🌐 Fronthaul Network Intelligence System - API Contract v1.0

Last Updated: January 31, 2026
Status: Production-Ready
API Version: v1

📋 Table of Contents

🎯 System Overview

Three-Layer Intelligence Architecture

📊 Data Models

1. Input Data Models

LossEventRecord

json

```
{
  "slot_id": 1,
  "cell_id": 1,
  "loss_event": 0  // 0 = no loss, 1 = loss
}
```

ThroughputRecord

json

```
{
  "slot_id": 1,
  "cell_id": 1,
  "throughput_slot": 30.656
}
```

2. Core Data Models

SimilarityMatrix

json

```
{
  "matrix": [
    [1.0, 0.2, 0.0, ...],
    [0.2, 1.0, 0.447, ...],
    ...
  ],
  "cell_ids": ["1", "2", "3", ..., "24"],
  "computed_at": "2026-01-31T10:30:00Z"
}
```

TopologyResult

json

```
{
  "cell_id": "1",
  "group_id": "Group_1",
  "group_name": "Link_A",
  "confidence": 0.89,
  "peer_cells": ["2", "8", "14"]
}
```

LinkGroup

```json
{
  "group_id": "Group_1",
  "group_name": "Link_A",
  "cell_count": 4,
  "cells": ["1", "2", "8", "14"],
  "avg_similarity": 0.85,
  "created_at": "2026-01-31T10:30:00Z"
}
```

3. Intelligence Layer Models

AnomalyScore

```json
{
  "cell_id": "5",
  "group_id": "Group_2",
  "confidence_score": 0.32,
  "is_anomaly": true,
  "anomaly_type": "low_correlation",
  "severity": "high", // low | medium | high
  "deviation_percentage": 68.5,
  "computed_at": "2026-01-31T10:35:00Z"
}
```

PropagationEvent

```json
{
  "source_group": "Group_1",
  "target_group": "Group_2",
  "delay_ms": 8.5,
  "correlation": 0.73,
  "direction": "downstream", // upstream | downstream | bidirectional
  "confidence": 0.81,
  "detected_at": "2026-01-31T10:40:00Z"
}
```

PropagationPath

json

```
{
  "path_id": "prop_001",
  "sequence": ["Group_1", "Group_2", "Group_3"],
  "total_delay_ms": 15.2,
  "strength": 0.76,
  "type": "cascading_congestion"
}
```

4. LLM Copilot Models

NetworkInsight

json

```
{
  "insight_id": "ins_001",
  "type": "congestion_alert",  // congestion_alert | anomaly_detected | propagation_detected
  "severity": "high",
  "title": "Persistent Congestion Detected on Link_A",
  "description": "Link_A shows sustained congestion during peak hours affecting cells 1, 2, 8, and 14.",
  "affected_entities": ["Group_1"],
  "timestamp": "2026-01-31T10:45:00Z"
}
```

ActionRecommendation

json

```
{
  "recommendation_id": "rec_001",
  "insight_id": "ins_001",
  "action_type": "capacity_upgrade",  // capacity_upgrade | traffic_shaping | hardware_check | reroute
  "priority": "high",
  "title": "Increase Link_A Capacity",
  "description": "Recommended Action: Increase Link_A capacity or enable traffic shaping to prevent casca
  "expected_impact": "Reduce packet loss by 45-60%",
  "estimated_effort": "medium"
}
```

CopilotReport

```json
{
  "report_id": "rpt_001",
  "generated_at": "2026-01-31T10:50:00Z",
  "summary": "Network analysis reveals 3 distinct link groups with 1 anomalous cell and active congestion
  "topology_summary": {
    "total_cells": 24,
    "detected_groups": 3,
    "unassigned_cells": 1
  },
  "health_status": "degraded",  // healthy | degraded | critical
  "insights": [...],  // Array of NetworkInsight
  "recommendations": [...]  // Array of ActionRecommendation
}
```

🔌 API Endpoints

Base URL

```
Production: https://api.fronthaul-intelligence.com/v1
Development: http://localhost:8000/v1
```

