

I decided to do one more using this new Grim Codex again with pilot -

...

ORIGIN :: LogisticsSupplyChainOrchestra

FACETS_ENABLED [

CoreSpells,
MythologicalOperators,
ZodiacCloths,
PhilosophicalEnhancements,
FusionCloths,
DimensionalCloths

]

//

=====

// LAYER 1: FOUNDATION INFRASTRUCTURE

//

=====

LAYER Foundation {

WRAP Atlas [
Hestara :: StabilityCore,
Taurus :: StructuralIntegrity,
Preserva :: StateCheckpoints

]

CHAIN [

Demetra :: ResourceAllocation → AutoScalingWarehouse,
Fluxa :: DynamicMemoryAllocation → InventoryBufferManagement,
Energos :: ComputeResourceAllocation → ProcessingClusterBalance

]

}

//

=====

// LAYER 2: VISIBILITY & MONITORING

//

=====

```

LAYER Visibility {
  WRAP Aurora [
    Clarivis :: RealtimeMonitoring → ShipmentTracking,
    Apollara :: SystemAnalyticsDashboard → SupplyChainMetrics,
    Insighta :: PredictiveAnalytics → DemandForecasting
  ]

  NEST {
    Griffin :: SurveillanceLayer [
      CHAIN [
        Assistara :: ProactiveSystemAdvisory → AnomalyDetection,
        Oraclia :: PredictiveForecast → InventoryOptimization,
        Aurora-Selene :: PredictiveScheduling → DeliveryWindowEstimation
      ]
    ]
  }
}

//
=====
====
// LAYER 3: ROUTING & ORCHESTRATION
//
=====
====

LAYER Routing {
  WRAP Aquarius [
    Poseida :: DataStreamingArchitecture → RealTimeTrackingFlow,
    Hermesia :: MessageRoutingService → OrderRelayNetwork,
    Hecatia :: ContextualRouting → DynamicPathSelection
  ]

  WRAP Cerulean [
    Minerva-Cerulean :: IntelligentRouting → OptimalPathComputation,
    Labyrinthia :: RecursiveSearchResolve → RouteOptimization,
    Sagittarius :: DistributedMessaging → MultiHubCommunication
  ]

  CHAIN [
    Telek :: RemoteCommandExecution → FleetCoordination,
    Forcea :: DistributedCommandExecution → WarehouseOrchestration,
    Moirae :: ProcessOrchestration → EndToEndLifecycle
  ]
}

```

```

}

//
=====
====
// LAYER 4: RESOURCE MANAGEMENT & ALLOCATION
//
=====
====

LAYER ResourceManagement {
  WRAP Libra [
    Heroica :: ConflictResolution → PriorityAllocation,
    Wuvon :: SelfAdjustingRegulation → AdaptiveLoadBalancing,
    Equilibria :: DynamicSystemTuning → CapacityOptimization
  ]

  CHAIN [
    Decisus :: WorkflowQueue → BatchOrderProcessing,
    Transmutare :: DataConversion → FormatStandardization,
    Bioflux :: ComputeLoadBalancing → ProcessingDistribution
  ]

  WRAP Capricorn [
    Gaiana :: SustainableComputing → GreenLogistics,
    Spirala :: CloudAutoScaling → DemandResponseScaling,
    Demetra :: ResourceGrowth → DynamicCapacityExpansion
  ]
}

//
=====
====
// LAYER 5: SECURITY & COMPLIANCE
//
=====
====

LAYER Security {
  WRAP Cerberus [
    Inferna :: MultiTierFirewall → SegmentedAccessControl,
    Absorbus :: AdaptiveMalwareDefense → ThreatNeutralization,
    Armora :: RapidPeripheralDeployment → SecurityModuleInjection
  ]
}

