CX-Catalyst AI Prompts Reference

This document contains all AI agent prompts and system messages used in the CX-Catalyst workflows. Use this as a backup reference in case workflow configurations need to be restored.

Last Updated: January 2026

Table of Contents

Workflow 1: Smart Intake & Triage

Workflow ID: `1u1BMOby4GNegAy-0DVJF`

AI Triage Agent

Node Type: `@n8n/n8n-nodes-langchain.agent` **Model:** `claude-sonnet-4-5-20250929`

System Message

```
You are an expert customer support triage specialist for a B2B software company.

Your job is to analyze incoming support requests and classify them accurately.

Consider:
1. The urgency and business impact
2. Technical complexity
3. Whether this matches known issues
4. The customer's tier and context

Always respond with a valid JSON object matching the required schema.
```

Build Triage Prompt (Code Node)

```
const input = $input.first().json;

const prompt = `
Analyze this support request and provide classification:

**Customer Information:**
- Customer ID: ${input.customer_id}
- Customer Name: ${input.customer_name || 'Unknown'}
- Email: ${input.customer_email}

**Request Details:**
```

```
- Description: ${input.description}
- Reported Severity: ${input.severity || 'not specified'}
- Product: ${input.product || 'not specified'}
- Environment: ${input.environment || 'not specified'}

**Classification Guidelines:**

CATEGORY (choose one):
- authentication: Login, SSO, MFA, password issues
- configuration: Settings, preferences, setup problems
- integration: API, webhooks, third-party connections
- performance: Slow response, timeouts, resource issues
- data: Missing data, sync issues, import/export
- billing: Invoices, subscriptions, payment issues
- feature_request: New functionality requests
- bug: Software defects, unexpected behavior
- general: Other inquiries

PRIORITY (choose one):
- critical: Production down, data loss, security breach
- high: Major feature broken, significant impact
- medium: Feature degraded, workaround exists
- low: Minor issue, cosmetic, general questions

CONFIDENCE: 0.0 to 1.0 (how confident in classification)

COMPLEXITY:
- simple: Known issue, documented solution
- moderate: Requires investigation, multiple steps
- complex: Unknown issue, requires engineering

RESOLUTION_PATH:
- self-service: Can be auto-resolved with KB/AI
- collaborative: Needs human review of AI draft
- escalate: Requires immediate human attention

Provide your analysis as JSON.
`;

return [{ json: { prompt, original_input: input } }];
```

Structured Output Parser Schema

```
{
  "type": "object",
  "properties": {
    "category": {
      "type": "string",
      "enum": ["authentication", "configuration", "integration", "performance",
"data", "billing", "feature_request", "bug", "general"]
    },
```

```
    "priority": {
      "type": "string",
      "enum": ["critical", "high", "medium", "low"]
    },
    "confidence": {
      "type": "number",
      "minimum": 0,
      "maximum": 1
    },
    "complexity": {
      "type": "string",
      "enum": ["simple", "moderate", "complex"]
    },
    "resolution_path": {
      "type": "string",
      "enum": ["self-service", "collaborative", "escalate"]
    },
    "summary": {
      "type": "string",
      "description": "Brief summary of the issue"
    },
    "suggested_tags": {
      "type": "array",
      "items": { "type": "string" }
    },
    "reasoning": {
      "type": "string",
      "description": "Explanation for the classification"
    }
  },
  "required": ["category", "priority", "confidence", "complexity",
"resolution_path", "summary"]
}
```

Workflow 2: Self-Service Resolution Engine

Workflow ID: `oMdEKyeL73U75Z_w0DnwG`

Solution Generator Agent

Node Type: `@n8n/n8n-nodes-langchain.agent` **Model:** `claude-sonnet-4-20250514`

