PUBLICATION

EXAM-TM40L-POST

POST-TEST — SL 4L: SOFTWARE ENGINEER

Maven Smart System (MSS) — USAREUR-AF

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

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

POST-TEST — SL 4L: SOFTWARE ENGINEER

MAVEN SMART SYSTEM (MSS) — USAREUR-AF

Field	Detail
Course	SL 4L: Software Engineer
Form	Post-Test
Level	SL 4L (Specialist)
Audience	Software engineers; prerequisite: SL 1+20+30 + TypeScript/Python + REST API familiarity
Time Allowed	45 minutes
Passing Score	70% (46/66)

INSTRUCTIONS

This assessment evaluates mastery of course learning objectives. A passing score of 70% is required to receive credit. Complete independently without reference to training materials.

SECTION 1 — MULTIPLE CHOICE

Circle the letter of the best answer. (2 points each)

1. In OSDK authentication architecture, the three components required to initialize a Foundry client are:

A. Username, password, and domain B. OAuth2 client ID, client secret, and redirect URI C. API key, project ID, and namespace D. Client instance, authentication token, and generated type definitions

2. When querying an Object Type via OSDK using `ResourceIterator`, you MUST:

A. Retrieve only the first page of results — additional pages require a separate API call pattern B. Set a `limit` parameter equal to the total expected record count to retrieve all records in one call C. Iterate through all pages until the iterator is exhausted to ensure all records are retrieved D. Use `Promise.all` to fetch all pages concurrently for performance

3. A `ResourceIterator` that is only partially consumed (e.g., `break` after the first 10 records) will:

A. Cause a memory leak on the Foundry server B. Retrieve only the first 10 records — subsequent records from that query are not fetched C. Buffer all remaining records in memory before the break takes effect D. Throw an unhandled exception when the connection is closed

4. After executing an Action via OSDK, the correct pattern to confirm completion is:

A. Retrieve the task ID from the Action response and poll the task status endpoint until the status is "SUCCEEDED" or "FAILED" B. Assume completion after a 5-second `setTimeout` delay C. Listen for a WebSocket event on the affected Object Type for the change D. Re-query the Object and check whether the expected property change is present

5. When traversing a Link Type via OSDK to get related objects (e.g., from a Vehicle to its MaintenanceRecords), the correct approach is:

A. Make a separate raw HTTP call to the Foundry datasets API to fetch the related records B. Re-query the MaintenanceRecord Object Type with a filter on the vehicle ID C. Use the OSDK Link traversal method on the object instance to navigate the relationship within the Ontology D. Export both Object Types to arrays and perform an in-memory join in TypeScript

6. A TypeScript Function on Objects (FOO) that needs to compute a derived property for 500 Vehicle objects should use which pattern?

A. A per-object loop that calls the Ontology API once per vehicle (N+1 calls) B. A `Promise.all` of 500 concurrent individual object API calls C. A bulk query that fetches all required related records in a single operation before computing derived values D. A cached query that runs once daily and stores results in a global variable

7. A TypeScript Action validator must have at minimum how many distinct validation conditions?

A. Three distinct conditions, including at least one cross-field validation B. Two — one for required fields and one for format C. One — the primary validation check D. Five — one per expected failure mode

8. In a Slate application, after the user executes an Action (submits a form), the correct pattern for updating the displayed data is:

A. Set a 3-second timer and re-fetch data after the delay B. Manually reload the entire application window after the Action completes C. Redirect the user to a loading page and back after 2 seconds D. Update a state variable based on the Action's completion event, triggering a reactive data refresh in the components bound to that state

9. When implementing WebSocket subscriptions to Ontology Object change notifications via OSDK, an important consideration is:

A. WebSocket subscriptions are rate-limited to one per project B. Subscription handlers must be idempotent — the same change event may be delivered more than once and the handler must handle duplicates correctly C. WebSocket connections automatically reconnect after any network interruption D. Subscription data is encrypted end-to-end and cannot be logged for debugging

10. The C2DAO code review and deployment checklist requires which of the following credential-handling verifications?

A. No credentials are hardcoded in source code; all credentials are injected via environment variables or secrets management at runtime B. Credentials are stored in a `.env` file committed to the repository with appropriate `.gitignore` rules C. Credentials are rotated at least quarterly regardless of deployment frequency D. A separate code review is conducted by the vendor for credential-related changes

11. A TypeScript Action validator that rejects a vehicle mileage update when the new mileage is less than the current mileage is an example of:

