DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

STANDARDS AND GOVERNANCE

MSS-GOV

USAREUR-AF

USAREUR-AF MSS NAMING AND GOVERNANCE STANDARDS

Standards And Governance

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

DRAFT — NOT FOR OFFICIAL USE. FOR TRAINING PLANNING PURPOSES ONLY.

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

USAREUR-AF MSS NAMING AND GOVERNANCE STANDARDS

HEADQUARTERS, UNITED STATES ARMY EUROPE AND AFRICA Wiesbaden, Germany

Proponent: USAREUR-AF C2DAO **Authority:** AR 25-1, *Army Information Technology* (Jul 2019); AR 25-30, *Army Publishing Program* (Jun 2022); DA PAM 25-40, *Army Publishing: Action Officers' Guide* (Nov 2023); Army CIO Data and Analytics Policy (April 2024); UDRA v1.1 (February 2025) **Distribution:** Distribution authorized to U.S. Government agencies and their contractors only. Other requests must be referred to Headquarters, C2DAO, Wiesbaden, Germany. **Version:** 1.0 | March 2026

NOTE

All TMs in the Maven training series reference "USAREUR-AF naming conventions." This document is the authoritative source. When in doubt, consult C2DAO or the responsible data steward.

SECTION 1 — PROJECT NAMING

1-1. Format

```
[UNIT]-[FUNCTION]-[DESCRIPTOR]
```

Component	Description	Example
UNIT	Unit identifier (USAREUR, VCORPS, 21TSC, G2, G6, S3, etc.)	`VCORPS`
FUNCTION	Functional domain (OPDATA, LOG, PERS, INTL, PM, KM, AI)	`LOG`
DESCRIPTOR	Short plain-English descriptor of the project's purpose	`FuelTracking`

Examples: `VCORPS-LOG-FuelTracking`, `USAREUR-G2-ThreatAnalysis`, `21TSC-PM-ProgramDashboard`, `USAREUR-KM-LessonsLearned`

1-2. Rules

- Unit and function identifiers: ALL CAPS; descriptor: CamelCase or snake_case

- No spaces — use hyphens between components
- Maximum 50 characters
- No classification markings, personal names, nicknames, or humor in project names
- Projects in shared enterprise space require C2DAO coordination before creation

---

SECTION 2 — DATASET NAMING

2-1. Format

```
[tier]_[unit]_[source]_[content]
```

Component	Description	Example
tier	Pipeline tier: `raw`, `bronze`, `silver`, `gold`	`silver`
unit	Unit or system of record	`gcss`
source	Source system or feed	`dailyreport`
content	What the data represents	`equipment_status`

Examples: `raw_gcss_dailyreport_equipment_status`, `silver_ipps_weekly_personnel_strength`, `gold_g2_monthly_threat_summary`, `bronze_manual_sitrep_submissions`

2-2. Rules

- All lowercase, underscores only (no hyphens, no spaces)
- Tier prefix mandatory: `raw` → `bronze` → `silver` → `gold`
- Do not abbreviate content descriptors to the point of ambiguity
- Include unit/system of record so consumers know provenance without reading lineage
- Cross-unit aggregates use `enterprise` as the unit component (e.g., `gold_enterprise_readiness_summary`)

SECTION 3 — OBJECT TYPE AND PROPERTY NAMING

3-1. Object Type Format

```
[Domain][Entity]
```

- PascalCase, no underscores, no hyphens; singular noun
- Domain prefix for shared enterprise types; omit for unit-local types

Examples: `LogEquipmentItem`, `PersStrengthRecord`, `MissionEvent`, `ProgramMilestone`, `ThreatIndicator`, `LessonLearned`

3-2. Property Naming

camelCase; descriptive and unambiguous. Use standardized suffixes:

Suffix	Use	Example
`Id`	Unique identifier	`equipmentId`, `unitId`
`Date`	Calendar date (no time)	`reportingDate`, `completionDate`
`Dtg`	Date-time group	`submissionDtg`, `lastUpdatedDtg`
`Status`	Enumerated status value	`readinessStatus`, `missionStatus`
`Count`	Integer quantity	`personnelCount`, `vehicleCount`
`Rate`	Percentage or ratio (0–1 or 0–100, documented)	`missionCapableRate`
`Name`	Human-readable label	`unitName`, `programName`
`Code`	Standardized code value	`unitIdentificationCode`, `dodaac`

3-3. Property Rules

- No generic names: `value`, `data`, `info`, `misc` — be specific
- Date properties: ISO 8601 (`YYYY-MM-DD`) unless documented otherwise
- Enum values: `ALL_CAPS_UNDERSCORE` (e.g., `MISSION_CAPABLE`, `NOT_READY`, `UNKNOWN`)
- Boolean properties: prefix with `is` or `has` (e.g., `isMissionCapable`, `hasOverdueActions`)

