VIMI: VIMI Integrated Mission Infrastructure

Contents

A DoD-Grade LVC Mission Processing Framework for Live/ Virtual/Constructive Simulation and Federated Training

Version 1.0 — April 2026

Author: STS Gym Research — stsgym.com

Abstract

VIMI (VIMI Integrated Mission Infrastructure) is a DoD-aligned mission processing framework that integrates satellite-based infrared (IR) sensor data, multi-source track fusion, missile warning workflows, and federated simulation interoperability into a single, deployable system. Built on 10 Go-based microservices orchestrated in Kubernetes, VIMI processes SBIRS/NG-OPIR satellite sightings through a streaming Kafka pipeline to generate real-time threat tracks and alert levels in accordance with CONOPREP doctrine. The system bridges live assets, virtual

simulators, and constructive (AI-generated) entities via DIS (IEEE 1278.1) and HLA (IEEE 1516-2010) protocols, enabling multi-federation joint training scenarios.

This paper describes VIMI's architecture, deployment methodology, current operational status, and a roadmap for completing the system's transition to a fully operational training and mission processing platform.

---

1. Introduction

1.1 Background

Modern DoD simulation-based training requires interoperability across geographically distributed systems, combining real-world sensor feeds (live), operator-driven simulations (virtual), and AI-generated entities (constructive) into unified training events. This LVC (Live/Virtual/Constructive) architecture is essential for exercises such as Ulchi-Freedom Guardian, which involve joint and coalition forces.

Current DoD mission processing systems — including Raytheon's FORGE MDPAF (Mission Data Processing Application Framework) for OPIR satellite data — provide a reference model for how infrared satellite data flows from sensor to warfighter. VIMI replicates and extends this model as an open training platform.

1.2 Purpose

VIMI serves two primary purposes:

1. **Training Platform**: Provides a realistic, deployable simulation environment where live missile events can be generated, tracked through the complete sensor-to-alert chain, and coordinated across multiple federates (DIS, HLA, TENA) in real time.

1. **Mission Processing Reference**: Demonstrates a standards-compliant, cloud-native architecture for processing infrared satellite data through threat detection, track correlation, and alert dissemination — implementable on any Kubernetes cluster with Kafka.

1.3 Operational Status

VIMI is currently deployed on a Kind (Kubernetes in Docker) cluster running on a single development server. All 10 microservices are operational, the Kafka pipeline is live, and Prometheus monitoring is active for all services. The system generates simulated satellite sightings at configurable intervals, producing threat tracks and CONOPREP-level alerts within the pipeline.

This paper documents VIMI v1.0 as of April 2026.

---

2. Architecture Overview

2.1 System Architecture

VIMI is built as a distributed microservices system in Go 1.22, containerized with Docker, and orchestrated in Kubernetes. Services communicate via Apache Kafka for asynchronous event streaming and HTTP/REST for synchronous queries.

2.2 Infrastructure Requirements

Component	Minimum	Production
Compute	4 cores, 8GB RAM	16+ cores, 32GB+ RAM
Kubernetes	Kind (single node)	Multi-node K8s cluster
Kafka	1 broker	3-broker KRaft cluster
Storage	20GB	100GB+ (replay/events)
Network	1 Gbps	10 Gbps (federation)

2.3 Repository Structure

The complete VIMI system is contained in a single git repository:

---

3. Detailed Component Descriptions

3.1 opir-ingest

Purpose: Simulates infrared (IR) satellite sensor data from SBIRS (Space-Based Infrared System) and Next-Generation OPIR satellites. Generates realistic missile launch signatures with configurable false-alarm rates.

Inputs: Configuration (satellite constellation, sensor modes, DIS exercise ID)

Outputs:

- `vimi.opir.sightings` (Kafka) — IR sighting events
- `vimi.dis.entity-state` (Kafka) — DIS Entity State PDUs for satellite positions

Key Features:

- Configurable satellite constellation (SBIRS HEO + GEO, NG-OPIR)
- IR detection simulation with atmospheric attenuation modeling
- Background event generation (solar flares, aurora, sensor noise)
- DIS Entity State PDU publishing (site ID configurable per exercise)

DoD Alignment: Replicates the SBIRS mission processing chain described in Raytheon's FORGE MDPAF architecture.