A. Cross-field validation B. A single-field range validation C. A required-field check D. A format validation

12. A Platform SDK write transaction for committing dataset changes is preferred over a simple write because:

A. Transactions bypass schema validation for faster writes B. Transactions encrypt data at rest as part of the write operation C. Transactions are atomic — either all records are written successfully or none are, preventing partial writes that could corrupt downstream consumers D. Transactions automatically trigger dependent pipeline refreshes

13. The minimum number of test cases required for a TypeScript Action validator under SL 4L standards is:

A. 3 — one per validation condition B. 5 — standard unit test coverage minimum C. 15 — comprehensive coverage for production-grade code D. 8 — sufficient to cover each validation condition with both passing and failing cases

14. In a Foundry multi-tenant environment, why is hardcoding a tenant ID never acceptable — even in a test context?

A. Tenant IDs are considered PII and cannot be stored in code B. Hardcoded tenant IDs can cause cross-tenant data access in a different deployment context, violating data isolation requirements C. Tenant IDs change quarterly and hardcoded values will break on rotation D. It is acceptable in test contexts — the restriction applies only to production code

15. A C2DAO input sanitization requirement for a Slate application that accepts a unit designation string from users means you must:

A. Strip or reject characters that could enable injection attacks, validate length bounds, and ensure the value matches expected format patterns before using it in any query or write operation B. Validate that the string contains only ASCII characters C. Convert the string to uppercase before processing D. Verify the unit designation exists in the Ontology before accepting the input

16. Per the Army Data Plan, Strategic Objective 7 (SO 7) directs the Army to build a "Cloud, Data, DevSecOps Enabled Workforce." When mapped to the DDOF Playbook lifecycle, the DevSecOps pipeline tempo that SO 7 requires corresponds to:

A. Annual release cycles aligned with the Army fiscal year B. The DDOF 30-day MVP mandate — SWEs build pipelines that deliver minimum viable products within 30 days per DDOF Phase 4 C. Quarterly sprint reviews with C2DAO governance approval at each quarter boundary D. Continuous deployment with no governance gates — speed is the primary metric

17. Per UDRA v1.1, every data product in the Army enterprise must carry 15 required metadata fields. Three of these fields are designated security-critical. The security-critical metadata fields are:

A. Creator, Last Modified Date, and File Size B. Classification Marking, Access Control List, and Data Steward Contact C. Schema Version, Retention Policy, and Update Frequency D. Domain Owner, Quality Score, and Lineage Reference

---

SECTION 2 — SHORT ANSWER

Answer in 2–5 sentences. (6 points each)

SA-1. Describe the complete OSDK workflow for authenticating to a Foundry Ontology, executing a paginated filtered query on a Vehicle Object Type for vehicles with `status = "RED"`, and handling the case where the result set spans multiple pages.

SA-2. You are building a TypeScript Action validator for a maintenance work order submission. The validator must check: (1) the mileage field is a positive integer, (2) the new mileage is greater than the vehicle's current mileage, (3) the work order date is not in the future, and (4) the technician ID exists in the Technician Object Type. Describe the validator structure, including how you handle cross-field validation (condition 2) and the async lookup (condition 4).

SA-3. Describe the N+1 anti-pattern in the context of a TypeScript FOO that computes a `maintenance_risk_score` property for each Vehicle based on the count of maintenance records in the last 90 days. Show the anti-pattern and describe the correct bulk query pattern to replace it.

SA-4. A Slate application displays a list of open maintenance work orders and an Action button to close a work order. After the user clicks "Close Work Order," the list does not update to reflect the change. Describe how you would fix this using proper state management (not a timer), and explain why the timer approach is incorrect.

SA-5. Walk through the C2DAO deployment checklist for a new TypeScript OSDK integration you have built. List at least six checklist items and identify the two credential-related items that are the most common failure points during review.

SA-6. Describe how software engineering capability supports two WFF functions. For each, identify the WFF track (SL 4A through SL 4F) and give a concrete example of a custom TypeScript OSDK application or integration that supports decision-making in that function.

SCORING SUMMARY

Section	Questions	Points Each	Total Points
Multiple Choice	17	2	34
Short Answer	6	6	36
Total	—	—	**70**

Passing: 49/70 (70%) — Post-test only. Pre-test is diagnostic.

ANSWER KEY — INSTRUCTOR USE ONLY

Do not distribute to students.