Object Type Naming and Lifecycle Standards

Naming Rules: - Use natural business language — spell out full terms, no abbreviations in type names. - No versioned names — `Message_v2` is prohibited. Modify the existing type or create a new purpose-specific type. - No `[tag]` prefixes in object type names — use Ontology object type groups for categorization. - Property naming conventions: - Timestamps: `{action}_at_timestamp` (e.g., `created_at_timestamp`, `submitted_at_timestamp`) - Authors: `{action}_by_user` (e.g., `created_by_user`, `approved_by_user`) - Foreign key columns: `{foreign_object_type}_id` or `{link_api_name}_{foreign_object_type}_id`

Lifecycle Maturity:

Every object type in the MSS Ontology must carry one of three maturity statuses:

Status	Definition	Governance
Experimental	Under development, schema may change without notice	Sandbox/dev branches only
Active	Production-ready, schema changes require change request	Protected; governed by C2DAO standards
Deprecated	Superseded or scheduled for removal	Read-only; migration plan required

Ownership: Every object type requires a designated point of contact responsible for schema currency, data quality, and access control.

Source: Palantir Developer Community — Ontology and Pipeline Design Principles, adapted for USAREUR-AF governance.

SECTION 4 — LINK TYPE NAMING

4-1. Format

```
[SourceObjectType]_[relationship]_[TargetObjectType]
```

- snake_case; relationship verb in present tense, directional

Examples: `LogEquipmentItem_assignedTo_Unit`, `MissionEvent_supports_Program`, `LessonLearned_appliesTo_MissionEvent`, `ProgramMilestone_hasRisk_ProgramRisk`

4-2. Rules

- Direction matters — link name reads left-to-right as a sentence fragment
- No generic relationships: `relatedTo`, `linkedTo` — be specific
- Bidirectional links: both directions named logically

SECTION 5 — ACTION NAMING

5-1. Format

```
[Verb][ObjectType][Qualifier]
```

- PascalCase; imperative verb (Submit, Update, Approve, Close, Flag, Create, Archive)

Examples: `SubmitSitrep`, `UpdateEquipmentStatus`, `ApproveMilestone`, `FlagThreatIndicator`, `ArchiveLessonLearned`, `CreateProgramRisk`

5-2. Rules

- Verb must accurately reflect the operation — distinguish Create / Update / Delete / Submit / Approve / Archive; do not use `Edit` for everything
- Actions writing to shared enterprise Object Types require Data Steward review before publication
- Destructive actions (Delete, Archive) require additional confirmation logic in the Action validator

---

SECTION 6 — WORKSHOP APPLICATION NAMING

6-1. Format

```
[Unit] [Function] [Dashboard/Form/App] v[N]
```

- Title case; version number appended only when multiple versions coexist; drop when only one exists

Examples: `VCORPS Readiness Dashboard`, `21TSC Fuel Status Dashboard v2`, `USAREUR Program Health Report`, `G6 MSS Access Request Form`, `S3 Mission Event Tracker`

6-2. Rules

- Application title is user-facing — write for the consumer, not the builder
- Include unit identifier so users in a shared environment know the owner
- Applications published to shared enterprise space require C2DAO coordination
- All commander-facing dashboards must display a **data-as-of** timestamp prominently

SECTION 7 — PIPELINE AND TRANSFORM NAMING

7-1. Pipeline Builder Job Names

```
[unit]_[source]_to_[destination]_[frequency]
```

Examples: `gcss_daily_equipment_to_bronze_daily`, `ipps_weekly_personnel_to_silver_weekly`, `enterprise_readiness_aggregation_to_gold_daily`

7-2. Code Transform Names (Python)

- snake_case function names; descriptive of the transformation, not the data
- Required docstring: input dataset(s), output dataset, transformation logic summary, last-updated date

```
@transform_df(
    Output("/USAREUR-AF/gold/enterprise_readiness_summary"),
    personnel=Input("/USAREUR-AF/silver/ipps_weekly_personnel_strength"),
    equipment=Input("/USAREUR-AF/silver/gcss_daily_equipment_status"),
)
def compute_readiness_summary(personnel, equipment):
    """
    Joins personnel strength and equipment status to produce
    enterprise readiness summary by unit.

    Inputs:
        personnel: IPPS-A weekly strength data (silver tier)
        equipment: GCSS-Army daily equipment status (silver tier)
    Output:
        Readiness summary at unit level (gold tier)
    Last updated: 2026-03
    """
```

7-3. Rules

- Do not name transforms after their author or assignment (not `smith_transform`, `s6_project_may`)

- Transforms must be idempotent — running twice on the same input produces the same output
- Incremental transforms must document the watermark column and increment logic in the docstring

---

SECTION 8 — BRANCH NAMING

8-1. Format

```
[type]/[unit]-[descriptor]-[YYYYMM]
```

Type	Use
`dev`	Active development branch
`review`	Submitted for Data Steward or C2DAO review
`hotfix`	Emergency production fix

Examples: `dev/vcorps-readiness-dashboard-202603`, `review/usareur-threat-ontology-update-202603`, `hotfix/sitrep-pipeline-null-fix-202603`

