ARCHITECTURE REFERENCE

ODT-EA

Enterprise Architecture — 05: Military and Defense Application

Architecture Reference

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

20 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

sidebar_position: 6 title: "Military Application"

ENTERPRISE ARCHITECTURE — 05: MILITARY AND DEFENSE APPLICATION

EA in the context of USAREUR-AF, theater commands, and defense data platforms

WHY MILITARY EA IS HARDER THAN COMMERCIAL EA

Commercial EA operates in a relatively stable environment: one organization, one legal framework, bounded technology choices, clear ownership. Military EA at the theater level operates under conditions that make all of that harder:

- **Multiple classification domains** (NIPR, SIPR, JWICS) with separate systems, different governance, and restricted data flows
- **NATO interoperability requirements** alongside U.S. joint standards
- **Rapid mission change** — the architecture must support today's exercise, next month's contingency, and a potential near-peer conflict
- **Acquisition lag** — technology decisions made 5 years ago constrain what's deployable today
- **Coalition and partner nation requirements** — data sharing across national systems with different standards and different trust levels
- **Distributed authority** — CTO/CIO functions exist at HQDA, theater, corps, division, brigade levels, each with partial authority and partial visibility

These aren't reasons to abandon EA. They're reasons EA is *especially* important — and especially likely to fail if done naively.

THE MILITARY EA STACK

At the theater level, the EA stack maps roughly as follows:

```
HQDA / DoD Level
├── DODAF-mandated architecture products
├── CDAO data strategy and AI policy
├── JCIDS capability requirements
├── RMF security standards
└── Palantir / DCGS / enterprise platform mandates

Theater / USAREUR-AF Level
├── Theater campaign plan data requirements
├── Cross-domain solution architecture (NIPR/SIPR/JWICS)
├── NATO STANAG interoperability requirements
├── Theater-specific capability gaps and investment priorities
└── Operational data platform governance (ODT)

Corps / Division Level
├── Echelon-specific capability requirements
├── Mission command system configurations
└── Data collection and reporting standards

Program / System Level
├── Individual system architectures
├── API and integration implementations
└── Data model implementations
```

EA at the theater level has to be coherent *downward* (guiding corps/division architectures) and *upward* (feeding HQDA architecture and investment processes) simultaneously.

DODAF: THE MANDATED FRAMEWORK

DODAF is the Department of Defense Architecture Framework — the required format for defense architecture products. Understanding its structure is essential for theater EA work.

The Key View Sets

Capability Views (CV): What the command needs to be able to do. - CV-1: Vision (narrative) - CV-2: Capability taxonomy - CV-3: Capability phasing (what's needed when) - CV-4: Capability dependencies

Operational Views (OV): How operations and work flow. - OV-1: High-level operational concept graphic - OV-2: Operational resource flow description - OV-5: Operational activity model (what activities happen and in what sequence) - OV-6: Operational rules / event-trace / state diagrams

Systems Views (SV): What systems exist and how they connect. - SV-1: Systems interface description - SV-4: Systems functionality description - SV-6: Systems data exchange matrix

Data and Information Views (DIV): The semantic/data layer. - DIV-1: Conceptual data model (entities and relationships) - DIV-2: Logical data model (attributes, constraints) - DIV-3: Physical data model (implementation schema)

Technical Standards Views (TV): What standards govern the architecture. - TV-1: Technical standards profile - TV-2: Technical standards forecast

The Practical Reality

Most DODAF products are produced to satisfy a requirement, not to drive decisions. The highest-value DODAF products in an operational context are: CV-2 (capability taxonomy), OV-5 (operational activities), DIV-1/2 (data models), and TV-1 (standards profile). These are the ones that actually inform acquisition, integration, and data governance decisions.

NATO INTEROPERABILITY: THE ADDITIONAL LAYER

USAREUR-AF is a NATO-committed theater command. This adds an interoperability requirement layer that sits alongside DODAF:

NATO Architecture Framework (NAF): NATO's architecture framework, which aligns with DODAF and MODAF. Theater EA must be NAF-compatible to support coalition interoperability.

STANAGs: NATO Standardization Agreements that define technical and procedural standards. Key for EA: - STANAG 4559 (Intelligence data exchange) - STANAG 5500 (Tactical data link) - STANAG 2014 (Operations order format) - APP-6 (Military symbols — basis for COP standards)

Key EA implication: The data/information view (DIV) must accommodate both U.S. authoritative data sources (MIM, UCORE, IC-ISM) and NATO data exchange standards. This is a semantic translation problem — which is exactly where ontologies and semantic modeling become operationally critical.

CROSS-DOMAIN SOLUTIONS: THE ARCHITECTURE CONSTRAINT THAT CHANGES EVERYTHING

In a classified environment, cross-domain solutions (CDS) are the most significant architectural constraint. CDS is the mechanism by which data moves between classification domains (NIPR ↔ SIPR ↔ JWICS) through approved, inspected channels.

