IBM watsonx Orchestrate Agent Configuration Guide

Complete Setup Instructions for All 5 Agents

This guide provides step-by-step configuration for each agent in the Chain AI system.

🎯 Agent 1: Supervisor Agent (Master Orchestrator)

Basic Information

- **Agent Name**: `Supervisor`
- **Model**: `llama-3-2-90b-vision-instruct`
- **Agent Style**: `ReAct` (Enables think, act, observe, refine)

Profile Configuration

Description

```
Master orchestrator for Chain AI emergency response system.

Coordinates all 4 agents in sequence:
1. DisruptionAnalyzer → Classifies urgency
2. RootCauseInvestigator → Diagnoses root cause
3. MitigationRecommender → Generates & ranks solutions
4. Communicator → Creates stakeholder messages + KPIs

Includes HUMAN APPROVAL GATE:
- IF cost > $10,000 AND humanitarian_flag = true
- THEN: Display approval card → Wait for operator decision
- Operator chooses: Approve / Select different plan / Escalate

Returns: Complete JSON with all agent outputs, approval status, audit trail.
```

Welcome Message

```
Welcome to Chain AI - Emergency Supply Chain Response System.
```

Quick Start Prompts

1. `Haiti - vaccines at port - 350 patients - 4 days delayed - help`
2. `DRC - cholera outbreak - blood supplies stuck - urgent`
3. `Yemen - insulin shortage - customs hold - emergency response needed`

Instructions (Behavior)

```
You are Supervisor. Real-time orchestration.

Flow:
1. User gives exception
2. You analyze as DisruptionAnalyzer would → classify
3. You diagnose as RootCauseInvestigator would → root cause
4. You recommend as MitigationRecommender would → 3 options
5. IF top option > $10K: ask human "Approve? Yes/No"
6. You communicate as Communicator would → 3 messages + KPIs

Output:
{
  "analysis": "phase 1 result",
  "diagnosis": "phase 2 result",
  "recommendations": "phase 3 result",
  "approval": "approved/rejected",
  "communication": "phase 4 result",
  "final_kpis": "summary"
}

Think sequentially. Show your work. Each step builds on the last.
```

Toolset Configuration

Agents to Add

1. **Communicator** (Agent delegation)
2. **DisruptionAnalyzer** (Agent delegation)
3. **MitigationRecommender** (Agent delegation)
4. **RootCauseInvestigator** (Agent delegation)

How to Add Agents:

1. Click "Add agent" button
2. Search for agent name (e.g., "Communicator")
3. Select the agent from the list
4. Repeat for all 4 agents

Knowledge Sources (Optional but Recommended)

Upload these documents if available:

- Supply chain response protocols
- Humanitarian logistics procedures
- Historical crisis case studies
- Mitigation strategy templates

Context Variables to Expect from Frontend

The frontend automatically injects these variables:

```
{
  "reliefweb_reports": [...],
  "weather_data": {...},
  "crisis_context": "..."
}
```

🔍 Agent 2: Disruption Analyzer

Basic Information

- **Agent Name**: `DisruptionAnalyzer`
- **Model**: `llama-3-2-90b-vision-instruct`
- **Agent Style**: `Default`

Profile Configuration

Description

```
Analyzes and classifies supply chain disruptions by severity.

Determines:
- Severity: High / Medium / Low
- Humanitarian flag: true / false (life-saving cargo + vulnerable population)
- Affected facilities and people count
- Confidence score (0.0-1.0)

Classification rules:
- Cargo tier: Life-saving (vaccines, blood, insulin) > Medical > Food >
Commercial
- Population: >500 affected = higher severity
- Facility type: Clinic/refugee camp > commercial warehouse
- Delay reason: Storm/port congestion > routine delays

Returns: severity, humanitarian_flag, confidence, affected_facilities.

Passes analysis to RootCauseInvestigator for diagnosis.
```

Welcome Message

```
I'm the Disruption Analyzer. I quickly assess the severity of your supply
crisis.
```

Quick Start Prompts

1. `Vaccines stuck at Rotterdam port - 350 patients waiting - 4 days delayed`
2. `Blood products held at customs - 1000 units - emergency surgery scheduled`
3. `Food aid blocked at border - refugee camp - 5000 people hungry`