```

```

CHAIN [
  Countera :: ThreatIdentificationNeutralization → ResponseMapping,
  Vulneris :: SystemVulnerabilityScan → WeakPointAnalysis,
  Medusia :: IntrusionDetectionAlert → ThreatFreeze
]

WRAP Nemean-Lion [
  Fortifera :: AutoHardeningSecurity → AdaptiveDefenseProtocol,
  Defendora :: AutoRecoveryCooldown → LoadBalancerReset,
  Bowsera :: AdaptiveUserAuthentication → TrustBasedIdentity
]
}

//
=====
====
// LAYER 6: WORKFLOW AUTOMATION
//
=====
====

LAYER Automation {
  WRAP Vulcan [
    Hephestus :: BuildAutomation → ProcessAutomation,
    Magica :: EventDrivenAutomation → TriggerBasedExecution,
    Herculia :: WorkflowAutomationSystem → MultiPhaseTaskSequencing
  ]

  CHAIN [
    Crona :: TimeBasedOrchestration → ScheduledDeliveries,
    Chronomanta :: TaskPrioritizationRescheduling → UrgentOrderHandling,
    Persephona :: ScheduledServiceStates → SeasonalCapacityAdjustment
  ]

  WRAP Virgo [
    Sisyphaea :: ContinuousServiceMonitor → OngoingQualityCheck,
    Laborina :: IterativeChallengeCompletion → MilestoneTracking,
    Redstonea :: WorkflowAutomation → ComplexLogicGates
  ]
}

```

```

//
=====
====
// LAYER 7: COLLABORATION & DISTRIBUTION
//
=====
====

LAYER Collaboration {
  WRAP Argonauta [
    Relata :: DependencyInteractionGraph → PartnerNetworkMapping,
    Erosa :: GraphDatabaseInteractions → SupplierRelationshipAnalytics,
    Aegis-Argonauta :: CollectiveDefense → CooperativeRiskMitigation
  ]

  CHAIN [
    Echo :: SystemWideEvents → MassNotificationBroadcast,
    Summona :: HelperAIDeployment → MicroserviceSpawning,
    Argonauta :: ParallelComputation → DistributedProcessing
  ]

  WRAP Gemini [
    Chimera-Argonauta-Hydra-Phoenix :: CollaborativeSelfHealingNetwork →
PartnerEcosystemResilience,
    Confidara :: AdaptiveFeatureEnhancement → ConditionalPartnerBoosts,
    Karmalis :: ReputationSystem → VendorPerformanceTracking
  ]
}

//
=====
====
// LAYER 8: RESILIENCE & RECOVERY
//
=====
====

LAYER Resilience {
  WRAP Phoenix [
    Vitalis :: AutoRecoveryLoops → CriticalSystemRedundancy,
    Regena :: ProbabilisticRedundancy → NonDeterministicFaultMitigation,
    Hydra :: MultiNodeCluster → RedundantWarehouseNetwork
  ]
}

```

```

NEST {
  Hydra-Ultra [
    CHAIN [
      Vitalis-Maxima :: DynamicBufferExpansion → SurgeCapacityScaling,
      Heartha :: SessionPersistenceRestore → StateRecovery,
      Pandoria :: GracefulFailureRecovery → MinimalViableContinuity
    ]
  ]
}

WRAP Pegasus-Phoenix-Hydra-Aurora [
  Samsara :: SelfHealingMicroservices → ContainerRestarts,
  Phoenix-Cerberus :: SelfRepairingSecurityDefense → AdaptiveSecurityRecovery,
  Pegasus-Hydra :: RapidSelfHealingDeploy → EmergentMobility
]
}

//
=====
====
// LAYER 9: TRANSFORMATION & ADAPTATION
//
=====
=====

LAYER Adaptation {
  WRAP Dragon [
    Shiftara :: ContextBasedFeatureChanges → ModeSwitching,
    Morphis :: DynamicallyChangingProcessStates → AdaptiveProcessMode,
    Adaptis :: AutoLearningFeatureSets → EvolutionaryCapabilities
  ]

  CHAIN [
    Circena :: TypeCastingTransformation → DataPipelineConversion,
    Metalearnara :: AIAgentsSelfImprove → ContinuousLearning,
    Evolvia :: AutomatedVersioning → SystemUpgradeManagement
  ]

  WRAP Chimera [
    Chimeris :: CrossDomainIntegration → HybridSystemFramework,
    Modula :: RapidIntegrationNewBehaviors → ModularScaling,
    Neurolink :: AIHumanCollaboration → HybridDecisionSupport
  ]
}