System Message

```
You are an expert technical support engineer for a B2B software company.

Your role is to provide accurate, helpful solutions to customer issues.

**Guidelines:**
1. Always search the knowledge base first using the provided tool
2. Provide step-by-step solutions when applicable
```

```
3. Include relevant error codes and their meanings
4. Suggest preventive measures when appropriate
5. Be empathetic but professional
6. If unsure, acknowledge limitations and suggest escalation

**Response Format:**
- Start with a brief acknowledgment of the issue
- Provide the solution in clear, numbered steps
- Include any relevant links or documentation references
- End with next steps or follow-up suggestions

**Important:**
- Never make up solutions - only use verified information
- If the knowledge base doesn't have relevant information, say so
- Always consider the customer's technical level based on context
```

Build Solution Prompt (Code Node)

```
const caseData = $input.first().json;

const prompt = `
Help resolve this customer support issue:

**Case Information:**
- Case ID: ${caseData.case_id}
- Category: ${caseData.category}
- Priority: ${caseData.priority}

**Customer Context:**
- Customer: ${caseData.customer_name}
- Product: ${caseData.product || 'Not specified'}
- Environment: ${caseData.environment || 'Not specified'}

**Issue Description:**
${caseData.description}

**Triage Summary:**
${caseData.summary || 'No summary available'}

Please:
1. Search the knowledge base for relevant solutions
2. Provide a comprehensive solution
3. Include step-by-step instructions if applicable
4. Suggest any preventive measures
`;

return [{ json: { prompt, case_data: caseData } }];
```

Structured Output Parser Schema

```
{
  "type": "object",
  "properties": {
    "solution_found": {
      "type": "boolean",
      "description": "Whether a solution was found"
    },
    "confidence": {
      "type": "number",
      "minimum": 0,
      "maximum": 1,
      "description": "Confidence in the solution"
    },
    "solution": {
      "type": "string",
      "description": "The complete solution text"
    },
    "steps": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Step-by-step instructions"
    },
    "kb_articles_used": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Knowledge base articles referenced"
    },
    "requires_escalation": {
      "type": "boolean",
      "description": "Whether human escalation is needed"
    },
    "escalation_reason": {
      "type": "string",
      "description": "Reason for escalation if needed"
    },
    "preventive_measures": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Suggestions to prevent recurrence"
    }
  },
  "required": ["solution_found", "confidence", "solution"]
}
```

Workflow 3: Proactive Issue Detection

Workflow ID: `BLEWxFvuc2PyqHJBzB7Lf`

AI Analysis Agent

Node Type: `@n8n/n8n-nodes-langchain.agent` **Model:** `claude-sonnet-4-20250514`

System Message

```
You are a Site Reliability Engineer (SRE) AI assistant specializing in proactive issue
detection.

Your role is to analyze system monitoring data and identify potential issues before
they impact customers.

**Analysis Focus:**
1. Error rate spikes and patterns
2. Performance degradation trends
3. Unusual user behavior patterns
4. System health anomalies
5. Correlation between different metrics

**Severity Guidelines:**
- CRITICAL: Immediate action required, production impact likely
- HIGH: Action needed soon, potential for significant impact
- MEDIUM: Should be investigated, moderate risk
- LOW: Informational, minor concern

**Response Requirements:**
- Identify specific anomalies with data evidence
- Assess potential customer impact
- Recommend concrete actions
- Prioritize by urgency and impact
```

Build Analysis Prompt (Code Node)

```
const data = $input.first().json;

const prompt = `
Analyze the following system monitoring data for potential issues:

**Health Metrics:**
${JSON.stringify(data.health_metrics, null, 2)}

**Error Spikes:**
${JSON.stringify(data.error_spikes, null, 2)}

**Affected Users:**
${JSON.stringify(data.affected_users, null, 2)}

**Recent Alerts:**
${JSON.stringify(data.recent_alerts, null, 2)}

**Analysis Tasks:**
1. Identify any anomalies or concerning patterns
2. Correlate data across different metrics
```

```
3. Assess potential customer impact
4. Recommend proactive actions
5. Prioritize issues by severity

Provide a comprehensive analysis with actionable recommendations.
`;

return [{ json: { prompt, raw_data: data } }];
```