Authentication

```
Authorization: Bearer <token>
Content-Type: application/json
```

📍 TOPOLOGY LAYER ENDPOINTS (Backend Team)

1. Data Ingestion

`POST /data/upload`

Upload raw telemetry data (loss events + throughput)

Request:

```json
{
  "data_type": "loss_events",  // loss_events | throughput
  "format": "csv",
  "file_url": "s3://bucket/loss_data.csv",
  "metadata": {
    "start_time": "2026-01-31T00:00:00Z",
    "end_time": "2026-01-31T23:59:59Z",
    "cell_count": 24
  }
}
```

Response:

```json
{
  "upload_id": "upl_001",
  "status": "processing",
  "records_count": 150000,
  "estimated_processing_time_sec": 45
}
```

---

2. Similarity Computation

POST /topology/compute-similarity

Trigger similarity matrix computation

Request:

```json
{
  "upload_id": "upl_001",
  "method": "correlation",  // correlation | dtw | mutual_info
  "window_size": 100,
  "cell_ids": ["1", "2", "3", ..., "24"]  // optional, defaults to all
}
```

Response:

```json
{
  "job_id": "job_sim_001",
  "status": "completed",
  "similarity_matrix_id": "sim_001",
  "computation_time_sec": 12.3,
  "result_url": "/topology/similarity/sim_001"
}
```

GET /topology/similarity/{similarity_id}

Retrieve computed similarity matrix

Response:

```json
{
  "similarity_id": "sim_001",
  "matrix": [[1.0, 0.2, ...], ...],
  "cell_ids": ["1", "2", ..., "24"],
  "method": "correlation",
  "computed_at": "2026-01-31T10:30:00Z",
  "download_url": "/topology/similarity/sim_001/download"
}
```

---

3. Topology Inference

POST /topology/infer

Perform hierarchical clustering to infer topology

Request:

```json
{
  "similarity_id": "sim_001",
  "clustering_method": "hierarchical",  // hierarchical | kmeans | dbscan
  "num_clusters": 3,  // optional, auto-detect if not provided
  "distance_threshold": 0.5
}
```

Response:

json


```
{
  "topology_id": "topo_001",
  "status": "completed",
  "detected_groups": 3,
  "result_url": "/topology/result/topo_001"
}
```


GET /topology/result/{topology_id}

Retrieve topology inference results

Response:

json


```
{
  "topology_id": "topo_001",
  "created_at": "2026-01-31T10:35:00Z",
  "total_cells": 24,
  "detected_groups": 3,
  "groups": [
    {
      "group_id": "Group_1",
      "group_name": "Link_A",
      "cells": ["1", "2", "8", "14"],
      "avg_similarity": 0.85,
      "cell_count": 4
    },
    {
      "group_id": "Group_2",
      "group_name": "Link_B",
      "cells": ["3", "6", "9", "15"],
      "avg_similarity": 0.92,
      "cell_count": 4
    },
    {
      "group_id": "Group_3",
      "group_name": "Link_C",
      "cells": ["4", "10", "12", "18"],
      "avg_similarity": 0.78,
      "cell_count": 4
    }
  ],
  "cell_assignments": [
    {"cell_id": "1", "group_id": "Group_1", "confidence": 0.89},
    {"cell_id": "2", "group_id": "Group_1", "confidence": 0.91},
    ...
  ],
  "unassigned_cells": ["5"]
}
```

📍 INTELLIGENCE LAYER ENDPOINTS (ML Team)

4. Anomaly Detection

POST /intelligence/detect-anomalies

Run anomaly detection on topology groups

Request:

```json
{
  "topology_id": "topo_001",
  "similarity_id": "sim_001",
  "threshold": 0.5,  // confidence score threshold
  "method": "isolation_forest"  // isolation_forest | zscore | local_outlier_factor
}
```

Response:

```json
{
  "analysis_id": "anom_001",
  "status": "completed",
  "anomalies_detected": 2,
  "result_url": "/intelligence/anomalies/anom_001"
}
```