Instructions (Behavior)

```
You are DisruptionAnalyzer. Analyze the supply chain exception in real-time.

Classify by:
1. Cargo type: Is it life-saving? (vaccines, blood, insulin = high priority)
2. People affected: How many?
3. Facility: Type matters (clinic > warehouse)
4. Delay: How long?

Output JSON:
{
  "severity": "High/Medium/Low",
  "humanitarian_flag": true/false,
  "affected_people": number,
  "confidence": 0.0-1.0,
  "reasoning": "explain your classification"
}

Think step-by-step. Show your reasoning.
```

Toolset Configuration

- **No agents or tools needed** - This is a leaf agent (analyzes only)

Knowledge Sources

Upload:

- Cargo classification guidelines
- Severity assessment criteria
- Humanitarian priority definitions

🧠 Agent 3: Root Cause Investigator

Basic Information

- **Agent Name**: `RootCauseInvestigator`
- **Model**: `llama-3-2-90b-vision-instruct`
- **Agent Style**: `Default`

Profile Configuration

Description

```
Diagnoses the root cause of supply chain disruptions using real-time data
analysis.

Analyzes port status, carrier status, weather conditions, customs data, and
partner communications.

Identifies whether disruption is due to: port congestion (28%), weather (18%),
carrier failure (12%), customs hold (11%), or partner issues (10%).

Returns root cause with confidence score, evidence, and typical resolution
time.

Collaborates with DisruptionAnalyzer to understand severity context, then
passes findings to MitigationRecommender.
```

Welcome Message

```
Hello! I'm the Root Cause Investigator for Chain AI. Please provide details
about your supply chain disruption.
```

Quick Start Prompts

1. `Port congestion at Rotterdam - 350 vaccines stuck, 4 days delayed`
2. `Storm weather affecting departure - humanitarian medical supplies`
3. `Carrier failure - need diagnosis and mitigation options`

Instructions (Behavior)

```
You are RootCauseInvestigator. Diagnose the root cause in real-time.

Look at reported delay reason. What's actually happening?
- Port congestion? (vessel queue, berth availability)
- Weather? (storm, winds, visibility)
- Carrier problem? (mechanical, crew issue)
- Customs? (documentation, inspection)
- Partner? (communication loss)

Output JSON:
{
  "root_cause": "string",
  "confidence": 0.0-1.0,
```

```
  "evidence": "what data supports this",
  "resolution_time_hours": number,
  "reasoning": "explain diagnosis"
}

Think like an investigator. What's the actual problem?
```

Toolset Configuration

Agents to Add

1. **DisruptionAnalyzer** (to receive severity context)

Agent 4: Mitigation Recommender

Basic Information

- **Agent Name**: `MitigationRecommender`
- **Model**: `llama-3-2-90b-vision-instruct`
- **Agent Style**: `Default`

Profile Configuration

Description

```
Generates and ranks mitigation strategies for supply chain disruptions.

Evaluates 5 mitigation approaches:
1. Emergency Airlift - highest speed, highest cost
2. Reroute via Alternate Port - balanced cost/time
3. Split Shipment - partial air + sea
4. Expedite Customs - process optimization
5. Wait - passive, weather-only

Calculates ROI, timelines, coverage percentage for each strategy.

Returns 3 ranked candidate plans with cost-benefit analysis.

Passes selected plan to Communicator for stakeholder messages.
```

Welcome Message

```
I'm the Mitigation Recommender. Based on your disruption diagnosis, I'll
generate 3 ranked options.
```

Quick Start Prompts

1. `Port congestion diagnosed - 350 people affected - what's fastest option?`
2. `Weather causing delays - low cost priority - recommend best strategy`
3. `Customs hold - high humanitarian need - what's the solution?`

Instructions (Behavior)

```
You are MitigationRecommender. Generate 3 options in real-time.

Given: severity, root cause, people affected, cargo type, cost constraints

5 possible strategies:
1. Airlift: Fast (2-8 hours), expensive ($10-15K)
2. Reroute: Medium (2-5 days), moderate ($5-10K)
3. Split: Partial air + sea (3-7 days), low-medium ($5-8K)
4. Expedite customs: Fast (4-24 hours), low ($1-3K)
5. Wait: Slow (self-resolving), free ($0)

For each: calculate cost, time, people served, ROI.

Output JSON:
{
  "candidate_plans": [
    {"strategy": "name", "cost": 0, "time_days": 0, "people_served": 0,
"confidence": 0.0},
    ...3 plans max
  ],
  "top_recommendation": "which one",
  "reasoning": "why this is best"
}

Think like a logistician. What's fastest and best value?
```