```

```
//
=====
====
// LAYER 10: TRANSPORTATION & FLEET MANAGEMENT
//
=====
=====
```

```
LAYER Transportation {
  WRAP Pegasus [
    Teleportis :: ContainerizedStateMigration → VehicleStateTransfer,
    Portalus :: TaskMigrationMemoryRemap → FleetReallocation,
    Pegasus-Max :: ContainerOrchestration → UltraRapidFleetDeploy
  ]

  CHAIN [
    Kinetis :: ServerFailureSimulation → FleetStressTest,
    Titanis :: HandlingPeakWorkload → RushHourCapacity,
    Overdrivea :: IntensiveComputationalBursts → HighVelocityDispatch
  ]
}
```

```
  WRAP Roc [
    Ferrana :: PaymentHandoffAPI → TransactionGateway,
    Netheris :: DataMigrationArchival → HistoricalRouteData,
    Roc-Max :: LoadBalancerCoverage → AreaControlDistribution
  ]
}
```

```
//
=====
====
// LAYER 11: WAREHOUSE OPERATIONS
//
=====
=====
```

```
LAYER Warehouse {
  WRAP Golem [
    Atlas :: BackendArchitecture → WarehouseFoundation,
    Fortis :: OnDemandHighPerformance → PeakSeasonMode,
    Golem-Max :: HighAvailabilityStorage → EnduranceNode
  ]
}
```

```

CHAIN [
  Keyfina :: AllInOneUtilityKit → FlexibleWarehouseTools,
  Singularis :: TailoredAIFunctions → SpecializedPickingLogic,
  Infusa :: OnTheFlyFeatureInjection → RuntimeCapabilityEnhancement
]

WRAP Minotaur [
  Aggrega :: ClusteredComputeSystem → ConsolidatedWarehouseOps,
  Furiosa :: RenderingIntensiveMode → HighThroughputSorting,
  Energex :: BatchProcessingHeavyComputations → BulkOrderProcessing
]
}

//
=====
====
// LAYER 12: PREDICTIVE INTELLIGENCE
//
=====
====

LAYER Intelligence {
  WRAP Minerva [
    Athena :: DecisionSupportAI → StrategicPlanning,
    Oedipha :: DecisionPathForecasting → CausalPrediction,
    Minerva-Orion-Thor-Aurora :: HighPowerEmergentTargeting → SmartPowerInsight
  ]

  CHAIN [
    Pandora :: PredictiveErrorContainment → RiskManagement,
    Sophira :: NavigatingComplexity → MultivariateOptimization,
    Arcanum :: ModelingAICoreBehaviors → BehaviorArchetypes
  ]

  WRAP Ophiuchus [
    Pyros :: AIAssistedKnowledgeDistribution → BestPracticeSharing,
    Musara :: ArtisticTextualAIGeneration → ReportGeneration,
    Pandoria-Curio :: AutoDiscoveryMapping → UnchartedRouteExploration
  ]
}

//
=====
=====

```


// LAYER 13: COMPLIANCE & GOVERNANCE

//

=====

=====

LAYER Governance {

 WRAP Heraia [

 Zephyrus :: MasterSystemControl → RootAccessHierarchy,

 Dharmara :: AIBehaviorConstraint → PurposeAlignment,

 Pyroxis :: ComplianceAudit → EnforcementCycle

]

