DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

ARCHITECTURE REFERENCE

ODT-GDAP

USAREUR-AF

Global Doctrine Alignment Platform (GDAP)

Tagline: Doctrine as Executable Architecture

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

20 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

GLOBAL DOCTRINE ALIGNMENT PLATFORM (GDAP)

TAGLINE: DOCTRINE AS EXECUTABLE ARCHITECTURE

1. PROBLEM STATEMENT

Coalition operations routinely suffer from friction caused by:

- Doctrinal divergence (terminology, task structure, authorities, assumptions)
- Misaligned command relationships
- Inconsistent information requirements
- Data models that do not reflect doctrinal intent
- Systems built without semantic grounding in doctrine

Doctrine is currently stored as static documents (PDFs, manuals, diagrams). It is:

- Not machine-readable
- Not atomized
- Not temporally comparable
- Not structurally mapped to data architecture
- Not cross-nation harmonized

As a result, doctrinal differences are discovered late — during execution, integration, or conflict.

We need doctrine to become structured, comparable, and executable.

2. VISION

Create a global, bi-temporal, machine-reasonable doctrinal knowledge graph that:

1. Decomposes doctrine into atomic semantic units
2. Maps doctrinal elements to data models and systems
3. Identifies divergences across nations

4. Tracks change over time
5. Surfaces operational and architectural friction before execution

Doctrine becomes:

- The first layer of architecture
- A semantic constraint on system design
- A decision-support engine for coalition operations
- A continuously learning knowledge system

3. CORE CAPABILITIES

3.1 Atomic Doctrine Model

Every doctrinal artifact is decomposed into structured objects:

Entity	Example
DoctrineDocument	ADP 3-0
DoctrineSection	Chapter 2
DoctrineElement	Paragraph 2-15
Concept	"Mission Command"
Task	"Conduct offensive operations"
Role	Commander
InformationRequirement	PIR, CCIR
Diagram	Conceptual model image
Definition	Structured glossary entry

Each DoctrineElement must contain:

- Unique ID
- Source nation
- Version
- Effective date
- Content type (paragraph, diagram, task, etc.)
- Related concepts

- Related tasks
- Data implications
- Confidence score (for NLP extraction)

3.2 Cross-National Alignment Engine

The system must:

- Compare US vs UK vs NATO doctrine
- Identify:
- Terminology mismatches
- Structural differences
- Missing tasks
- Authority differences
- Data expectation differences
- Score alignment confidence
- Generate divergence reports

Output Example:

```
US: "Commander owns risk acceptance."
UK: "Risk accepted at formation level."
Divergence Type: Authority Structure
Operational Risk: High
Data Model Impact: Decision Approval Workflow
```

3.3 Doctrine-to-Data Mapping Layer

Each doctrinal element links to:

- Required data entities
- Required attributes
- Reporting requirements
- System touchpoints
- Information flows

This enables:

- Automatic validation of whether a system supports doctrine
- Identification of data-model gaps
- Design guidance for new systems

3.4 Bi-Temporal Tracking

The system tracks:

- Effective time (when doctrine is valid)
- Transaction time (when stored/modified)
- Version lineage
- Diff analysis

Allows:

- Pre-war vs wartime doctrine comparison
- Trend analysis
- Coalition drift monitoring
- Self-reflective learning

3.5 Self-Reflection Layer

Using analytics and ML:

- Identify frequently conflicting areas
- Detect doctrinal ambiguity
- Suggest harmonization opportunities
- Highlight stale sections
- Surface inconsistencies within a single nation's doctrine

4. SYSTEM ARCHITECTURE OVERVIEW

Layers

1. Ingestion Layer
2. Decomposition & NLP Layer
3. Ontology & Knowledge Graph Layer
4. Alignment & Reasoning Engine
5. Doctrine-to-Data Integration Layer
6. Analytics & Insight Layer
7. Interface Layer (Dashboards, APIs)

Mapping to ODT Workspace

```
GDAP Layer                          ODT Workspace
──────────────────────────────────  ──────────────────────────────────
1. Ingestion                        pipelines/ +
architecture/cda/tools/naf_extractor.py
2. Decomposition & NLP              architecture/cda/tools/ (extend naf_extractor)
3. Ontology & Knowledge Graph       architecture/mim + ddof + Operational-Capability-
Ontology
4. Alignment & Reasoning            architecture/cda/ (new: alignment engine)
5. Doctrine-to-Data Integration     architecture/ ↔ products/ bridge
6. Analytics & Insight              products/platform (ODE) + products/chase
7. Interface                        products/ (all apps) + architecture/cda/apps/ +
sites/
```

5. PIPELINE STEPS

Phase 1: Doctrine Ingestion & Atomization

Step 1 — Document Acquisition - Import PDFs, HTML, Word docs - Store raw artifacts - Extract metadata (nation, version, date)

Step 2 — Structural Parsing - Detect chapters, paragraph numbering, tables, diagrams - Preserve hierarchy

Step 3 — Atomic Decomposition Break into atomic DoctrineElements: - Paragraph-level segmentation - Definition extraction - Task extraction - Role extraction - Authority statements - Conditional statements

Step 4 — Content Typing Classify each atomic unit: - Definition, Task, Authority, Process, Information Requirement, Concept, Constraint

Step 5 — Ontology Mapping Map each extracted element to: - Canonical concept dictionary - Cross-nation shared ontology - Warfighting function taxonomy - METL references

Phase 2: Cross-National Alignment

Step 6 — Semantic Embedding Generation Generate embeddings for concepts, tasks, authority statements.

Step 7 — Cross-Nation Similarity Matching Compute concept similarity scores, structural alignment, authority alignment, process equivalence.