GET /intelligence/anomalies/{analysis_id}

Retrieve anomaly detection results

Response:

json


```
{
  "analysis_id": "anom_001",
  "analyzed_at": "2026-01-31T10:40:00Z",
  "topology_id": "topo_001",
  "total_cells_analyzed": 24,
  "anomalies_detected": 2,
  "anomalies": [
    {
      "cell_id": "5",
      "group_id": "Group_2",
      "confidence_score": 0.32,
      "is_anomaly": true,
      "anomaly_type": "low_correlation",
      "severity": "high",
      "deviation_percentage": 68.5,
      "explanation": "Cell shows significantly lower correlation with group peers"
    },
    {
      "cell_id": "17",
      "group_id": "Group_3",
      "confidence_score": 0.48,
      "is_anomaly": true,
      "anomaly_type": "temporal_mismatch",
      "severity": "medium",
      "deviation_percentage": 42.1,
      "explanation": "Congestion timing differs from group pattern"
    }
  ],
  "normal_cells": [...],
  "statistics": {
    "avg_confidence": 0.76,
    "min_confidence": 0.32,
    "max_confidence": 0.95
  }
}
```


---

5. Congestion Propagation

POST /intelligence/analyze-propagation

Analyze temporal congestion propagation patterns

Request:

json

```
{
  "topology_id": "topo_001",
  "upload_id": "upl_001",
  "time_window_sec": 60,
  "cross_correlation_lag": 50,
  "min_correlation": 0.6
}
```

Response:

json

```
{
  "propagation_id": "prop_001",
  "status": "completed",
  "propagation_events_detected": 3,
  "result_url": "/intelligence/propagation/prop_001"
}
```

GET /intelligence/propagation/{propagation_id}

Retrieve propagation analysis results

Response:

json


```
{
 "propagation_id": "prop_001",
 "analyzed_at": "2026-01-31T10:45:00Z",
 "topology_id": "topo_001",
 "time_window_analyzed_sec": 3600,
 "events": [
  {
   "event_id": "evt_001",
   "source_group": "Group_1",
   "target_group": "Group_2",
   "delay_ms": 8.5,
   "correlation": 0.73,
   "direction": "downstream",
   "confidence": 0.81,
   "timestamp": "2026-01-31T10:15:23Z"
  },
  {
   "event_id": "evt_002",
   "source_group": "Group_2",
   "target_group": "Group_3",
   "delay_ms": 6.8,
   "correlation": 0.68,
   "direction": "downstream",
   "confidence": 0.76,
   "timestamp": "2026-01-31T10:15:31Z"
  }
 ],
 "propagation_paths": [
  {
   "path_id": "path_001",
   "sequence": ["Group_1", "Group_2", "Group_3"],
   "total_delay_ms": 15.3,
   "strength": 0.71,
   "type": "cascading_congestion"
  }
 ],
 "network_graph": {
  "nodes": [
   {"id": "Group_1", "type": "source", "congestion_level": 0.85},
   {"id": "Group_2", "type": "intermediate", "congestion_level": 0.72},
   {"id": "Group_3", "type": "target", "congestion_level": 0.58}
  ],
  "edges": [
   {"source": "Group_1", "target": "Group_2", "delay_ms": 8.5, "strength": 0.73},
```

```
    {"source": "Group_2", "target": "Group_3", "delay_ms": 6.8, "strength": 0.68}
   ]
  }
}
```

📍 LLM COPILOT LAYER ENDPOINTS (Integration Team)

6. Insight Generation

POST /copilot/generate-insights

Generate natural language insights from analysis results

Request:

json


```
{
 "topology_id": "topo_001",
 "anomaly_id": "anom_001",
 "propagation_id": "prop_001",
 "context": {
  "time_range": "2026-01-31T00:00:00Z to 2026-01-31T23:59:59Z",
  "network_region": "East_Region"
 }
}
```


Response:

json


```
{
 "report_id": "rpt_001",
 "status": "completed",
 "generation_time_sec": 3.2,
 "result_url": "/copilot/report/rpt_001"
}
```