EA implications of CDS: - Data architecture must explicitly model which data can flow across which domain boundaries - The data model must include classification/releasability metadata as a first-class attribute - System architectures must document CDS integration points and latency/bandwidth constraints - The authoritative source of record for any given data entity may live in a different domain than the system that needs to consume it

The common failure: Designing a beautiful data architecture and then discovering the critical data lives in a domain the consuming system can't reach — or that the CDS introduces latency that breaks the operational use case.

Fix: CDS topology is an input to data architecture design, not an afterthought. The data/information view must include domain boundaries as a first-class element.

THE PALANTIR FOUNDRY CONTEXT

Foundry is, in practice, a platform that implements significant portions of the EA data layer. Understanding how Foundry's capabilities map to EA concepts:

EA Concept	Foundry Implementation
Authoritative source registry	Dataset provenance, source tracking
Conceptual/logical data model	Ontology (object types, relationships, properties)
Data integration pipeline	Pipeline Builder / Code Workbook transforms
Semantic layer	Ontology object types as governed entities
Data governance	Dataset access controls, branch-based review
Operational views (OV)	Workshop applications / Slate dashboards
Systems interface (SV-6)	Pipeline source/target documentation
Cross-domain data flow	AIP Gateway / cross-domain Foundry instances

This means the **ontology in Foundry is the physical instantiation of the DIV (Data and Information View)**. Good EA data modeling directly produces deployable Foundry ontology design. Bad EA produces ontologies that don't reflect the real data landscape and have to be rebuilt.

Key principle: Ontology design should be *doctrine-driven* (grounded in authoritative publications like MIM, FM 3-0, ADP 5-0) and *EA-governed* (aligned to the theater capability map and operational activity model). Ontologies that are designed bottom-up from available data sources instead of top-down from operational requirements produce technically correct but operationally useless models.

THE THEATER CTO'S EA AGENDA

For a command-level CTO at USAREUR-AF, the EA agenda organizes around five imperatives:

1. Establish the Capability Map

Define what the theater command needs to be able to do — not what systems it has, but what capabilities it requires. This becomes the organizing framework for all investment, acquisition, and data governance decisions.

2. Build the Data/Information View

Document what data the theater depends on, who is authoritative for each entity, where it lives (which domain, which system), and how it flows. This is the foundation for Foundry ontology governance, CDS architecture, and semantic interoperability with NATO.

3. Define the Technology Reference Architecture

Specify approved platforms (Foundry, DCGS-A, ATAK, etc.), integration patterns, CDS-approved data flow patterns, and security standards. This is what gets written into acquisition requirements and SOW language.

4. Establish ARB Governance

Stand up the Architecture Review Board with defined scope, criteria, and authority. Connect it to the Program Synchronization Board (PSB) and acquisition pipeline so EA review happens before design freeze.

5. Build the Roadmap

Develop a sequenced, dependency-aware roadmap that maps capability gaps to programs, programs to data requirements, and data requirements to platform investments. Make the critical path visible.

THE DOCTRINE-DATA CONNECTION

The most distinctive aspect of defense EA at the data layer is the requirement to ground semantic models in published doctrine. This is not just a formalism — it has operational consequences:

- A data entity called "Unit" defined locally means something only within that system
- A data entity called "Unit" grounded in MIM 5.3, ADP 5-0, and FM 3-0 means something across every system that follows the same standards
- Interoperability is only possible when data models share authoritative semantic grounding

This is why **ontology-as-code** methodologies — converting MIM 5.3 specifications directly into deployable Python packages and Foundry ontologies — are operationally significant, not just technically elegant. They close the loop between published doctrine and deployed data architecture, eliminating the semantic drift that accumulates when data models are built by engineers who've never read the doctrinal publication.

EA MATURITY: WHERE TO START

For a theater CTO establishing EA from a low-maturity baseline, the sequence:

Phase	Focus	Key Output
0 — Foundation	Establish vocabulary, get leadership alignment on EA purpose	EA charter, school-of-thought decision
1 — Inventory	Document what exists (systems, data, capabilities)	App inventory, initial data map, capability list
2 — Model	Structure the inventory into governed views	Capability map, data/information view, tech standards
3 — Govern	Stand up ARB, connect to investment/acquisition processes	ARB charter, intake criteria, portfolio alignment
4 — Roadmap	Define the transformation path with sequenced initiatives	Capability heat map, roadmap with dependencies
5 — Maintain	Make EA a living practice, not a one-time deliverable	Review cadence, update triggers, metrics

The trap is skipping phases 0–2 and jumping to phase 4 ("leadership wants a roadmap"). A roadmap built without an accurate current-state inventory is fiction. Start with the inventory.

Previous: 04 — Governance