System Architecture

CJADC2 Orchestration Platform

Version: 1.0 **Author:** John Sasser **Status:** Draft for Review

Table of Contents

1. System Overview
2. Component Architecture
3. Agent Descriptions
4. Data Flow
5. Infrastructure
6. Deployment Topology
7. Appendix: Glossary

1. System Overview

The CJADC2 platform implements an **event-driven architecture** for command and control decision support. The system processes sensor detections through a pipeline of specialized agents, each performing a distinct function in the intelligence-to-action workflow.

Pipeline Flow: Sensor → Classifier → Correlator → Planner → Authorizer → Effector

Design Principles

PRINCIPLE	DESCRIPTION
Event Sourcing	All state changes are captured as immutable events in NATS JetStream
Separation of Concerns	Each agent has a single responsibility in the processing pipeline
Policy as Code	Authorization logic externalized to OPA with Rego policies
Human-in-the-Loop	All consequential actions require explicit human approval
Observability First	Full distributed tracing and metrics from day one

2. Component Architecture

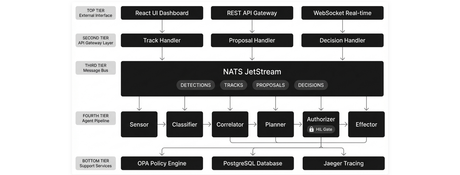

Figure 2.1: Four-tier system architecture with external interfaces, API gateway, message bus, and agent pipeline

Technology Stack

COMPONENT	TECHNOLOGY	PURPOSE
Messaging	NATS JetStream 2.10	Store-and-forward pub/sub with exactly-once delivery
Policy Engine	OPA 0.60 / Rego	Declarative policy enforcement at each pipeline stage
Persistence	PostgreSQL 16	Decisions, effects, tracks, and audit trail
Agents	Go (BaseAgent framework)	High-performance event processors with lifecycle management
Frontend	React + TypeScript	Real-time operator dashboard with WebSocket updates
Observability	Prometheus + Jaeger	Metrics collection and distributed tracing

3. Agent Descriptions

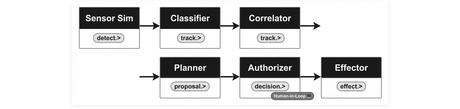

Figure 3.1: Agent pipeline with Human-in-the-Loop approval gate

AGENT	INPUT	OUTPUT	RESPONSIBILITY
Sensor	Timer|Config	`detect.{sensor_id}.{type}`	Generate synthetic detections with position, velocity, and confidence
Classifier	`detect.>`	`track.classified.{class}`	Enrich with classification (friendly|hostile|unknown) and type
Correlator	`track.classified.>`	`track.correlated.{threat}`	Fuse duplicate tracks, assign threat levels (10s window)
Planner	`track.correlated.>`	`proposal.pending.{priority}`	Generate action proposals with rationale, validate against OPA
Authorizer	`proposal.>`	`decision.{status}.{action}`	Present to operator, capture approval/denial, persist to DB
Effector	`decision.approved.>`	`effect.{status}.{action}`	Execute with idempotency protection, log for audit

> **Human-in-the-Loop Gate**
>
> The Authorizer agent enforces the critical safety constraint: **no ActionProposal proceeds to Effector without explicit human approval**. Proposals expire if not actioned within the configured timeout.

4. Data Flow

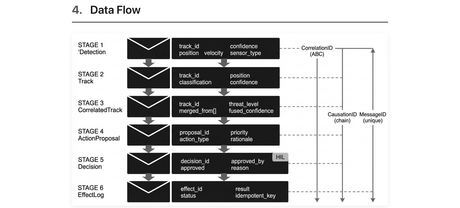

Figure 4.1: Message envelope transformation through the pipeline with correlation tracing

Message Correlation

Every message carries an envelope with correlation fields enabling end-to-end tracing:

FIELD	PURPOSE	EXAMPLE
CorrelationID	Links all messages from a single originating detection	`corr-ABC`
CausationID	References the immediate parent message	`msg-001`
MessageID	Unique identifier for this specific message (UUIDv7)	`msg-002`

5. Infrastructure

NATS Stream Configuration

STREAM	SUBJECTS	RETENTION	MAX AGE
DETECTIONS	`detect.>`	Limits	24h
TRACKS	`track.>`	Limits	72h
PROPOSALS	`proposal.>`	WorkQueue	1h
DECISIONS	`decision.>`	Limits	7d
EFFECTS	`effect.>`	Limits	30d

OPA Policy Structure

AGENT	POLICY PATH	PURPOSE
All agents	`cjadc2/origin`	Validate message source attestation
Classifier	`cjadc2/data_handling`	Check data clearance permissions
Planner	`cjadc2/proposals`	Validate proposal rules and constraints
Effector	`cjadc2/effects`	Verify approval chain before execution

Performance Targets

METRIC	TARGET	NOTES
End-to-End Latency (p95)	< 500ms	Excluding human decision time
Detections/sec	1,000	Sensor emission rate
Tracks/sec	500	Classifier throughput
Agent Memory	64-256 MB	Per agent instance

6. Deployment Topology

Development Environment

All services run in a single Docker Compose network with the following port assignments:

SERVICE	PORT	PURPOSE
API Gateway	8080	REST API + WebSocket
UI Dashboard	3001	React frontend
NATS	4222 / 8222	Client / Management
PostgreSQL	5432	Database
OPA	8181	Policy evaluation
Prometheus	9090	Metrics
Jaeger	16686	Trace UI

Production Considerations

> **Edge Deployment Pattern**
>
> For disconnected or intermittent connectivity scenarios, agents run locally with NATS Leaf nodes that bridge to the central cluster when connectivity is available. OPA bundles are synced periodically, and PostgreSQL replicas provide local persistence.

COMPONENT	SCALING STRATEGY
Agents	Horizontally scalable via NATS consumer groups
NATS	Clustered with JetStream replication (R3)
PostgreSQL	Read replicas for query load distribution
OPA	Multiple instances with shared bundle server
API Gateway	Load balanced, stateless instances

A. Appendix: Glossary

CJADC2
Combined Joint All-Domain Command and Control

HIL
Human-in-the-Loop approval requirement

OPA
Open Policy Agent for declarative authorization

JetStream
NATS persistence layer for at-least-once delivery

Correlation ID
Unique identifier tracing events through the pipeline

Causation ID
Reference to the immediate parent message

ActionProposal
Suggested response awaiting human approval

Effect
Executed outcome of an approved action

BaseAgent
Go framework providing NATS, metrics, and lifecycle

Rego
Policy language used by OPA