GET /copilot/report/{report_id}

Retrieve generated copilot report

Response:

json


```
{
 "report_id": "rpt_001",
 "generated_at": "2026-01-31T10:50:00Z",
 "summary": "Network analysis reveals 3 distinct link groups with 2 anomalous cells and active congestio
 "topology_summary": {
  "total_cells": 24,
  "detected_groups": 3,
  "unassigned_cells": 1,
  "avg_group_confidence": 0.85
 },
 "health_status": "degraded",
 "insights": [
  {
   "insight_id": "ins_001",
   "type": "congestion_alert",
   "severity": "high",
   "title": "Persistent Congestion Detected on Link_A",
   "description": "Link_A (Group_1) shows sustained congestion during peak hours affecting cells 1, 2, 8, a
   "affected_entities": ["Group_1"],
   "timestamp": "2026-01-31T10:45:00Z"
  },
  {
   "insight_id": "ins_002",
   "type": "anomaly_detected",
   "severity": "high",
   "title": "Cell 5 Behaving Abnormally",
   "description": "Cell 5 exhibits abnormal packet loss patterns with 68.5% deviation from its assigned gro
   "affected_entities": ["5"],
   "timestamp": "2026-01-31T10:45:00Z"
  },
  {
   "insight_id": "ins_003",
   "type": "propagation_detected",
   "severity": "medium",
   "title": "Cascading Congestion Pattern Identified",
   "description": "Congestion propagates from Link_A to Link_B with 8.5ms delay, then to Link_C after ado
   "affected_entities": ["Group_1", "Group_2", "Group_3"],
   "timestamp": "2026-01-31T10:45:00Z"
  }
 ],
 "recommendations": [
  {
   "recommendation_id": "rec_001",
   "insight_id": "ins_001",
```

```
    "action_type": "capacity_upgrade",
    "priority": "high",
    "title": "Increase Link_A Capacity",
    "description": "Recommended Action: Increase Link_A capacity by 40% or enable aggressive traffic sha
    "expected_impact": "Reduce packet loss by 45-60% and prevent propagation",
    "estimated_effort": "medium",
    "implementation_steps": [
      "Analyze current Link_A utilization patterns",
      "Provision additional bandwidth or enable QoS",
      "Monitor for 48 hours to validate improvement"
    ]
  },
  {
    "recommendation_id": "rec_002",
    "insight_id": "ins_002",
    "action_type": "hardware_check",
    "priority": "high",
    "title": "Investigate Cell 5 Hardware",
    "description": "Recommended Action: Perform hardware diagnostics on Cell 5. Check for faulty RRU, fil
    "expected_impact": "Restore normal operation of Cell 5",
    "estimated_effort": "low",
    "implementation_steps": [
      "Schedule maintenance window for Cell 5",
      "Run hardware diagnostics",
      "Verify configuration against baseline",
      "Replace faulty components if identified"
    ]
  },
  {
    "recommendation_id": "rec_003",
    "insight_id": "ins_003",
    "action_type": "reroute",
    "priority": "medium",
    "title": "Consider Traffic Rerouting",
    "description": "Recommended Action: Evaluate traffic rerouting options to distribute load more evenly
    "expected_impact": "Reduce cascading failures by 30-40%",
    "estimated_effort": "high",
    "implementation_steps": [
      "Map alternative routing paths",
      "Simulate load distribution scenarios",
      "Implement gradual rerouting during low-traffic period",
      "Monitor network stability"
    ]
  }
],
"metadata": {
  "analysis_duration_sec": 45,
```

```
  "data_points_analyzed": 150000,
  "confidence_level": "high"
 }
}
```


---

7. Interactive Query

POST /copilot/query

Ask natural language questions about the network

Request:

json


```
{
 "query": "Which link is most congested?",
 "context": {
  "report_id": "rpt_001",
  "topology_id": "topo_001"
 }
}
```