Step 8 — Divergence Classification Label divergence types: terminology difference, structural difference, authority mismatch, missing capability, information expectation mismatch.

Step 9 — Risk Scoring Score impact: operational, command relationship, data model impact, system workflow impact.

Phase 3: Doctrine-to-Data Mapping

Step 10 — Data Entity Extraction From doctrine: identify information requirements, extract implied data attributes, identify decision checkpoints.

Step 11 — Data Model Alignment Compare against existing system schemas, Foundry ontologies, enterprise data models.

Step 12 — Gap Detection Output: missing fields, incompatible structures, overloaded semantics, non-aligned identifiers.

Phase 4: Temporal & Self-Learning Layer

Step 13 — Version Tracking Store diff history, structural change detection, concept drift analysis.

Step 14 — Drift Detection Detect doctrinal convergence, divergence over time, emerging patterns.

Step 15 — Recommendation Engine Suggest harmonization proposals, data schema updates, policy review triggers, joint training alerts.

Phase 5: Exploitability Analysis (DVEE)

Step 16 — Authority Graph Construction Build a graph of roles, decision rights, escalation paths, cross-nation approval dependencies.

Step 17 — Latency Modeling Simulate decision chain length, required inputs, escalation steps. Calculate estimated minimum action time, coalition variance.

Step 18 — Adversarial Edge Analysis Identify predictable triggers, detect fixed escalation thresholds, detect mandatory pauses, identify information requirements that can be denied or degraded.

Step 19 — Coalition Seam Detection Cross-nation compare authority levels, flag asymmetry seams.

Step 20 — Exploitation Risk Scoring Score across operational impact, time sensitivity, legal rigidity, coalition divergence, adversary likelihood of exploitation. Produce Exploitability Index, Coalition Seam Index, Latency Risk Map.

6. DOCTRINE VULNERABILITY & EXPLOITABILITY ENGINE (DVEE)

Data Model Extensions

Add structured tagging to DoctrineElements:

Field	Description
AuthorityLevel	Who decides
EscalationRequired	Boolean
EscalationTier	Level required
TimeSensitivity	Implied time window
RiskAcceptanceLevel	Explicit/implicit
LegalTrigger	Yes/No
InformationDisclosureConstraint	Classification boundary
CrossDomainDependency	Required other function
DecisionLatencyEstimate	Derived metric

Vulnerability Dimensions

1. **Latency Vulnerability** — Where doctrine introduces delay: multi-level approval chains, legal review requirements, coalition concurrence requirements
2. **Authority Rigidity** — Centralized vs delegated authority, conflict in coalition command structure, ambiguous authority language
3. **Threshold Gaming** — Defined escalation triggers, reporting thresholds, explicit quantitative triggers. Adversaries can operate just below these.
4. **Information Bottlenecks** — Required data before action, cross-domain coordination dependencies, over-specified reporting requirements
5. **Policy-Doctrine Mismatch** — Policy stricter than doctrine, doctrine more permissive than law, coalition asymmetry

7. USE CASE DOMAINS

7.1 Operational & Planning

- Joint operation readiness assessment
- Mission-specific doctrine extraction
- Authority simulation
- Contested information environment modeling
- Coalition expansion risk analysis

7.2 Architecture & System Design

- Doctrine-driven system requirements generation
- Schema validation against doctrine
- Data product certification
- Acquisition evaluation

7.3 Training & Education

- Adaptive PME (Professional Military Education)
- Leader preparation briefs
- Coalition pre-deployment training
- Doctrine quiz & assessment engine

7.4 Strategic & Policy

- Doctrine drift detection
- Policy impact forecasting
- Strategic signaling analysis
- Escalation threshold analysis

7.5 Intelligence & Counter-Intelligence

- Adversary doctrine modeling (Russian, Chinese, Iranian)
- Exploitation targeting
- Red team automation

7.6 Enterprise Governance

- Authority transparency dashboard
- Overlap & redundancy detection
- Bureaucratic efficiency modeling

7.7 AI & Automation

- Autonomous workflow guardrails
- Doctrine-aware agents
- Automated harmonization suggestions

7.8 Historical & Research

- Doctrine evolution mapping
- Conflict outcome correlation
- Institutional learning analysis

7.9 Legal & Compliance

- ROE conflict detection
- Audit readiness

7.10 Economic & Industrial Base

- Defense contractor integration
- Cross-nation industrial interoperability

8. INTERFACES

Operational View

- Coalition readiness dashboard
- Divergence heatmaps
- Authority conflict visualization

Architect View

- Doctrine-to-data mapping explorer
- Schema impact visualizer
- System compatibility index

Strategist View

- Trend over time
- Coalition convergence score
- Risk projection model

9. SUCCESS METRICS

- Reduction in integration rework
- Reduced time to coalition system alignment
- Increased doctrinal harmonization
- Measurable reduction in authority conflict incidents
- System design cycle time reduction

10. FINAL-STATE VISION

Once doctrine is structured, it becomes:

- A **decision graph** — who decides what, when, under what authority
- An **authority graph** — delegation chains, escalation paths, risk ownership
- A **latency graph** — how long decisions take, where bottlenecks form
- An **information dependency graph** — what data must exist before action
- A **coalition seam graph** — where national differences create friction
- An **exploitability graph** — where adversaries find advantage
- An **architecture constraint layer** — what systems must support

This is not doctrine digitization. This is:

A universal model of how institutions intend to behave.

Once behavior is modeled, it can be compared, simulated, stress-tested, hardened, optimized, exploited, and harmonized.

Coalition semantic dominance.