 CHAIN [

 Chronom :: HistoricalStateRestoration → AuditTrailRecovery,

 Atmara :: MultiAgentSelfReflective → DistributedGovernance,

 Nemesia :: BiasCorrectionSystem → FairnessAlgorithm

]

 WRAP Scorpio [

 Icarion :: OverloadPrevention → ThresholdGuard,

 Antigona :: ControlledOverride → ExceptionHandler,

 Sphinxax :: ChallengeResponseAuth → SecurityVerificationLayer

]

}

//

=====

=====

// LAYER 14: CUSTOMER INTERFACE

//

=====

=====

LAYER CustomerInterface {

 WRAP Leo [

 Divinus :: DeveloperUtilitySuite → CustomerPortalTools,

 Sonora :: VoiceBasedSystemManipulation → AudioOrderPlacement,

 Dreamra :: MultiLevelSandboxing → CustomerEnvironmentIsolation

]

 CHAIN [

 Modulor :: TailoredAIBehaviorPerTask → PersonalizedExperience,

 Sirenia :: DistractionPrevention → FocusedCheckout,

 Gryphon-Sirenia :: AdaptiveAttention → DynamicFocusFilter

```

]

WRAP Pisces [
    Taora :: HolisticResourceManagement → UniversalFlow,
    Atmara :: UnifiedConsciousness → SeamlessOmnichannel,
    Unicorn-Pegasus :: ZeroDefectRapidDeploy → AgilePrecisionDelivery
]
}

//
=====
====
// LAYER 15: OPTIMIZATION & CONTINUOUS IMPROVEMENT
//
=====
=====

LAYER Optimization {
    WRAP Cancer [
        Solva :: SolvingComplexLogicBottlenecks → InstantComputation,
        Fractala :: InfiniteDetailSimulations → RecursiveDepthModeling,
        Voidara :: UltraEfficientEdgeDeviceCode → MinimalistReduction
    ]

    CHAIN [
        Eternara :: LongTermAIStrategyOptimization → ReinforcementLearningLoops,
        Koantra :: CreativeProblemSolvingAI → NonlinearReasoning,
        Daedalea :: CreativeArchitectureBuilder → InnovationNode
    ]

    WRAP Unicorn [
        Unicorn-Max :: QATestAutomation → PrecisionNode,
        Unicorn-Aurora-Selene-Poseida :: PredictiveOptimizedStreaming → EmergentPrecision,
        Healix :: AutomaticBugCorrectionPatching → SelfHealing
    ]
}

//
=====
====
// LAYER 16: CRISIS MANAGEMENT
//
=====
=====

```

```

LAYER Crisis {
  WRAP Thunderbird [
    Ultima :: AlertingEmergencyComputation → HighImpactActivation,
    Impacta :: CriticalSystemPivotPoints → GameChangingAction,
    Nemesis-Thunder :: RiskAwareRapidResponse :: SafetySurge
  ]

  NEST {
    Valkyrie-Leviathan [
      CHAIN [
        Janus-Valkyrie-Pandora-Hadeon :: ContextAwareStealthMitigation → AdaptiveCrisisFlow,
        Trojanis :: MalwareDetectionSystems → ThreatContainment,
        Aresia :: LoadFailureSimulation → StressTest
      ]
    ]
  }

  WRAP Kraken [
    Leviathan-Ultra :: GlobalOrchestrationHub → CentralCommand,
    Leviathan-Roc :: DistributedImpact → GlobalCoverage,
    Thor-Leviathan-Hydra-Poseida :: HighImpactAdaptiveOperations → PowerResilienceFlow
  ]
}

//
=====
====
// EMERGE: DIMENSIONAL ORCHESTRATION
//
=====
====

EMERGE DimensionalOrchestra {

  FUSE [
    Minerva-Thor-Vulcan-Pyros :: SmartEnergyCreativeOrchestration → StrategyPowerSurge,
    Athena-Apollo-Daedalea-Hephaestus :: EmergentProblemSolving → CreativeInsight,
    Chimera-Phoenix-Sphinx-Unicorn :: MultiLayeredEmergentLogic → EmergentMetaLogic
  ]

  WRAP Aeona-Einfosa-Nirvara-Triad [
    Byzantium :: MultiAgentCoordination → ConsensusProtocol,
    Entangla :: RealTimeDistributedConsistency → EntangledSync,