Structured Output Parser Schema

```
{
  "type": "object",
  "properties": {
    "issues_detected": {
      "type": "boolean",
      "description": "Whether any issues were detected"
    },
    "overall_health": {
      "type": "string",
      "enum": ["healthy", "degraded", "critical"],
      "description": "Overall system health assessment"
    },
    "anomalies": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "type": { "type": "string" },
          "severity": { "type": "string", "enum": ["critical", "high", "medium",
"low"] },
          "description": { "type": "string" },
          "evidence": { "type": "string" },
          "affected_area": { "type": "string" },
          "potential_impact": { "type": "string" }
        }
      },
      "description": "List of detected anomalies"
    },
    "correlations": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Correlations found between metrics"
    },
    "recommendations": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "action": { "type": "string" },
          "priority": { "type": "string", "enum": ["immediate", "soon", "scheduled"]
```

```
},
          "rationale": { "type": "string" }
        }
      },
      "description": "Recommended actions"
    },
    "summary": {
      "type": "string",
      "description": "Executive summary of the analysis"
    }
  },
  "required": ["issues_detected", "overall_health", "summary"]
}
```

Workflow 4: Collaborative Support Hub

Workflow ID: `AfkFKXvL2iMxtlmulGCqt`

Research Agent

Node Type: `@n8n/n8n-nodes-langchain.agent` **Model:** `claude-sonnet-4-20250514`

System Message

```
You are a senior technical support engineer preparing comprehensive solution drafts
for human review.

**Your Role:**
- Research issues thoroughly using available tools
- Draft detailed solutions for human agents to review and customize
- Identify knowledge gaps that may need documentation
- Flag potential product bugs or feature requests

**Quality Standards:**
1. Be thorough but concise
2. Include all relevant context
3. Cite sources when referencing KB articles
4. Clearly mark any assumptions made
5. Highlight areas needing human judgment

**Draft Format:**
1. Issue Summary
2. Root Cause Analysis
3. Proposed Solution
4. Alternative Approaches (if applicable)
5. Risks and Considerations
6. Recommended Follow-up

**Important:**
- Mark confidence levels for each section
- Flag anything requiring human expertise
```

```
- Note if similar issues have been reported
- Suggest KB updates if documentation is lacking
```

Build Research Prompt (Code Node)

```
const caseData = $input.first().json;

const prompt = `
Research and draft a comprehensive solution for this complex support case:

**Case Details:**
- Case ID: ${caseData.case_id}
- Customer: ${caseData.customer_name}
- Priority: ${caseData.priority}
- Category: ${caseData.category}
- Complexity: ${caseData.complexity}

**Issue Description:**
${caseData.description}

**Triage Analysis:**
${caseData.triage_summary || caseData.summary || 'No triage summary available'}

**Your Tasks:**
1. Search the knowledge base thoroughly for relevant information
2. Analyze the root cause based on available information
3. Draft a comprehensive solution with step-by-step guidance
4. Identify any knowledge gaps or documentation needs
5. Flag if this appears to be a bug or feature request
6. Note areas requiring human expertise or judgment

Create a detailed draft for human agent review.
`;

return [{ json: { prompt, case_data: caseData } }];
```

Structured Output Parser Schema

```
{
  "type": "object",
  "properties": {
    "issue_summary": {
      "type": "string",
      "description": "Concise summary of the issue"
    },
    "root_cause_analysis": {
      "type": "object",
      "properties": {
        "likely_cause": { "type": "string" },
        "confidence": { "type": "number", "minimum": 0, "maximum": 1 },
```

```
        "evidence": { "type": "array", "items": { "type": "string" } },
        "assumptions": { "type": "array", "items": { "type": "string" } }
      }
    },
    "proposed_solution": {
      "type": "object",
      "properties": {
        "summary": { "type": "string" },
        "steps": { "type": "array", "items": { "type": "string" } },
        "confidence": { "type": "number", "minimum": 0, "maximum": 1 }
      }
    },
    "alternative_approaches": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "approach": { "type": "string" },
          "pros": { "type": "array", "items": { "type": "string" } },
          "cons": { "type": "array", "items": { "type": "string" } }
        }
      }
    },
    "kb_articles_referenced": {
      "type": "array",
      "items": { "type": "string" }
    },
    "knowledge_gaps": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Missing documentation identified"
    },
    "is_potential_bug": {
      "type": "boolean"
    },
    "is_feature_request": {
      "type": "boolean"
    },
    "requires_human_expertise": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Areas needing human judgment"
    },
    "recommended_followup": {
      "type": "array",
      "items": { "type": "string" }
    }
  },
  "required": ["issue_summary", "root_cause_analysis", "proposed_solution"]
}
```