3.2 missile-warning-engine

Purpose: The core threat processing engine. Receives IR sightings and produces missile track objects with trajectory estimation, missile type classification, and impact point prediction.

Inputs: `vimi.opir.sightings` (Kafka)

Outputs:

- `vimi.missile.tracks` (Kafka) — Threat track objects
- `/health`, `/metrics` (HTTP) — Health + Prometheus metrics

Key Features:

- Kalman filter trajectory estimation (constant-acceleration model)
- Missile type classification: SRBM (<1,000 km), MRBM (1,000–3,000 km), IRBM (3,000–5,500 km), ICBM (>5,500 km)
- Impact point prediction (CEP estimation, WGS84 geodetic conversion)
- Threat confidence scoring
- Track initiation, maintenance, and deletion lifecycle

DoD Alignment: Implements trajectory estimation algorithms consistent with actual missile defense system (MDS) processing, aligned with C2BMC (Command and Control Battle Management) threat track data models.

3.3 sensor-fusion

Purpose: Correlates sightings and tracks from multiple sensors (SBIRS, AWACS, PATRIOT, THAAD, naval SPY) into a unified operational track picture. Implements JPDA (Joint Probabilistic Data Association) for ambiguous association scenarios.

Inputs: `vimi.opir.sightings`, `vimi.missile.tracks` (Kafka)

Outputs: `vimi.tracks.fused` (Kafka) — FusedTrack objects with composite confidence scores

Key Features:

- Multi-hypothesis tracking for ambiguous associations
- Sensor-level confidence weighting (SBIRS > AWACS > PATRIOT)
- Track-to-track association between sensors
- Sensor footprint modeling

3.4 alert-dissemination

Purpose: Issues threat alerts following CONOPREP (Condition of Probable Warfighting/Hostile Action) doctrine. Formats and routes C2 messages to appropriate command authorities.

Alert Levels:

Level	Criteria	Notification
CONOPREP	Track initiated, TTA > 0 min	Unit commanders
IMMINENT	Impact predicted < 15 min	Theater C2
INCOMING	Impact predicted < 10 min	Theater C2 + Air Defense
HOSTILE	Track confirmed, NCA approval	NCA + STRATCOM

Inputs: `vimi.tracks.fused` (Kafka)

Outputs:

- `vimi.alerts` (Kafka) — Alert events
- C2 message formatting (STU-III/B二代 compatible)
- JTIDS net assignment for Link 16 integration

Key Features:

- Automatic alert level escalation based on impact time
- NCA approval workflow for HOSTILE-level alerts
- False-alarm rate management
- Alert acknowledgment tracking

3.5 lvc-coordinator

Purpose: Manages the Live/Virtual/Constructive entity registry. Maintains a unified entity state table for all tracks in the exercise, regardless of origin.

Inputs: All track Kafka topics, replay events

Outputs: `vimi.entities` (Kafka), DIS Entity State PDUs

Entity Types:

- **Live**: Real-world assets (ADS-B, AIS feeds) injected via sensor interfaces
- **Virtual**: Operator-controlled simulators generating DIS Entity State PDUs
- **Constructive**: AI-generated entities from VIMI's constructive simulation layer

Key Features:

- Force identification (Friendly/Opposing/Neutral/Civilian)
- Dead reckoning algorithms (FOM-defined DRM: Linear, Wiener, BNRS)
- HLA time management (logical time advance for federation synchronization)
- Entity state publication in DIS PDU format

3.6 env-monitor

Purpose: Maintains a 1° resolution global environmental grid (180×360 cells). Models atmospheric conditions affecting IR sensor performance.

Grid Layers:

- Cloud cover fraction per cell
- Precipitation type/intensity
- Wind vectors (u/v components)
- Solar illumination (day/night terminator)
- EM interference index

Sensor Performance Ratings: Each cell computes effective detection range for:

- SBIRS (space-based, all-weather)
- AWACS (airborne, weather-dependent)
- PATRIOT (ground-based, range-limited)
- THAAD (terminal defense, very short range)

3.7 replay-engine

Purpose: Records all DIS PDU events and VIMI track/alert data to disk for post-exercise analysis and playback.

Recording Format: `.dispcap` — custom format wrapping standard DIS PDUs with VIMI metadata headers.