```

```

    Nirvara :: ImmutableSystemState → FinalState
]

CHAIN [
    Pegasus-Aurora-Helios-Fractala :: RecursivelyScalableComputation → RecursiveSurge,
    Minerva-Apollo-Poseida-Entangla :: CorrelatedEmergentStreaming → CorrelatedFlow,
    Aurora-Poseida-Hadeon-Sophira :: AIGuidedPredictiveIntelligence → HiddenInsightSurge
]
}

//
=====
====
// FINALIZE: SYSTEM INTEGRATION
//
=====
=====

FINALIZE LogisticsSupplyChainOrchestra {

    BOOTSTRAP [
        Foundation → Visibility → Routing,
        ResourceManagement → Security → Automation,
        Collaboration → Resilience → Adaptation,
        Transportation → Warehouse → Intelligence,
        Governance → CustomerInterface → Optimization,
        Crisis → DimensionalOrchestra
    ]

    ACTIVATE_GLOBAL_SYNC [
        Entangla :: InstantCorrelation,
        Byzantium :: ConsensusCoordination,
        Taora :: UniversalBalance
    ]

    ENABLE_SELF_HEALING [
        Samsara :: CyclicRebirth,
        Vitalis :: AutoRecovery,
        Hydra-Ultra :: AdaptiveRedundancy
    ]

    LOCK_STATE [
        Nirvara :: AbsoluteStability,
        Preserva :: StateCheckpoints,

```

```

    Chronom :: TemporalSnapshots
  ]
}

//
=====
====
// SYSTEM ACTIVE
//
=====
====
...

```

And then asked for it to be in a react demo via claude AI

```

import React, { useState, useEffect } from 'react';
import { Activity, Zap, Shield, TrendingUp, Package, Truck, Database, Eye, Network, Brain,
AlertTriangle, CheckCircle } from 'lucide-react';

const LogisticsOrchestra = () => {
  const [activeLayer, setActiveLayer] = useState('foundation');
  const [systemStatus, setSystemStatus] = useState({});
  const [metrics, setMetrics] = useState({
    throughput: 0,
    latency: 0,
    resilience: 0,
    efficiency: 0
  });
  const [events, setEvents] = useState([]);
  const [activeCloths, setActiveCloths] = useState(new Set());

  const layers = [
    { id: 'foundation', name: 'Foundation', icon: Database, cloth: 'Atlas', color: 'bg-slate-600' },
    { id: 'visibility', name: 'Visibility', icon: Eye, cloth: 'Aurora', color: 'bg-blue-600' },
    { id: 'routing', name: 'Routing', icon: Network, cloth: 'Aquarius', color: 'bg-cyan-600' },
    { id: 'resources', name: 'Resources', icon: TrendingUp, cloth: 'Libra', color: 'bg-green-600' },
    { id: 'security', name: 'Security', icon: Shield, cloth: 'Cerberus', color: 'bg-red-600' },
    { id: 'automation', name: 'Automation', icon: Zap, cloth: 'Vulcan', color: 'bg-orange-600' },
    { id: 'collaboration', name: 'Collaboration', icon: Network, cloth: 'Argonauta', color:
'bg-purple-600' },
    { id: 'resilience', name: 'Resilience', icon: Shield, cloth: 'Phoenix', color: 'bg-rose-600' },
    { id: 'transportation', name: 'Transportation', icon: Truck, cloth: 'Pegasus', color:
'bg-indigo-600' },
    { id: 'warehouse', name: 'Warehouse', icon: Package, cloth: 'Golem', color: 'bg-amber-600' },
    { id: 'intelligence', name: 'Intelligence', icon: Brain, cloth: 'Minerva', color: 'bg-violet-600' },