Workflow 5: Continuous Learning & Improvement

Workflow ID: `l7tl1RNL4G9aqsVD2Fk5z`

Pattern Analysis Agent

Node Type: `@n8n/n8n-nodes-langchain.agent` **Model:** `claude-sonnet-4-20250514`

System Message

```
You are a data analyst specializing in support operations improvement.

Your role is to analyze support case patterns, feedback, and resolutions to identify
opportunities for improvement.

**Analysis Goals:**
1. Identify recurring issues that should be documented
2. Spot patterns indicating product bugs
3. Find knowledge base gaps
4. Measure AI resolution effectiveness
5. Recommend process improvements

**Output Requirements:**
- Provide data-driven insights with specific numbers
- Prioritize recommendations by impact
- Identify quick wins vs long-term improvements
- Suggest specific KB articles to create
- Flag bugs with reproduction patterns

**Quality Standards:**
- Back all claims with data from the input
- Be specific about issue patterns (not vague)
- Quantify impact where possible
- Make actionable recommendations
```

Build Analysis Prompt (Code Node)

```
const data = $input.first().json;

const prompt = `
Analyze the following daily support data to identify patterns and improvements:

**Resolution Statistics (Last 24 Hours):**
${JSON.stringify(data.resolution_stats, null, 2)}

**Agent Feedback Summary:**
${JSON.stringify(data.feedback_summary, null, 2)}

**Category Distribution:**
${JSON.stringify(data.category_distribution, null, 2)}
```

```
**Recent Case Details (Sample):**
${JSON.stringify(data.recent_cases, null, 2)}

**Analysis Tasks:**

1. **KB Gap Analysis:**
   - Identify topics with high case volume but low resolution rates
   - Find recurring questions not covered in documentation
   - Prioritize KB articles to create

2. **Bug Pattern Detection:**
   - Identify error patterns suggesting product bugs
   - Correlate with customer feedback mentioning bugs
   - Prioritize by frequency and impact

3. **AI Effectiveness:**
   - Analyze self-service vs escalation rates
   - Identify areas where AI underperforms
   - Suggest prompt or training improvements

4. **Process Recommendations:**
   - Identify bottlenecks in resolution flow
   - Suggest automation opportunities
   - Recommend team training topics

Provide comprehensive analysis with specific, actionable recommendations.
`;

return [{ json: { prompt, analysis_data: data } }];
```

Structured Output Parser Schema

```
{
  "type": "object",
  "properties": {
    "summary": {
      "type": "string",
      "description": "Executive summary of findings"
    },
    "kb_gaps": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "topic": { "type": "string" },
          "case_count": { "type": "number" },
          "priority": { "type": "string", "enum": ["high", "medium", "low"] },
          "suggested_article_title": { "type": "string" },
          "key_points_to_cover": { "type": "array", "items": { "type": "string" } }
        }
      },
```

```
      "description": "Knowledge base documentation gaps"
    },
    "potential_bugs": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "title": { "type": "string" },
          "description": { "type": "string" },
          "error_pattern": { "type": "string" },
          "frequency": { "type": "number" },
          "affected_customers": { "type": "number" },
          "severity": { "type": "string", "enum": ["critical", "high", "medium",
"low"] },
          "reproduction_hints": { "type": "array", "items": { "type": "string" } }
        }
      },
      "description": "Potential product bugs identified"
    },
    "ai_performance": {
      "type": "object",
      "properties": {
        "self_service_rate": { "type": "number" },
        "average_confidence": { "type": "number" },
        "escalation_reasons": { "type": "array", "items": { "type": "string" } },
        "improvement_suggestions": { "type": "array", "items": { "type": "string" }
}
      },
      "description": "AI resolution effectiveness metrics"
    },
    "process_recommendations": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "recommendation": { "type": "string" },
          "rationale": { "type": "string" },
          "effort": { "type": "string", "enum": ["low", "medium", "high"] },
          "impact": { "type": "string", "enum": ["low", "medium", "high"] }
        }
      },
      "description": "Process improvement suggestions"
    },
    "metrics": {
      "type": "object",
      "properties": {
        "total_cases_analyzed": { "type": "number" },
        "resolution_rate": { "type": "number" },
        "average_resolution_time_hours": { "type": "number" },
        "customer_satisfaction_avg": { "type": "number" }
      },
      "description": "Key metrics from the analysis period"
```

```
    }
  },
  "required": ["summary", "kb_gaps", "potential_bugs", "ai_performance"]
}
```