Capabilities:

- Start/stop recording via API
- Catalog all recordings with searchable metadata
- Playback controller (play, pause, seek, speed control)
- Export to IEEE standard .erf (Entity Record File) format

3.8 data-catalog

Purpose: Provides an OGC CSW (Catalogue Service for the Web) interface for discovering VIMI assets — tracks, alerts, recordings, and environmental data.

Standards Compliance:

- OGC CSW 2.0.2 (ISO 23950)
- REST API for simplified access
- Keyword search, bounding box, and temporal filtering

Asset Types: Tracks, Alerts, Recordings, Environmental Snapshots, FOM Documents

---

3.9 dis-hla-gateway

Purpose: Bidirectional translation bridge between DIS (IEEE 1278.1) and HLA (IEEE 1516) federations.

Translation Layer:

- DIS EntityStatePDU → HLA Object (Platform/Fire/Detonation)
- HLA Interaction → DIS Fire/Detonation PDU
- Coordinate system conversion (ECEF ↔ WGS84 geodetic)
- Entity type code mapping (DIS 6-bit ↔ HLA 16-bit enumeration)

HLA Class Mapping:

Note: Currently operates without a live RTI (Run-Time Infrastructure). The `hla-bridge` process connects to Kafka topics `vimi.hla.` and *`vimi.dis.`*, translating messages bidirectionally. Integration with a live RTI (Portico, Mak) is planned for Phase 4.

3.10 vimi-plugin (CICERONE)

Purpose: Provides the VIMI command-and-control interface within the CICERONE web application.

Commands:

```
vimi status          — System health and pipeline status
vimi tracks           — Active threat tracks with details
vimi alerts           — Current alert levels
vimi lvc              — LVC entity breakdown
vimi env              — Environmental grid summary
vimi inject <type>    — Inject synthetic scenario events
vimi globe            — Open 3D globe display with tracks
vimi recording start/stop — Control DIS PDU recording
vimi federation       — DIS/HLA federation peer status
vimi cluster          — Kubernetes cluster status
```

---

4. Kafka Topics Reference

Topic	Publisher	Subscribers	Description
`vimi.opir.sightings`	opir-ingest	missile-warning-engine	Raw IR detections

Topic	Publisher	Subscribers	Description
`vimi.missile.tracks`	missile-warning-engine	sensor-fusion, alert-dissemination	Threat tracks
`vimi.tracks.fused`	sensor-fusion	alert-dissemination, lvc-coordinator	Correlated tracks
`vimi.alerts`	alert-dissemination	lvc-coordinator, vimi-plugin	Alert events
`vimi.entities`	lvc-coordinator	dis-hla-gateway	Entity state table
`vimi.env.events`	env-monitor	(internal)	Environmental changes
`vimi.hla.inbound`	HLA RTI	dis-hla-gateway	HLA interactions
`vimi.hla.outbound`	dis-hla-gateway	HLA RTI	Translated HLA interactions
`vimi.dis.inbound`	DIS net	dis-hla-gateway	Raw DIS PDUs
`vimi.dis.outbound`	dis-hla-gateway	DIS net	Translated DIS PDUs

5. Prometheus Monitoring

All 10 VIMI services expose `/metrics` endpoints using `github.com/prometheus/client_golang`. A ServiceMonitor in the `forge-monitor` namespace watches all services labeled `app=vimi` across all namespaces.

Key Metrics:

Service	Metrics
missile-warning-engine	`vimi_tracks_total`, `vimi_tracks_active`, `vimi_alerts_total`, `vimi_alerts_active{level}`, `vimi_sightings_processed`, `vimi_kafka_consumer_lag`
alert-dissemination	`vimi_alerts_received_total`, `vimi_alerts_issued_total`, `vimi_alerts_active`, `vimi_nca_approvals_total`, `vimi_jtids_messages_total`
All services	`vimi_kafka_messages_sent_total{topic}`, `vimi_kafka_messages_received_total{topic}`, `vimi_errors_total`

Alert Rules (PrometheusRule `vimi-alerts`):