8-2. Rules

- Never build directly on `main` — always use a dev branch
- Branch names: lowercase with hyphens
- Delete branches after merge — no stale dev branch accumulation
- Production promotion requires Data Steward sign-off documented in the review ticket
- `main` is the production state — treat it as operational

SECTION 9 — CODE AND REPOSITORY STANDARDS

9-1. File and Module Naming

- Python modules: `snake_case.py`; config files: `snake_case.yaml` or `snake_case.json`
- No spaces in any filename committed to version control
- **Curriculum directories** use underscores as the separator between prefix and code:

- Training Materials: `TM_40A_intelligence`, `TM_50G_orsa_advanced`
- Exercises: `EX_40A_intelligence`, `EX_50G_orsa`
- Format: `{PREFIX}_{LEVEL}{TRACK}_{descriptor}` — all underscores, no hyphens

9-2. Variable and Function Naming

Language	Variables/Functions	Classes/Types	Constants
Python	`snake_case`	`PascalCase`	`ALL_CAPS_UNDERSCORE`
TypeScript	`camelCase`	`PascalCase`	`ALL_CAPS_UNDERSCORE`

9-3. Commit Message Format

```
[type]: [short description] ([unit/project])

[optional body — what changed and why]
```

Types: `feat`, `fix`, `refactor`, `docs`, `test`, `chore`

Examples: - `feat: add fuel consumption forecast model (VCORPS-LOG)` - `fix: null handling in IPPS strength join (USAREUR-PERS)` - `docs: update SL 3 workshop chapter (maven-training)`

9-4. Security Rules (Non-Negotiable)

- **Never** commit credentials, connection strings, or access tokens — use environment variables or approved credential stores
- **Never** commit operational data payloads, even anonymized
- **Never** use f-strings or string concatenation to construct SQL — parameterize all queries
- All ingestion scripts must validate inputs at the boundary before processing

SECTION 10 — GOVERNANCE CHECKPOINTS

10-1. Before Creating a New Object Type in a Shared Domain

- ☐ Coordinate with C2DAO — confirm no existing Object Type covers the use case
- ☐ Define all properties, types, and enum values before creation
- ☐ Identify the responsible Data Steward

- ☐ Document expected consumers (who will query this Object Type?)
- ☐ Apply USAREUR-AF naming conventions per Section 3

10-2. Before Publishing a Workshop Application

- ☐ Data-as-of timestamp displayed on all data-backed views
- ☐ Access controls reviewed — only authorized users can reach sensitive data
- ☐ Tested with production-representative data (not just synthetic)
- ☐ C2DAO notified if application writes back to shared enterprise Object Types via Actions
- ☐ Application title follows naming standard (Section 6)

10-3. Before Promoting a Pipeline to Production

- ☐ Transform is idempotent — tested by running twice
- ☐ Failure alerting configured (email or notification to responsible owner)
- ☐ Downstream consumers notified 24 hours before changes to existing production pipelines
- ☐ Data Steward sign-off documented
- ☐ Branch naming follows standard (Section 8)

10-4. Before Deploying Code to Production (SL 4L / SL 4M)

See the full governance checklist in the applicable TM appendix: - SL 4M Appendix A — Model Governance Checklist - SL 4L Appendix B — SWE Security Checklist

APPENDIX A — APPROVED ABBREVIATIONS

Abbreviation	Full Term
AOR	Area of Responsibility
C2DAO	Command and Control Data and Analytics Office
CDA	Command Data Analytics
DODAAC	Department of Defense Activity Address Code
DTG	Date-Time Group
FOO	Functions on Objects
GFEBS	General Fund Enterprise Business System

Abbreviation	Full Term
GCSS	Global Combat Support System
IPPS-A	Integrated Personnel and Pay System — Army
KM	Knowledge Management / Knowledge Manager
LOGREP	Logistics Report
MSS	Maven Smart System
OPDATA	Operational Data
ORSA	Operations Research and Systems Analysis
OSDK	Ontology Software Development Kit
PERSTAT	Personnel Status
PM	Program Manager
RAG	Red-Amber-Green (status indicator)
SITREP	Situation Report
SWE	Software Engineer
USAREUR-AF	United States Army Europe and Africa
UIC	Unit Identification Code

APPENDIX B — GOVERNANCE CHECKLIST (BEFORE PROMOTING TO PRODUCTION)

```
[ ] Naming conventions followed (this document, applicable section)
[ ] Data Steward identified and notified
[ ] C2DAO coordination completed (if required — see Section 10)
[ ] Downstream consumers notified (for pipeline changes)
[ ] Access controls reviewed
[ ] Data-as-of timestamp present (for dashboards)
[ ] Testing completed on production-representative data
[ ] Branch naming follows standard
[ ] Commit messages follow standard
[ ] No hardcoded credentials or connection strings
[ ] All SQL parameterized
[ ] Idempotency verified (for pipelines/transforms)
[ ] TM-specific governance checklist completed (SL 4M App. A / SL 4L App. B)
```

USAREUR-AF Operational Data Team Version 1.0 | March 2026 | Proponent: C2DAO