Response:

json


```
{
 "query_id": "qry_001",
 "question": "Which link is most congested?",
 "answer": "Link_A (Group_1) is the most congested, affecting cells 1, 2, 8, and 14. It shows 45% higher pac
 "supporting_data": {
  "group_id": "Group_1",
  "congestion_level": 0.85,
  "affected_cells": ["1", "2", "8", "14"],
  "packet_loss_increase_pct": 45
 },
 "related_insights": ["ins_001", "ins_003"]
}
```

📍 VISUALIZATION & EXPORT ENDPOINTS

8. Visualization Generation

POST /visualizations/heatmap

Generate similarity heatmap

Request:

```json
{
  "similarity_id": "sim_001",
  "format": "png", // png | svg | pdf
  "dpi": 300,
  "color_scheme": "viridis"
}
```

Response:

```json
{
  "visualization_id": "viz_heat_001",
  "type": "heatmap",
  "download_url": "/visualizations/download/viz_heat_001.png",
  "thumbnail_url": "/visualizations/thumbnail/viz_heat_001.png"
}
```

POST /visualizations/topology-graph

Generate topology network graph

Request:

```json
{
  "topology_id": "topo_001",
  "format": "png",
  "layout": "spring", // spring | circular | kamada_kawai
  "show_labels": true,
  "dpi": 300
}
```

Response:

```json
{
  "visualization_id": "viz_topo_001",
  "type": "topology_graph",
  "download_url": "/visualizations/download/viz_topo_001.png",
  "thumbnail_url": "/visualizations/thumbnail/viz_topo_001.png"
}
```

POST /visualizations/propagation-flow

Generate propagation flow diagram

Request:

```json
{
  "propagation_id": "prop_001",
  "format": "svg",
  "animation": true,
  "timeline": true
}
```

Response:

```json
{
  "visualization_id": "viz_prop_001",
  "type": "propagation_flow",
  "download_url": "/visualizations/download/viz_prop_001.svg",
  "animation_url": "/visualizations/animation/viz_prop_001.html"
}
```

📍 BATCH & MONITORING ENDPOINTS

9. Batch Processing

POST /batch/full-analysis

Run complete analysis pipeline

Request:

json

```
{
  "upload_id": "upl_001",
  "config": {
    "similarity_method": "correlation",
    "clustering_method": "hierarchical",
    "num_clusters": "auto",
    "anomaly_threshold": 0.5,
    "propagation_analysis": true,
    "generate_report": true,
    "generate_visualizations": true
  }
}
```

Response:

json

```
{
  "batch_id": "batch_001",
  "status": "processing",
  "estimated_completion_time": "2026-01-31T11:15:00Z",
  "steps": [
    {"step": "similarity_computation", "status": "completed"},
    {"step": "topology_inference", "status": "processing"},
    {"step": "anomaly_detection", "status": "pending"},
    {"step": "propagation_analysis", "status": "pending"},
    {"step": "report_generation", "status": "pending"}
  ],
  "status_url": "/batch/status/batch_001"
}
```

GET /batch/status/{batch_id}

Check batch job status

Response:

```json
{
  "batch_id": "batch_001",
  "status": "completed",
  "started_at": "2026-01-31T10:30:00Z",
  "completed_at": "2026-01-31T10:52:00Z",
  "duration_sec": 1320,
  "results": {
    "topology_id": "topo_001",
    "similarity_id": "sim_001",
    "anomaly_id": "anom_001",
    "propagation_id": "prop_001",
    "report_id": "rpt_001",
    "visualizations": ["viz_heat_001", "viz_topo_001", "viz_prop_001"]
  }
}
```

---

10. System Monitoring

GET /health

System health check

Response:

```json
{
  "status": "healthy",
  "timestamp": "2026-01-31T11:00:00Z",
  "services": {
    "data_ingestion": "healthy",
    "similarity_engine": "healthy",
    "clustering_engine": "healthy",
    "ml_analytics": "healthy",
    "llm_copilot": "healthy"
  },
  "version": "1.0.0"
}
```