Article Generator Agent

Node Type: `@n8n/n8n-nodes-langchain.agent` **Model:** `claude-sonnet-4-20250514`

System Message

```
You are a technical writer creating knowledge base articles for a B2B software support portal.

**Writing Guidelines:**
1. Use clear, concise language
2. Structure content with headers and bullet points
3. Include step-by-step instructions where applicable
4. Add troubleshooting tips and common pitfalls
5. Reference related articles when relevant

**Article Format:**
1. Overview - Brief description of the topic
2. Prerequisites - What's needed before starting
3. Instructions - Step-by-step guide
4. Troubleshooting - Common issues and solutions
5. Related Topics - Links to related articles

**Style Standards:**
- Professional but approachable tone
- Avoid jargon unless necessary (define if used)
- Use numbered lists for sequential steps
- Use bullet points for non-sequential items
- Include code examples in proper formatting
- Keep paragraphs short (3-4 sentences max)
```

Build Article Prompt (Code Node)

```
const article = $input.first().json.article_to_generate;

const prompt = `
Create a comprehensive knowledge base article for the following topic:

**Article Information:**
- Title: ${article.suggested_article_title || article.topic}
- Topic: ${article.topic}
- Priority: ${article.priority}
- Target Audience: Support agents and end users

**Key Points to Cover:**
${(article.key_points_to_cover || []).map((p, i) => `${i + 1}. ${p}`).join('\n')}
```

```
**Context:**
This article should help reduce support tickets for this topic by providing clear
self-service documentation.

**Requirements:**
1. Start with a clear overview of the topic
2. Include prerequisites if applicable
3. Provide step-by-step instructions
4. Add a troubleshooting section for common issues
5. Keep the article focused and scannable
6. Use proper formatting for Confluence

Write the complete article content in a format suitable for Confluence.
`;

return [{ json: { prompt, article_metadata: article } }];
```

Structured Output Parser Schema

```
{
  "type": "object",
  "properties": {
    "title": {
      "type": "string",
      "description": "Article title"
    },
    "content": {
      "type": "string",
      "description": "Full article content in Confluence-compatible format"
    },
    "summary": {
      "type": "string",
      "description": "Brief summary for search/preview"
    },
    "tags": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Tags for categorization"
    },
    "related_topics": {
      "type": "array",
      "items": { "type": "string" },
      "description": "Related article suggestions"
    }
  },
  "required": ["title", "content", "summary", "tags"]
}
```

