what input types it handles; what it does not handle; what happens when data is missing or stale; what the output does not include
Building Without Defining Correct Output	Starting development without a precise definition of what correct output looks like	Define correct output in operational terms before building: "a paragraph of 100–150 words, flagging all units below 80%, formatted per USAREUR-AF LOGSTAT standard, with source Object IDs cited for each flagged unit"

Failure Mode	Description	Prevention
Deploying Without Human-Review Gates	Any production AIP Logic workflow deployed without a documented and enforced human review step	Enforce the gate in workflow logic — a "Review" button that can be bypassed is not a review gate
Confusing a Demo With a System	Treating a demo that works once — with curated data, in development — as evidence a workflow works reliably	Close the gap with a comprehensive test suite against operational data samples, staged rollout, monitored production period, and a tested rollback plan

SUMMARY: THE AI ENGINEER'S DESIGN CHECKLIST

Before submitting any AIP Logic workflow for C2DAO review, confirm:

Item	Confirmed
Use case decomposed through five-question framework (Section 4)	
Human review step defined: who, what, how, enforcement mechanism (Section 6)	
All factual claims grounded in retrieved Ontology data (Section 3)	
Prompt tested against nominal, edge, adversarial, and off-task inputs (Section 7)	
Failure modes documented	
Agent step limit and loop-detection configured, if applicable (Section 5)	
Output distinguishes retrieved facts from LLM-generated assessments (Section 2)	
Scope boundaries documented (Section 9)	
Rollback procedure defined and tested	
C2DAO authorization obtained before production deployment (Section 8)	

TRANSITION TO SL 4H

This guide establishes the mental models applied throughout SL 4H. SL 4H task instruction assumes this conceptual foundation — it will not re-explain why RAG grounding matters, why human review gates are enforced in workflow logic, or why prompts require version control.

Proceed to TM-40H, Chapter 1, when you can answer the following without referencing this guide:

1. What is the difference between what an AI engineer builds and what an ML engineer builds on MSS?
2. Why does grounding LLM output in Ontology data reduce hallucination risk?
3. What five questions must be answered before developing an AI use case?
4. What is the difference between meaningful human review and rubber-stamping?
5. Why is a prompt treated as system configuration, not casual instruction?

If any answer requires looking it up, re-read the relevant section before proceeding.

GOVERNING REFERENCES (ADDITIONAL)

Document	Relevance
Army DIR 2024-03	Digital Engineering Policy — Army-wide digital engineering adoption directive
FM 3-12	Cyberspace Operations and Electromagnetic Warfare — operational context for AI systems in Army networks
DA PAM 25-2-5	Software Assurance — software security and assurance standards

CURRICULUM NOTES

Prerequisite: SL 3 (Advanced Builder) is REQUIRED before beginning SL 4H or this guide.

Advanced track: SL 4H graduates should pursue **SL 5H (Advanced AI Engineer)** as the next step in the specialist progression. SL 5H addresses multi-agent orchestration at scale, fine-tuning integration, production AI system governance, and AI safety leadership for USAREUR-AF.

Peer specialist cross-references: - **SL 4M (ML Engineer):** The boundary is explicit — ML engineers build and own the model; AI engineers consume its outputs in AIP Logic orchestration. Coordinate at the model/workflow interface before any production deployment. - **SL 4L (Software Engineer):** AI workflow outputs delivered through OSDK applications require SWE implementation. Coordinate on application architecture and CBAC compliance. - **SL 4G (ORSA):** AI workflows that automate ORSA-style synthesis still require ORSA methodology review. An AI-generated analysis is not a decision product until a qualified ORSA validates the analytical approach.

NOTE — New Doctrine Content in SL 4H: SL 4H now includes an ADP 3-13 doctrine grounding for AI/ML (the first ADP to reference AI explicitly), a PED-to-AI/ML pipeline mapping table (section 1-4, FM 2-0), a UDRA VAULTIS-A governance NOTE establishing that ML model outputs are governed data products, and a Strategic Guidance subsection (1-5a) with DDOF Playbook v2.2, Army Data Plan, and DoD Responsible AI Strategy references. These sections connect the technical concepts in this guide to their doctrinal authorities.

WFF awareness: WFF-qualified users (SL 4A through SL 4F — Intelligence, Fires, Movement and Maneuver, Sustainment, Protection, and Mission Command) are the primary consumers of AI-augmented workflows. Every AI workflow must be designed for the specific WFF operational context: an Intelligence (SL 4A) assessment workflow has different human-review requirements than a Sustainment (SL 4D) LOGSTAT synthesis. Match the human-in-the-loop design to the WFF staff section's product authority and decision timeline.

CONCEPTS GUIDE — SL 4H COMPANION, AI ENGINEER, MAVEN SMART SYSTEM (MSS) Headquarters, United States Army Europe and Africa, Wiesbaden, Germany, 2026 Distribution restriction: Distribution authorized to U.S. Government agencies and their contractors only. Other requests must be referred to Headquarters, C2DAO, Wiesbaden, Germany.