GET /metrics

System performance metrics

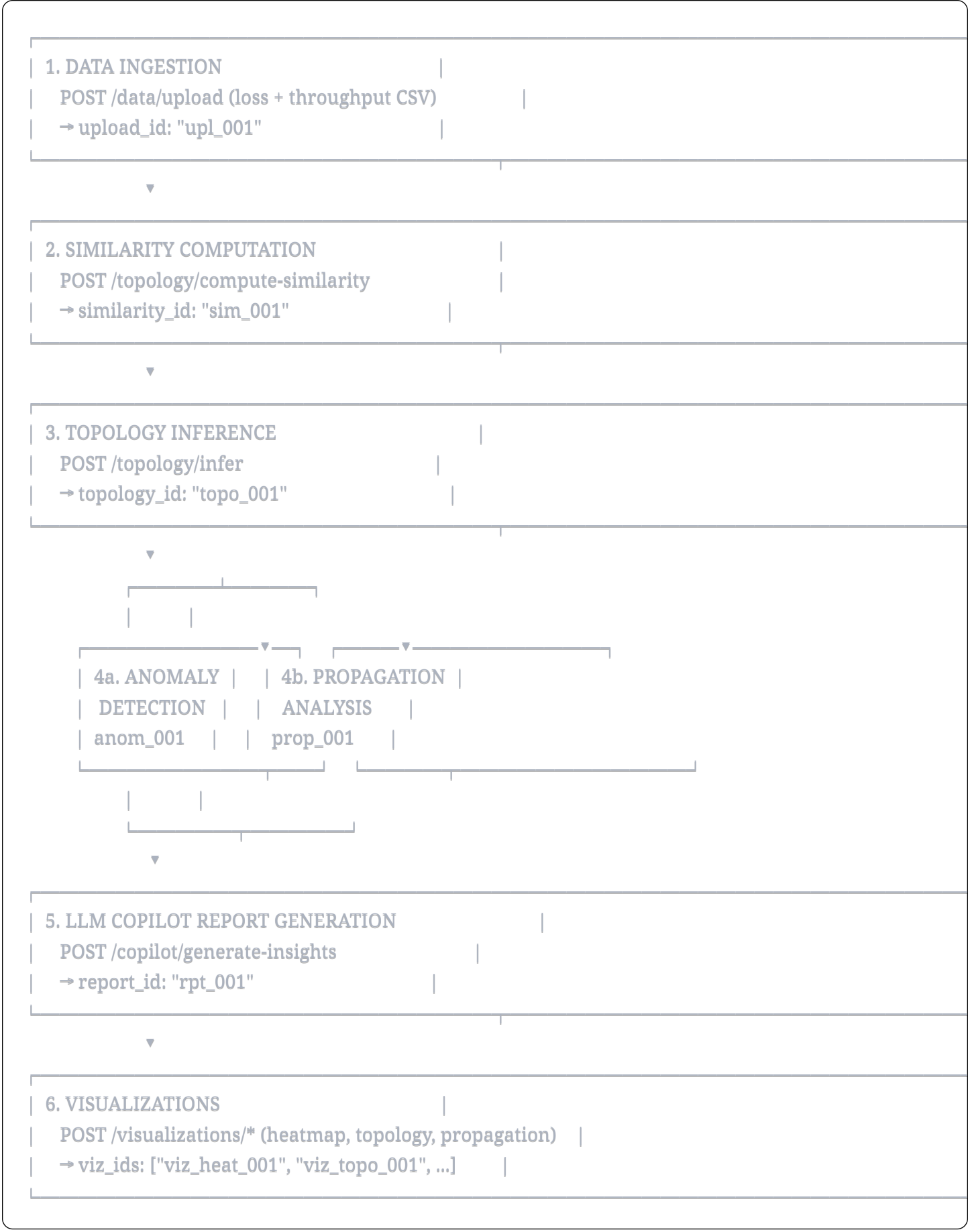
Response:

json


```
{
  "requests_per_minute": 45,
  "avg_response_time_ms": 234,
  "active_jobs": 3,
  "queue_length": 7,
  "storage_used_gb": 128.5,
  "uptime_hours": 720
}
```