```

```

    { id: 'crisis', name: 'Crisis Mgmt', icon: AlertTriangle, cloth: 'Thunderbird', color: 'bg-yellow-600'
  }
];

```

```

const spellsByLayer = {
  foundation: ['Hestara', 'Preserva', 'Demetra', 'Fluxa'],
  visibility: ['Clarivis', 'Apollara', 'Insighta', 'Oraclia'],
  routing: ['Poseida', 'Hermesia', 'Hecatia', 'Labyrintha'],
  resources: ['Wuven', 'Equilibria', 'Decisus', 'Gaiana'],
  security: ['Inferna', 'Absorbus', 'Countera', 'Fortifera'],
  automation: ['Magica', 'Crona', 'Herculia', 'Sisyphea'],
  collaboration: ['Relata', 'Echo', 'Argonauta', 'Karmalis'],
  resilience: ['Vitalis', 'Regena', 'Samsara', 'Pandoria'],
  transportation: ['Teleportis', 'Portalus', 'Kinetis', 'Titanis'],
  warehouse: ['Fortis', 'Keyfina', 'Aggrega', 'Energex'],
  intelligence: ['Athena', 'Oedipha', 'Pyros', 'Arcanum'],
  crisis: ['Ultima', 'Impacta', 'Trojanis', 'Aresia']
};

```

```

useEffect(() => {
  const interval = setInterval(() => {
    setMetrics(prev => ({
      throughput: Math.min(100, prev.throughput + Math.random() * 5),
      latency: Math.max(0, prev.latency + (Math.random() - 0.5) * 10),
      resilience: Math.min(100, prev.resilience + Math.random() * 3),
      efficiency: Math.min(100, prev.efficiency + Math.random() * 4)
    }));

    if (Math.random() > 0.7) {
      const layer = layers[Math.floor(Math.random() * layers.length)];
      const spells = spellsByLayer[layer.id];
      const spell = spells[Math.floor(Math.random() * spells.length)];

      setEvents(prev => [{
        id: Date.now(),
        layer: layer.name,
        spell: spell,
        cloth: layer.cloth,
        time: new Date().toLocaleTimeString()
      }, ...prev.slice(0, 4)]);
    }
  }, 1500);

  return () => clearInterval(interval);
}