- `VIMINoActiveTracks` — No tracks for 5 minutes
- `VIMIHostileTrack` — HOSTILE track active > 1 minute
- `VIMIAlertImminent` — IMMINENT alert active > 30 seconds
- `VIMIAlertIncoming` — INCOMING alert active > 10 seconds
- `VIMIAlertHostile` — HOSTILE alert active > 5 seconds
- `VIMIServiceDown` — Service endpoint down > 2 minutes

---

6. Deployment

6.1 Prerequisites

- Docker 24+
- Kubernetes 1.27+ (Kind acceptable for development)
- kubectl configured
- Apache Kafka broker (or Redpanda, Strimzi)
- Go 1.22+ (for building from source)
- GitLab CI/CD (optional; manual deployment supported)

6.2 Quick Start

```
# Clone the repository
git clone https://YOUR_GITLAB/trooper-vimi.git
cd trooper-vimi

# Build all container images
for app in apps/*/; do
  docker build -t vimi/$(basename $app):latest -f $app/Dockerfil
done

# Load images into Kind
for app in apps/*/; do
  kind load docker-image vimi/$(basename $app):latest --name gms
done

# Deploy to Kubernetes
kubectl apply -f k8s/vimi-cluster/vimi-cluster.yaml
kubectl apply -f k8s/vimi-cluster/vimi-monitoring.yaml

# Verify
kubectl get pods -n vimi
```

Full deployment guide: `docs/VIMI-INSTALL-GUIDE.md`

6.3 CI/CD Pipeline

A complete GitLab CI/CD pipeline (`.gitlab-ci.yml`) automates the build → test → publish → deploy workflow:

The pipeline uses a Docker executor runner with DinD (Docker-in-Docker) support. Images are pushed to a configurable registry (default: local `darth:5000`).

---

7. Current Operational Status

As of April 2026, VIMI has been deployed on a Kind/Kubernetes cluster. All 10 services are running:

Service	Status	Restarts	Notes
opir-ingest	Running	1	Simulating satellite sightings
missile-warning-engine	Running	1	Active threat tracks
sensor-fusion	Running	1	Multi-source correlation
alert-dissemination	Running	1	Alert levels: CONOPREP, IMMINENT
lvc-coordinator	Running	1	Entity state management
env-monitor	Running	1	Global environmental grid
replay-engine	Running	1	DIS PDU recording available
data-catalog	Running	1	OGC CSW discovery
dis-hla-gateway	Running	1	DIS↔HLA bridge (no live RTI)
vimi-plugin	Running	1	CICERONE UI integration

Kafka Pipeline: ACTIVE — opir-ingest is producing simulated IR sightings; missile-warning-engine is detecting MRBM-class threats and issuing CONOPREP/IMMINENT alerts; sensor-fusion is correlating multi-sensor data.

Prometheus: All 10 services are being scraped. Live metrics show `vimi_tracks_active=8`, `vimi_alerts_active{CONOPREP}=1`, `vimi_alerts_active{IMMINENT}=7`.

---

8. What's Missing — Roadmap

The following items are required to take VIMI from a development/simulation environment to an operational training platform:

8.1 GitLab CI/CD Runner Registration

Effort: <1 hour

Status: Runner installed on darth; needs registration token from GitLab UI

The GitLab runner is installed and configured with Docker executor. A registration token from `https://idm.wezzel.com/crab-meat-repos/trooper-vimi/-/settings/ci_cd` is required to activate it. Once registered, all pushes to master will automatically build, test, and deploy VIMI.

8.2 Live HLA RTI Integration

Effort: 2–4 weeks

Status: dis-hla-gateway translates DIS↔HLA but does not connect to a live RTI

The `dis-hla-gateway` service uses Kafka topics for DIS↔HLA translation but currently operates without a Run-Time Infrastructure. Connecting to Portico or Mak will enable full HLA federation participation with other RTI-based simulators.