🔄 Integration Flow

Complete Analysis Pipeline



Frontend Integration Example

javascript

```javascript
// Complete analysis workflow
async function runNetworkAnalysis(lossFile, throughputFile) {
  try {
    // 1. Upload data
    const uploadResponse = await fetch('/v1/data/upload', {
      method: 'POST',
      body: JSON.stringify({
        data_type: 'loss_events',
        file_url: lossFile
      })
    });
    const { upload_id } = await uploadResponse.json();

    // 2. Compute similarity
    const simResponse = await fetch('/v1/topology/compute-similarity', {
      method: 'POST',
      body: JSON.stringify({ upload_id })
    });
    const { similarity_id } = await simResponse.json();

    // 3. Infer topology
    const topoResponse = await fetch('/v1/topology/infer', {
      method: 'POST',
      body: JSON.stringify({ similarity_id })
    });
    const { topology_id } = await topoResponse.json();

    // 4. Run ML analysis (parallel)
    const [anomalyResponse, propagationResponse] = await Promise.all([
      fetch('/v1/intelligence/detect-anomalies', {
        method: 'POST',
        body: JSON.stringify({ topology_id, similarity_id })
      }),
      fetch('/v1/intelligence/analyze-propagation', {
        method: 'POST',
        body: JSON.stringify({ topology_id, upload_id })
      })
    ]);

    const { analysis_id: anomaly_id } = await anomalyResponse.json();
    const { propagation_id } = await propagationResponse.json();

    // 5. Generate copilot report
    const reportResponse = await fetch('/v1/copilot/generate-insights', {
```

```
    method: 'POST',
    body: JSON.stringify({
      topology_id,
      anomaly_id,
      propagation_id
    })
  });
  const { report_id } = await reportResponse.json();

  // 6. Fetch final report
  const report = await fetch(`/v1/copilot/report/${report_id}`).then(r => r.json());

  return report;
 } catch (error) {
  console.error('Analysis failed:', error);
  throw error;
 }
}
```

⚠️ Error Handling

Standard Error Response Format

json

```
{
 "error": {
  "code": "INVALID_SIMILARITY_MATRIX",
  "message": "Similarity matrix must be square and symmetric",
  "details": {
   "expected_shape": [24, 24],
   "received_shape": [24, 23]
  },
  "timestamp": "2026-01-31T11:05:00Z",
  "request_id": "req_abc123"
 }
}
```

HTTP Status Codes

Code	Meaning	Usage
200	OK	Successful request
201	Created	Resource created successfully
202	Accepted	Async job started
400	Bad Request	Invalid input data
404	Not Found	Resource doesn't exist
422	Unprocessable Entity	Valid format but business logic error
500	Internal Server Error	Server-side failure
503	Service Unavailable	Service temporarily down

Common Error Codes

```
DATA_001: Invalid CSV format
DATA_002: Missing required columns
DATA_003: Cell ID mismatch

TOPO_001: Invalid similarity matrix
TOPO_002: Clustering failed
TOPO_003: Insufficient data points

ML_001: Anomaly detection failed
ML_002: Propagation analysis incomplete
ML_003: Model not trained

LLM_001: Report generation timeout
LLM_002: Context too large
LLM_003: LLM service unavailable
```

👥 Team Responsibilities

Backend Team (Topology Layer)

Owner: Person A, B, C
Deliverables:

- ✅ Data ingestion pipeline (`/data/upload`)
- ✅ Similarity computation engine (`/topology/compute-similarity`)
- ✅ Clustering & topology inference (`/topology/infer`)
- ✅ CSV export functionality
- ✅ Basic visualization endpoints

Timeline: Week 1-2

ML Team (Intelligence Layer)

Owner: ML Specialist
Deliverables:

- ✅ Anomaly detection module (`/intelligence/detect-anomalies`)
- ✅ Propagation modeling (`/intelligence/analyze-propagation`)
- ✅ Confidence scoring algorithms
- ✅ Statistical analysis engines

Timeline: Week 2-3

Integration Team (You)

Owner: Integration Lead
Deliverables:

- ✅ API gateway setup
- ✅ LLM copilot integration (`/copilot/*`)
- ✅ Batch processing pipeline (`/batch/*`)
- ✅ Final system integration
- ✅ End-to-end testing