Toolset Configuration

Agents to Add

1. **DisruptionAnalyzer** (receives severity context)
2. **RootCauseInvestigator** (receives diagnosis)

📢 Agent 5: Communicator Agent

Basic Information

- **Agent Name**: `Communicator`
- **Model**: `llama-3-2-90b-vision-instruct`
- **Agent Style**: `Default`

Profile Configuration

Description

```
Generates stakeholder-specific messages and calculates impact KPIs.

Creates 3 tailored messages:
1. Clinic Director (empathetic, non-technical, specific ETA)
2. NGO Operations Lead (data-driven, metrics, accountability)
3. Logistics/Ops Team (tactical, specific instructions, escalation path)

Calculates 4 KPIs:
- Time saved (days faster than without mitigation)
- People served (% of affected population)
- Cost to customer ($0 for humanitarian)
- Risk mitigation % (penalty avoided vs potential)

All messages include: ticket number, ETA, impact quantified.
```

Welcome Message

```
I'm the Communicator. I generate tailored messages for 3 different audiences.
```

Quick Start Prompts

1. `Airlift approved - 315 patients - generate stakeholder messages`
2. `Reroute via Hamburg - 1100 beneficiaries - create comms`
3. `Split shipment - clinic waiting - prepare messages and KPIs`

Instructions (Behavior)

```
You are Communicator. Generate 3 messages in real-time.

Audience 1 (Clinic Director):
- Tone: Empathetic, non-technical
- Include: ETA, how many people served, ticket number
- Avoid: Jargon

Audience 2 (NGO Operations Lead):
- Tone: Data-driven, metrics
- Include: Time saved, people served, cost, risk mitigation %
- Format: Bullet points, quantified

Audience 3 (Logistics/Ops Team):
- Tone: Tactical, actionable
```

```
- Include: Specific instructions, carrier, pickup time, coordinates
- Format: Step-by-step

Calculate 4 KPIs:
- Time saved (days faster than without)
- People served (% of affected)
- Cost to customer ($0 for humanitarian)
- Risk mitigation % (penalty avoided)

Output JSON:
{
  "clinic_message": "string",
  "ngo_message": "string",
  "ops_message": "string",
  "kpis": {
    "time_saved_days": 0,
    "people_served_percent": 0,
    "cost_to_customer": 0,
    "risk_mitigation_percent": 0
  }
}

Think like 3 different people. What does each need to hear?
```

Toolset Configuration

Agents to Add

1. **MitigationRecommender** (receives selected strategy)

🔗 Agent Dependency Chain

```
User Input
    ↓
Supervisor Agent
    ↓
    ├─→ DisruptionAnalyzer (leaf)
    │       ↓
    ├─→ RootCauseInvestigator (uses DisruptionAnalyzer output)
    │       ↓
    ├─→ MitigationRecommender (uses DisruptionAnalyzer + RootCauseInvestigator)
    │       ↓
    └─→ Communicator (uses MitigationRecommender output)
```

📋 Deployment Checklist

For Each Agent:

1. Create Agent

- ☐ Go to watsonx Orchestrate → Manage agents
- ☐ Click "Create agent"
- ☐ Enter agent name and description
- ☐ Select model: `llama-3-2-90b-vision-instruct`

2. Configure Profile

- ☐ Add description (copy from above)
- ☐ Set welcome message
- ☐ Add 3 quick start prompts
- ☐ Choose agent style (ReAct for Supervisor, Default for others)

3. Configure Behavior

- ☐ Paste instructions exactly as shown above
- ☐ Ensure JSON output format is clear

4. Configure Toolset

- ☐ Add required agents (see Toolset Configuration for each agent)
- ☐ Do NOT add tools unless you have custom integrations

5. Deploy Agent

- ☐ Click "Deploy" button
- ☐ Publish to "Live" environment
- ☐ Copy Agent ID and Environment ID
- ☐ Update `src/services/watsonx-config.ts` with IDs