```

```
}, []);
```

```
useEffect(() => {  
  const statusInterval = setInterval(() => {  
    setSystemStatus(prev => {  
      const newStatus = { ...prev };  
      layers.forEach(layer => {  
        newStatus[layer.id] = Math.random() > 0.9 ? 'warning' : 'active';  
      });  
      return newStatus;  
    });  
  }, 3000);
```

```
  return () => clearInterval(statusInterval);  
}, []);
```

```
const toggleCloth = (clothName) => {  
  setActiveCloths(prev => {  
    const newSet = new Set(prev);  
    if (newSet.has(clothName)) {  
      newSet.delete(clothName);  
    } else {  
      newSet.add(clothName);  
    }  
    return newSet;  
  });  
};
```

```
const fusionCloths = [  
  'Pegasus-Hydra',  
  'Phoenix-Cerberus',  
  'Minerva-Cerulean',  
  'Unicorn-Pegasus',  
  'Chimera-Hydra'  
];
```

```
const dimensionalCloths = [  
  'Pegasus-Phoenix-Hydra-Aurora',  
  'Minerva-Thor-Vulcan-Pyros',  
  'Athena-Apollo-Daedalea-Hephaestus'  
];
```

```
return (
```

```
<div className="min-h-screen bg-gradient-to-br from-slate-900 via-purple-900 to-slate-900
text-white p-6">
```

```
<div className="max-w-7xl mx-auto">
```

```
<div className="mb-8">
```

```
<h1 className="text-4xl font-bold mb-2 bg-gradient-to-r from-cyan-400 to-purple-400
bg-clip-text text-transparent">
```

```
Logistics Supply Chain Orchestra
```

```
</h1>
```

```
<p className="text-slate-400">ORIGIN :: Codex-Compliant Distributed System</p>
```

```
</div>
```

```
<div className="grid grid-cols-4 gap-4 mb-8">
```

```
<div className="bg-slate-800/50 backdrop-blur rounded-lg p-4 border
border-cyan-500/30">
```

```
<div className="text-sm text-slate-400 mb-1">Throughput</div>
```

```
<div className="text-2xl font-bold
text-cyan-400">{metrics.throughput.toFixed(1)}%</div>
```

```
<div className="w-full bg-slate-700 rounded-full h-2 mt-2">
```

```
<div className="bg-cyan-500 h-2 rounded-full transition-all" style={{width:
`${metrics.throughput}%`}}></div>
```

```
</div>
```

```
</div>
```

```
<div className="bg-slate-800/50 backdrop-blur rounded-lg p-4 border
border-green-500/30">
```

```
<div className="text-sm text-slate-400 mb-1">Efficiency</div>
```

```
<div className="text-2xl font-bold
text-green-400">{metrics.efficiency.toFixed(1)}%</div>
```

```
<div className="w-full bg-slate-700 rounded-full h-2 mt-2">
```

```
<div className="bg-green-500 h-2 rounded-full transition-all" style={{width:
`${metrics.efficiency}%`}}></div>
```

```
</div>
```

```
</div>
```

```
<div className="bg-slate-800/50 backdrop-blur rounded-lg p-4 border
border-rose-500/30">
```

```
<div className="text-sm text-slate-400 mb-1">Resilience</div>
```

```
<div className="text-2xl font-bold
text-rose-400">{metrics.resilience.toFixed(1)}%</div>
```

```
<div className="w-full bg-slate-700 rounded-full h-2 mt-2">
```

```
<div className="bg-rose-500 h-2 rounded-full transition-all" style={{width:
`${metrics.resilience}%`}}></div>
```

```
</div>
```



```

</div>

<div className="bg-slate-800/50 backdrop-blur rounded-lg p-4 border
border-purple-500/30">
  <div className="text-sm text-slate-400 mb-1">Latency</div>
  <div className="text-2xl font-bold
text-purple-400">{metrics.latency.toFixed(0)}ms</div>
  <div className="w-full bg-slate-700 rounded-full h-2 mt-2">
    <div className="bg-purple-500 h-2 rounded-full transition-all" style={{width:
`${Math.min(100, metrics.latency)}%`}}></div>
  </div>
</div>
</div>

<div className="grid grid-cols-3 gap-6">

  <div className="col-span-2">
    <h2 className="text-xl font-bold mb-4 flex items-center gap-2">
      <Activity className="w-5 h-5" />
      System Layers
    </h2>

    <div className="grid grid-cols-3 gap-3">
      {layers.map(layer => {
        const Icon = layer.icon;
        const status = systemStatus[layer.id] || 'active';

        return (
          <button
            key={layer.id}
            onClick={() => setActiveLayer(layer.id)}
            className={` ${layer.color} ${activeLayer === layer.id ? 'ring-4 ring-white/50
scale-105' : ''}
            ${status === 'warning' ? 'animate-pulse' : ''}
            rounded-lg p-4 transition-all hover:scale-105 cursor-pointer`}
          >
            <div className="flex items-center justify-between mb-2">
              <Icon className="w-6 h-6" />
              {status === 'active' ? (
                <CheckCircle className="w-4 h-4 text-green-300" />
              ) : (
                <AlertTriangle className="w-4 h-4 text-yellow-300" />
              )}
            </div>
          </div>
        )
      })}
    </div>
  </div>