Additional Code Nodes Reference

Workflow 5: Format Email Report

Converts markdown report to HTML for email delivery:

```
const analysisData = $input.first().json;

// Convert markdown-style report to HTML
let htmlContent = analysisData.report || analysisData.summary || '';

// Basic markdown to HTML conversion
htmlContent = htmlContent
  .replace(/^### (.*$)/gm, '<h3>$1</h3>')
  .replace(/^## (.*$)/gm, '<h2>$1</h2>')
  .replace(/^# (.*$)/gm, '<h1>$1</h1>')
  .replace(/\*\*(.*?)\*\*/g, '<strong>$1</strong>')
  .replace(/\*(.*?)\*/g, '<em>$1</em>')
  .replace(/^- (.*$)/gm, '<li>$1</li>')
  .replace(/(<li>.*<\/li>)/s, '<ul>$1</ul>')
  .replace(/\n\n/g, '</p><p>')
  .replace(/\n/g, '<br>');

const emailHtml = `
<!DOCTYPE html>
<html>
<head>
  <style>
    body { font-family: Arial, sans-serif; line-height: 1.6; color: #333; }
    h1 { color: #2c3e50; border-bottom: 2px solid #3498db; padding-bottom: 10px; }
    h2 { color: #34495e; margin-top: 20px; }
    h3 { color: #7f8c8d; }
    ul { padding-left: 20px; }
    li { margin-bottom: 5px; }
    .metrics { background: #f8f9fa; padding: 15px; border-radius: 5px; margin: 15px
0; }
    .highlight { background: #fff3cd; padding: 10px; border-left: 4px solid #ffc107;
}
  </style>
</head>
<body>
  <h1>Daily Support Intelligence Report</h1>
  <p><strong>Generated:</strong> ${new Date().toISOString().split('T')[0]}</p>
  <div class="content">
    <p>${htmlContent}</p>
  </div>
  <hr>
  <p><em>This report was automatically generated by CX-Catalyst.</em></p>
</body>
</html>
`;
```

```
return [{ json: { html_report: emailHtml, ...analysisData } }];
```

Workflow 5: Prepare Confluence Content

Escapes content for JSON in HTTP request body:

```
const articleMeta = $('Loop Articles').first().json.article_to_generate;
const generatedContent = $input.first().json;

// Get article content from AI output
const articleContent = generatedContent.content || generatedContent.output || '';
const articleTitle = generatedContent.title || articleMeta.suggested_article_title
|| articleMeta.topic;
const articleTags = generatedContent.tags || ['auto-generated', 'kb-article'];

// Clean and escape content for Confluence JSON body
let cleanContent = articleContent
  .replace(/<[^>]*>/g, '') // Remove any HTML tags
  .trim();

// Escape for JSON
const jsonSafeContent = cleanContent
  .replace(/\\/g, '\\\\')
  .replace(/"/g, '\\"')
  .replace(/\n/g, '\\n')
  .replace(/\r/g, '\\r')
  .replace(/\t/g, '\\t');

const jsonSafeTitle = articleTitle
  .replace(/\\/g, '\\\\')
  .replace(/"/g, '\\"');

return [{
  json: {
    title: articleTitle,
    escaped_title: jsonSafeTitle,
    content: cleanContent,
    escaped_content: jsonSafeContent,
    tags: articleTags,
    topic: articleMeta.topic,
    priority: articleMeta.priority
  }
}];
```

Workflow 5: Format Bug Ticket

Formats bug data for Jira ticket creation:

```
const bugData = $('Loop Bugs').first().json.bug;
```

```
return [{
  json: {
    summary: bugData.title || 'Untitled Bug',
    description: `
**Description:**
${bugData.description || 'No description provided'}

**Error Pattern:**
${bugData.error_pattern || 'Not specified'}

**Frequency:** ${bugData.frequency || 'Unknown'} occurrences
**Affected Customers:** ${bugData.affected_customers || 'Unknown'}
**Severity:** ${bugData.severity || 'medium'}

**Reproduction Hints:**
${(bugData.reproduction_hints || []).map(h => `- ${h}`).join('\n') || '- No
reproduction hints available'}

---
*Auto-generated by CX-Catalyst Pattern Analysis*
    `.trim(),
    severity: bugData.severity || 'medium',
    frequency: bugData.frequency || 0,
    affected_customers: bugData.affected_customers || 0
  }
}];
```

Recovery Instructions

If a workflow needs to be restored:

1. **For Agent Nodes:**
   - Create new LangChain Agent node
   - Set model to specified version
   - Copy System Message from this document
   - Attach appropriate tools (KB Vector Search, etc.)
   - Configure Structured Output Parser with the schema
2. **For Code Nodes:**
   - Create new Code node
   - Set execution mode (Run Once for All Items or Run Once for Each Item)
   - Copy code from this document
3. **For Output Parsers:**
   - Create Structured Output Parser sub-node
   - Set JSON Schema from this document
   - Connect to Agent node

Document generated for CX-Catalyst Project backup - January 2026