PRACTICAL EXERCISE

EX-40L

USAREUR-AF

EX_40L — Software Engineer

Practical Exercise — SL 4L Proficiency

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

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

EX_40L — SOFTWARE ENGINEER

PRACTICAL EXERCISE — SL 4L PROFICIENCY

Field	Value
Version	1.0 — March 2026
Prerequisite	SL 3 REQUIRED; SL 4L — Software Engineer Technical Manual (and SL 1 through SL 2)
Duration	3–4 hours
Environment	MSS with Python Transforms, TypeScript/OSDK access, Code Workspace — see ENVIRONMENT_SETUP.md

COMPANION RESOURCES

Resource	Reference
Technical Manual	SL 4L — Software Engineer
Syllabus	SYLLABUS_TM40L
Pre-Exercise Exam	EXAM_TM40L_PRE
Post-Exercise Exam	EXAM_TM40L_POST
Continuation Track	SL 5L — Advanced Software Engineer

WFF AWARENESS

The personnel readiness pipeline and OSDK action built in this exercise are operational tools consumed by WFF track personnel (TM-40A–F) — the readiness feed informs mission command, maneuver, and sustainment planning across echelons. Evaluators should verify that the TypeScript action interface and authorization model are appropriate for multi-track, multi-role consumer access.

SCENARIO

The OPDATA team needs a Python Transform pipeline that ingests a synthetic personnel readiness feed, normalizes and validates the data, exposes key entities via the Ontology (OSDK), and surfaces a TypeScript-based action that lets authorized users update a record status from an external application.

Training dataset: Synthetic personnel readiness feed, CSV format, ~1,000 records.

TASK LIST

Task 1 — Ingest and Validate (30 min)

- ☐ Write a Python Transform that ingests the CSV feed
- ☐ Validate: required fields present, date fields parseable, numeric fields in valid range
- ☐ Log validation failures to a separate error dataset — do not silently drop bad records
- ☐ Make the transform idempotent (re-run produces same output, no duplicates)

Standard	Criteria
Go	Transform runs; error dataset captures failures; idempotency verified by running twice
No-Go	Errors silently dropped, transform not idempotent, or validation logic absent

Task 2 — Normalize and Enrich (30 min)

- ☐ Normalize unit names to uppercase; trim whitespace from all string fields
- ☐ Add a `record_hash` column (SHA-256 of key fields) for change detection
- ☐ Add `ingested_at` timestamp
- ☐ Output to a clean dataset following naming standards

Standard	Criteria
Go	Normalization applied; hash and timestamp present; naming standard followed
No-Go	Normalization missing or hash column absent

Task 3 — Ontology Object Type and Properties (45 min)

- ☐ Create (or use a pre-created) `PersonnelRecord` Object Type in the Ontology

- ☐ Map properties: `personnel_id` (primary key), `unit`, `readiness_status`, `record_hash`, `ingested_at`
- ☐ Write an OSDK-backed Python snippet that retrieves all records for a given unit
- ☐ Verify the OSDK query returns correct results for a test unit

Standard	Criteria
Go	Object Type has correct property mapping; OSDK query returns correct records
No-Go	Property mapping is wrong or query returns empty/incorrect results

Task 4 — TypeScript Action (45 min)

- ☐ Write a TypeScript Action (Foundry Actions / OSDK) that accepts a `personnel_id` and a new `readiness_status` value and updates the record
- ☐ Add an authorization check: only users in the "data-steward" group can execute the action
- ☐ Test as an authorized user; verify the update persists
- ☐ Test as an unauthorized user; verify the action is blocked

Standard	Criteria
Go	Action updates correctly for authorized user; blocked for unauthorized user
No-Go	Authorization check absent or action fails for authorized user

Task 5 — Code Review Checklist (20 min)

Self-review your Python Transform against the SL 4L code quality checklist:

Criterion	Check
No hardcoded values	Pass / Fail
Parameterized configs	Pass / Fail
No sensitive data in logs	Pass / Fail
Inline comments on non-obvious logic	Pass / Fail

- ☐ Produce a written checklist with pass/fail for each criterion

Standard	Criteria
Go	Checklist is complete; any failures are acknowledged and explained
No-Go	Checklist absent or critical failures (hardcoded credentials, sensitive data in logs) not flagged

EVALUATOR NOTES

Scoring: 5 tasks. Go on 4 of 5 = overall Go. No-Go on Task 1 or Task 4 = automatic No-Go (data validation/idempotency and authorization are foundational SWE requirements).

Pre-Exercise Checklist

- ☐ Confirm Python Transforms are enabled and training accounts can create transforms
- ☐ Confirm the `PersonnelRecord` Object Type exists in the training Ontology with all required property slots
- ☐ Confirm the "data-steward" group is created and populated with exactly one test account (evaluator will test the unauthorized path with a separate account)
- ☐ Confirm OSDK TypeScript SDK is installed in the Code Workspace or accessible to the training account
- ☐ Have a second (non-data-steward) training account ready for authorization testing in Task 4

Common Failure Modes

Task	Common Failure	Evaluator Guidance
Task 1	Idempotency claimed but not demonstrated	Run the transform twice and diff the output datasets — if row count differs, No-Go
Task 1	Bad records silently dropped, no error dataset	Ask: "Where do records that fail validation go?" — if no error dataset, No-Go
Task 2	`record_hash` computed on all fields, not key fields	Ask participant which fields they included in the hash; including non-deterministic fields (e.g., `ingested_at`) breaks change detection
Task 3	OSDK query returns records for wrong unit	Run with a known test unit and verify expected count; incorrect count = No-Go
Task 4	Authorization check uses if/else on username, not group membership	Ask participant to show the authorization mechanism; username hardcoding is No-Go
Task 5	Self-review checklist passes everything	Ask participant to show their code for one item they marked as Pass — common issue: comments on obvious logic, not on non-obvious logic

Timing Notes

- Total exercise routinely runs 4+ hours — strongly recommend splitting over two sessions
- Task 4 (TypeScript Action) is the most common time sink; TypeScript setup in Foundry can take 30–45 min for first-time users
- If participant is significantly behind at Task 3, offer to split the exercise; mark Tasks 1–2 and assess Tasks 3–5 separately

NEXT STEPS

Participants who receive an overall Go on EX_40L are eligible to enroll in **SL 5L — Advanced Software Engineer**. SL 5L extends into CI/CD pipeline engineering, automated testing frameworks, and production Foundry application deployment. SL 5 is G–O (advanced specialist tracks).

SUPPLEMENTAL — BUILD WITH AIP OFFICIAL TUTORIALS

Tutorial	Key Concepts	Relevance
Advanced To Do App (OSDK)	Interfaces, media, runtime-derived properties, Platform SDK	OSDK patterns
Register an LLM via Function Interfaces	Custom LLM integration, TypeScript functions, webhooks	Function interfaces
Platform Governance App (Platform SDK)	Metadata management, linter alerts, admin workflows	Platform SDK, governance
Obfuscate Data with Cipher	CipherText properties, selective privacy	Data protection, CBAC

Install via build.palantir.com — search "Examples (Build with AIP)" in the application drawer.

ENVIRONMENT SETUP

See ENVIRONMENT_SETUP.md for full setup instructions.