Multiple Choice: 1. D — OSDK client requires: client instance + authentication token + generated type definitions. 2. C — ResourceIterator must be fully consumed to retrieve all pages. 3. B — Breaking early retrieves only those records — subsequent pages are not fetched. 4. A — Poll task status using the task ID from the Action response until SUCCEEDED or FAILED. 5. C — OSDK Link traversal method on the object instance is the correct pattern. 6. C — Bulk query fetches all related records in one call; N+1 per-object loop is the anti-pattern. 7. A — Minimum 3 distinct conditions, including at least one cross-field validation. 8. D — State variable update on Action completion triggers reactive data refresh. 9. B — Subscription handlers must be idempotent for duplicate event delivery. 10. A — No hardcoded credentials; all injected via environment variables or secrets management. 11. B — New mileage vs. current mileage comparison is a single-field range validation (one field context). 12. C — Transactions are atomic — all-or-nothing prevents partial writes. 13. D — Minimum 8 test cases (per SL 4L standards). 14. B — Hardcoded tenant IDs risk cross-tenant data access if deployed in different context. 15. A — Strip/reject injection characters, validate length and format before using in queries or writes. 16. B — The DDOF 30-day MVP mandate — SWEs build pipelines that deliver minimum viable products within 30 days per DDOF Phase 4. 17. B — Classification Marking, Access Control List, and Data Steward Contact are the three security-critical UDRA v1.1 metadata fields.

Short Answer Guidance:

SA-1. Full credit: (1) initialize Foundry client with `createClient(instanceUrl, authToken, ontologyTypes)`; (2) create a `ResourceIterator` query on Vehicle with filter `status = "RED"`; (3) iterate through the ResourceIterator with `for await (const vehicle of iterator)` — this automatically handles pagination fetching subsequent pages; (4) collect all results into an array or

process each record in the loop body; (5) handle errors with try/catch wrapping the iterator. Must explicitly state that ResourceIterator handles pagination automatically and that iteration must continue until exhausted.

SA-2. Full credit: validator structure — (1) validate mileage is a positive integer (type check + > 0); (2) async: fetch the current vehicle record by ID; compare new mileage > current mileage (cross-field: uses current vehicle state from the Ontology); (3) validate work order date ≤ today (date comparison); (4) async: look up technician ID in Technician Object Type — if not found, reject with clear error message; all four conditions checked; return array of validation errors for any failures; async conditions handled with await inside the validator function. Must address async handling and cross-field logic for full credit.

SA-3. Full credit: anti-pattern — for each Vehicle in the list (N vehicles), make a separate query to fetch its MaintenanceRecords and count records in the last 90 days = N+1 queries; correct pattern — one bulk query to fetch all MaintenanceRecords with `date >= 90_days_ago` in a single call; build an in-memory map of vehicle_id → record_count; in the FOO computation, look up each vehicle's count from the map instead of making per-vehicle API calls. Must show both patterns and explain why N+1 is a performance problem.

SA-4. Full credit: fix — when the "Close Work Order" Action completes, set a state variable (e.g., `lastClosedWorkOrderId` or `refreshTrigger`) to the result of the Action; the work order list component is bound to this state variable — when it changes, the component re-fetches the open work orders list; the list reactively updates without a timer; why timer is wrong — race condition: if the Action takes longer than the timer delay, the data refresh occurs before the Action is committed and the list will not show the change; also timers are arbitrary and brittle. Must include state variable update pattern AND explanation of why timer is incorrect.

SA-5. Full credit: any six from — (1) no hardcoded credentials in source code; (2) all credentials injected via environment variables; (3) input sanitization applied to all user-supplied values; (4) no sensitive data logged to console or monitoring systems; (5) all OSDK queries use correct pagination (ResourceIterator fully consumed); (6) Action validators have minimum 8 test cases; (7) TypeScript strict mode enabled; (8) all async calls have error handling; (9) no N+1 query patterns; (10) C2DAO branch description written and submitted for data steward review. Two most common credential failures: (1) credentials hardcoded in source files; (2) credentials present in `.env` files committed to the repository.

SA-6. Full credit: any two WFF tracks correctly identified with a SWE example — SL 4A (Intelligence): TypeScript Slate application built to display incoming OSINT entity extractions from the Intelligence Object Type; SL 4B (Fires): OSDK integration that queries Fires Object Types and pushes targeting data to a fires coordination Workshop application; SL 4C (Movement & Maneuver): custom TypeScript integration that reads vehicle position Object Types and renders a real-time maneuver tracking view; SL 4D (Sustainment): OSDK-based pipeline that ingests GCSS-Army supply data and writes to Sustainment Object Types for G4 Workshop dashboards; SL 4E (Protection): custom webhook integration that receives force protection sensor events and writes alerts to the Protection Ontology for analyst review; SL

4F (Mission Command): TypeScript Workshop widget that aggregates cross-WFF status indicators into a commander's common operating picture. Each response must identify the correct SL 4 letter (A–F) and provide a concrete software engineering deliverable for full credit.

USAREUR-AF Operational Data Team TM-40L Post-Test | Version 1.0 | March 2026