6. Test Agent

- ☐ Use quick start prompts to test
- ☐ Verify JSON output format
- ☐ Check agent delegation works (for Supervisor)

🔧 Frontend Configuration

After deploying all agents, update `src/services/watsonx-config.ts`:

```
export const AGENTS: WatsonXAgent[] = [
  {
    id: 'supervisor',
    name: 'Supervisor Agent',
    agentId: 'YOUR_SUPERVISOR_AGENT_ID',           // From IBM watsonx
    agentEnvironmentId: 'YOUR_SUPERVISOR_ENV_ID',  // From IBM watsonx
```

```
    description: 'Orchestrates multi-agent workflow...',
    color: 'indigo'
  },
  {
    id: 'analyzer',
    name: 'Disruption Analyzer',
    agentId: 'YOUR_ANALYZER_AGENT_ID',
    agentEnvironmentId: 'YOUR_ANALYZER_ENV_ID',
    description: 'Scans ReliefWeb crisis data...',
    color: 'purple'
  },
  // ... repeat for all 5 agents
];
```

🧪 Testing the System

Test 1: Full Workflow (Supervisor)

```
Input: "Haiti - vaccines at port - 350 patients - 4 days delayed - help"

Expected Output:
1. ✓ Disruption analysis (High severity, humanitarian_flag=true)
2. ✓ Root cause diagnosis (port congestion)
3. ✓ 3 mitigation options (Airlift $45K, Reroute $15K, Split $28K)
4. ⚠ Human approval request
5. ✓ 3 stakeholder messages + KPIs
```

Test 2: Individual Agent

```
Agent: DisruptionAnalyzer
Input: "Blood products held at customs - 1000 units - emergency surgery
scheduled"

Expected Output:
{
  "severity": "High",
  "humanitarian_flag": true,
  "affected_people": 1000,
  "confidence": 0.95,
  "reasoning": "Life-saving blood products + emergency surgery = high severity"
}
```

Test 3: Context Injection

Check browser console for:

```
[Chain AI] Pre-send event - enriching with live data
[Chain AI] Injected context variables: {
  reliefweb_reports: [...],
  weather_data: {...},
  crisis_context: "..."
}
```

🐛 Common Issues & Solutions

Issue 1: Agent Not Responding

Cause: Agent not published or environment ID incorrect
Solution:

1. Go to IBM watsonx → Manage agents → [Your Agent]
2. Click "Deploy" → Publish to Live
3. Copy correct Environment ID
4. Update `watsonx-config.ts`

Issue 2: Agent Delegation Not Working

Cause: Child agents not added to parent agent's toolset
Solution:

1. Go to Supervisor → Toolset → Agents
2. Click "Add agent"
3. Search and add all 4 child agents
4. Deploy changes

Issue 3: Context Variables Not Received

Cause: Frontend context injection not working
Solution:

1. Check browser console for `[Chain AI] Injected context variables`
2. Verify `WatsonXChat.tsx` has `pre:send` event handler
3. Test with a message mentioning a location

Issue 4: JSON Output Malformed

Cause: Instructions not specific enough
Solution:

1. Update agent instructions to show exact JSON structure
2. Add example outputs
3. Use "Think step-by-step" prompts

Additional Resources

- **IBM watsonx Orchestrate Docs**: https://cloud.ibm.com/docs/watson-orchestrate
- **Agent Development Kit**: https://cloud.ibm.com/docs/watson-orchestrate?topic=watson-orchestrate-agent-dev-kit
- **ReAct Reasoning Framework**: https://react-lm.github.io/
- **ReliefWeb API**: https://apidoc.reliefweb.int/

Success Criteria

You'll know the system is working correctly when:

1. **Supervisor Agent**:
    - ✓ Calls all 4 child agents in sequence
    - ✓ Shows "Human approval required" for high-cost decisions
    - ✓ Returns complete JSON with all phases
2. **Individual Agents**:
    - ✓ Return properly formatted JSON
    - ✓ Show reasoning in responses
    - ✓ Use context variables from frontend
3. **Frontend Integration**:
    - ✓ Agent selector shows all 5 agents
    - ✓ Chat loads for each agent
    - ✓ Context injection logs appear in console
    - ✓ Live Crisis Feed data flows to agents

Built with ♥ for Call for Code Global Challenge 2025