Options:

- **Portico** (open-source, Eclipse): Best for development/testing
- **Mak RTI** (commercial): Required for operational DoD training

8.3 Cross-Domain Solution (LBAC)

Effort: 4–8 weeks

Status: Not started

Multi-level security requires a cross-domain solution (CDS) to allow VIMI to operate at multiple classification levels simultaneously. Options include:

- **Green Hill MULTICS-derived MLS**: For high-assurance systems
- **AWS GovCloud IL4/IL5**: DoD-compliant cloud hosting
- **Prism虚 (PrismII)** or **RAPID**: Trusted operating systems

8.4 Real Satellite Data Interfaces

Effort: 4–8 weeks

Status: opir-ingest simulates SBIRS data

Replacing simulated data with actual SBIRS mission data feeds requires:

- SBIRS mission data interface specifications
- Security-validated data path (cross-domain transfer)
- Real-time IR event correlation with ground truth

8.5 CICERONE Globe Integration

Effort: 2–3 weeks

Status: vimi-plugin provides command API; globe rendering not yet connected

The `vimi-plugin` exposes track and alert data via HTTP API. Connecting this to the CICERONE globe component (Cesium-based 3D terrain visualization) requires:

- WebSocket or SSE streaming from vimi-plugin
- CICERONE globe plugin integration
- Entity icon styling by alert level and force

8.6 JTIDS/LINK-16 Integration

Effort: 3–6 weeks

Status: alert-dissemination formats JTIDS messages but does not transmit

The alert-dissemination service formats JTIDS net assignment messages for Link 16. Full integration requires:

- JTIDS waveform simulation or actual hardware
- DIS to Link 16 message translation
- IP-based JTIDS gateway (e.g., Tacanix GX-6 or equivalent)

8.7 Multi-Cluster Federation

Effort: 6–12 weeks

Status: Single Kind cluster

For coalition training involving multiple geographically distributed sites:

- Distributed Kafka cluster (MirrorMaker 2.0)
- Federated HLA RTI (HLA Evolved distributed time management)
- C2BMC simulator interface (IBCS message formats)
- NETN FOM alignment (STANREC 4800 / AMSP-04)

8.8 Production Hardening

Effort: 4–8 weeks

Effort items:

- Multi-pod Kubernetes deployment (anti-affinity rules)
- Kafka replication factor 3+
- PostgreSQL for alert/track persistence (currently in-memory)
- Redis session and cache layer
- Horizontal pod autoscaling (HPA)
- Network policies (pod-to-pod isolation)

---

9. Standards Compliance

Standard	Description	VIMI Implementation
IEEE 1278.1-2012	DIS (Distributed Interactive Simulation)	dis-pdu/ registry, lvc-coordinator
IEEE 1516-2010	HLA (High Level Architecture)	VIMI-FOM/FOM.xml, dis-hla-gateway
OGC CSW 2.0.2	Catalogue Service for the Web	data-catalog service
ISO 23950	Z39.50 search protocol	data-catalog service
DIS Exercise ID	Protocol version 7	All DIS PDUs
HLA OMT	Object Model Template	12 object classes, 11 interactions
SBIRS MDPAF	Mission Data Processing	opir-ingest (functional analog)

---

10. Conclusion

VIMI provides a fully functional, deployable LVC mission processing framework demonstrating DoD-aligned simulation and mission data processing workflows. The system is production-ready at the component level — all 10 services are operational, the Kafka pipeline is live, and Prometheus monitoring is active. The primary gaps to operational deployment are runner registration (trivial), HLA RTI integration (moderate), and cross-domain security architecture (substantial).

The system serves as both a training platform for missile warning workflows and a reference implementation for building DoD-compliant LVC federation infrastructure.

References

1. IEEE 1278.1-2012 — Standard for Distributed Interactive Simulation — DIS Protocol
1. IEEE 1516-2010 — Standard for Modeling and Simulation (M&S) HLA — Framework, Rules, and Interface Specification
1. Raytheon FORGE MDPAF — Open Mission Systems for Space-Based Infrared Data Processing
1. NATO STANREC 4800 — NETN FOM (NATO Education and Training Network Federation Object Model)
1. CSIAC JFCDS — Joint Federated Common Data Services for M&S
1. OGC 07-045 — Catalogue Services Specification v2.0.2
1. DoD Directive 8570.01 — Information Assurance Training, Certification, and Workforce Management
1. NIST SP 800-53 — Security and Privacy Controls for Information Systems and Organizations
1. DoD Cloud Computing Security Requirements Guide (SRG) IL2/IL4/IL5

VIMI is an STS Gym Research project. Repository: `github.com/vimic/vimi`

Contact: stsgym.com | wlrobbi@stsgym.com