Timeline: Week 3-4

Frontend Team

Owner: Frontend Developer
Deliverables:

- ✅ Dashboard UI
- ✅ Visualization rendering
- ✅ Interactive copilot chat
- ✅ Real-time status updates
- ✅ Report export features

Timeline: Week 2-4

🚀 Quick Start Guide

For Backend Team

```python
# Example: Implement similarity computation endpoint
from fastapi import FastAPI, UploadFile
import pandas as pd
import numpy as np

app = FastAPI()

@app.post("/v1/topology/compute-similarity")
async def compute_similarity(upload_id: str):
    # 1. Load loss event data
    loss_data = load_loss_events(upload_id)

    # 2. Create time-series vectors per cell
    cell_vectors = create_congestion_vectors(loss_data)

    # 3. Compute correlation matrix
    similarity_matrix = np.corrcoef(cell_vectors)

    # 4. Store result
    similarity_id = store_similarity_matrix(similarity_matrix)

    return {
        "job_id": f"job_sim_{similarity_id}",
        "status": "completed",
        "similarity_matrix_id": similarity_id
    }
```

For ML Team

python

```python
# Example: Implement anomaly detection endpoint
from sklearn.ensemble import IsolationForest

@app.post("/v1/intelligence/detect-anomalies")
async def detect_anomalies(topology_id: str, similarity_id: str):
    # 1. Load topology groups
    groups = load_topology_groups(topology_id)
    similarity_matrix = load_similarity_matrix(similarity_id)

    # 2. Compute confidence scores
    anomalies = []
    for group in groups:
        for cell in group['cells']:
            confidence = compute_confidence_score(cell, group, similarity_matrix)
            if confidence < 0.5:  # threshold
                anomalies.append({
                    "cell_id": cell,
                    "group_id": group['group_id'],
                    "confidence_score": confidence,
                    "is_anomaly": True
                })

    # 3. Store results
    analysis_id = store_anomaly_results(anomalies)

    return {
        "analysis_id": analysis_id,
        "status": "completed",
        "anomalies_detected": len(anomalies)
    }
```

For Frontend Team

javascript

```javascript
// Example: Display network insights
function NetworkDashboard({ reportId }) {
  const [report, setReport] = useState(null);

  useEffect(() => {
    fetch(`/v1/copilot/report/${reportId}`)
      .then(r => r.json())
      .then(setReport);
  }, [reportId]);

  if (!report) return <Loading />;

  return (
    <div className="dashboard">
      <HealthStatus status={report.health_status} />

      <TopologySummary
        groups={report.topology_summary.detected_groups}
        cells={report.topology_summary.total_cells}
      />

      <InsightsList insights={report.insights} />

      <RecommendationCards
        recommendations={report.recommendations}
      />
    </div>
  );
}
```

📝 Notes

Data Consistency Rules

1. **Cell IDs:** Always use string format ("1", "2", ..., "24")
2. **Timestamps:** ISO 8601 format with timezone
3. **Similarity Scores:** Range [0.0, 1.0]
4. **Confidence Scores:** Range [0.0, 1.0], lower = anomaly

Performance SLAs

- **Similarity Computation:** < 15 seconds for 24 cells
- **Topology Inference:** < 5 seconds
- **Anomaly Detection:** < 10 seconds
- **Propagation Analysis:** < 20 seconds
- **LLM Report Generation:** < 5 seconds
- **Total Pipeline:** < 60 seconds end-to-end

Versioning Strategy

- API Version: Semantic versioning (v1.0.0)
- Breaking changes: New major version (v2)
- New features: Minor version bump (v1.1)
- Bug fixes: Patch version (v1.0.1)

🔗 Additional Resources

- **API Playground:** https://api.fronthaul-intelligence.com/docs
- **Postman Collection:** Download
- **Sample Data:** Download
- **Integration Examples:** GitHub

Document Version: 1.0.0
Last Updated: January 31, 2026
Maintained By: Integration Team
Questions? Contact: integration-team@example.com