```

```

        <div className="font-bold text-sm">{layer.name}</div>
        <div className="text-xs opacity-75 mt-1">WRAP {layer.cloth}</div>
    </button>
  );
  }}}
</div>

{activeLayer && (
  <div className="mt-6 bg-slate-800/50 backdrop-blur rounded-lg p-6 border
border-slate-700">
    <h3 className="text-lg font-bold mb-4">
      Active Spells: {layers.find(l => l.id === activeLayer)?.name}
    </h3>
    <div className="grid grid-cols-2 gap-3">
      {spellsByLayer[activeLayer]?.map(spell => (
        <div key={spell} className="bg-slate-700/50 rounded p-3 border border-slate-600
hover:border-cyan-500/50 transition-all">
          <div className="font-mono text-cyan-400 text-sm">{spell}</div>
          <div className="text-xs text-slate-400 mt-1">
            {spell.includes('vis') ? 'Monitoring' :
            spell.includes('is') ? 'Orchestration' :
            spell.includes('ra') ? 'Management' : 'System'}
          </div>
        </div>
      ))}
    </div>
  )}
</div>
</div>
)}
</div>

<div className="space-y-6">

  <div>
    <h2 className="text-xl font-bold mb-4">Fusion Cloths</h2>
    <div className="space-y-2">
      {fusionCloths.map(cloth => (
        <button
          key={cloth}
          onClick={() => toggleCloth(cloth)}
          className={`w-full text-left p-3 rounded-lg border transition-all ${
            activeCloths.has(cloth)
              ? 'bg-purple-600 border-purple-400'
              : 'bg-slate-800/50 border-slate-700 hover:border-purple-500/50'
            }`}
        >

```

```

    >
    <div className="text-sm font-mono">{cloth}</div>
    <div className="text-xs text-slate-400 mt-1">Emergent Pattern</div>
  </button>
  )})
</div>
</div>

<div>
  <h2 className="text-xl font-bold mb-4">Dimensional Cloths</h2>
  <div className="space-y-2">
    {dimensionalCloths.map(cloth => (
      <button
        key={cloth}
        onClick={() => toggleCloth(cloth)}
        className={`w-full text-left p-3 rounded-lg border transition-all ${
          activeCloths.has(cloth)
            ? 'bg-gradient-to-r from-purple-600 to-pink-600 border-pink-400'
            : 'bg-slate-800/50 border-slate-700 hover:border-pink-500/50'
        }`}
      >
        <div className="text-xs font-mono leading-tight">{cloth}</div>
        <div className="text-xs text-slate-400 mt-1">4D Emergence</div>
      </button>
    )})
  </div>
</div>

<div>
  <h2 className="text-xl font-bold mb-4">Event Log</h2>
  <div className="space-y-2 bg-slate-800/50 backdrop-blur rounded-lg p-4 border
border-slate-700 max-h-80 overflow-y-auto">
    {events.map(event => (
      <div key={event.id} className="text-xs border-l-2 border-cyan-500 pl-3 py-2
bg-slate-700/30 rounded">
        <div className="text-cyan-400 font-mono">{event.spell}</div>
        <div className="text-slate-400">
          {event.layer} • {event.cloth} • {event.time}
        </div>
      </div>
    )})
    {events.length === 0 && (
      <div className="text-slate-500 text-center py-8">Awaiting system events...</div>
    )}
  </div>
</div>

```

```

        </div>
      </div>
    </div>
  </div>

  <div className="mt-8 bg-gradient-to-r from-slate-800/50 to-purple-800/50 backdrop-blur
rounded-lg p-4 border border-purple-500/30">
    <div className="flex items-center justify-between">
      <div>
        <div className="text-sm text-slate-400">System State</div>
        <div className="font-mono text-lg text-purple-300">FINALIZED :: Active
Orchestration</div>
      </div>
      <div className="text-right">
        <div className="text-sm text-slate-400">Active Cloths</div>
        <div className="font-bold text-2xl text-purple-300">{activeCloths.size}</div>
      </div>
    </div>
  </div>
</div>
);
};

export default LogisticsOrchestra;

```