next8n

TROUBLESHOOTING GUIDE

Workflow Automation Delivery Framework

ENTERPRISE EDITION

Version: 2.0

Date: December 28, 2025

Author: Mirza Iqbal

Contact: mirza.iqbal@next8n.com

Table of Contents

Troubleshooting Guide

Systematic Problem Resolution for Workflow Automation

Troubleshooting Philosophy

```
+======================================================================+
|                                                                      |
|   "EVERY ERROR IS A CLUE, NOT A DEAD END"                            |
|                                                                      |
|   Systematic troubleshooting beats random guessing every time.       |
|   Document as you go - your future self will thank you.              |
|                                                                      |
+======================================================================+
```

The Troubleshooting Mindset

```
1. STAY CALM        -> Panic leads to poor decisions
2. OBSERVE FIRST    -> Gather information before acting
3. CHANGE ONE THING -> Isolate variables for clear diagnosis
4. DOCUMENT         -> Track what you tried and what happened
5. LEARN            -> Every problem is a learning opportunity
```

Systematic Troubleshooting Methodology

The ISOLATE Framework

```
+----------------------------------------------------------------------+
|                    I.S.O.L.A.T.E. METHOD                             |
+----------------------------------------------------------------------+

I - IDENTIFY the symptoms
    - What exactly is happening?
    - When did it start?
    - Who/what is affected?
    - What changed recently?

S - SCOPE the problem
    - Is it one workflow or many?
    - Is it one node or the whole flow?
    - Is it intermittent or consistent?
    - Is it environment-specific?

O - OBSERVE the evidence
    - Check execution logs
    - Review error messages
    - Examine input/output data
    - Note timing patterns

L - LOCATE the source
    - Use binary search (disable half, test)
    - Check dependencies
    - Trace data flow
    - Identify the failing component

A - ANALYZE root cause
    - Why did this happen?
    - What conditions caused it?
    - Is there a pattern?
    - Could it happen again?

T - TEST the solution
    - Apply fix to isolated environment
    - Verify with known test cases
    - Check edge cases
    - Confirm no side effects

E - EXECUTE and document
    - Deploy the fix
    - Document the solution
    - Update runbooks
    - Share learnings
```

Quick Diagnosis Flowchart

```
WORKFLOW NOT RUNNING?
       |
       v
[Is the workflow active?]
    |         |
   NO        YES
    |         |
    v         v
Activate   [Check trigger type]
the flow        |
         +------+------+
         |             |
     WEBHOOK      SCHEDULED
         |             |
         v             v
    Is URL      Is schedule
    correct?    configured?
         |             |
        ...           ...
```

Diagnostic Questions Checklist

```
SCOPE QUESTIONS:
[ ] When did this start happening?
[ ] What changed recently?
[ ] Is it happening for all inputs or specific ones?
[ ] Can you reproduce it consistently?
[ ] Does it happen in test mode too?

ENVIRONMENT QUESTIONS:
[ ] Which n8n environment (cloud/self-hosted)?
[ ] What version of n8n?
[ ] Any recent updates or changes?
[ ] Are other workflows working?
[ ] Any infrastructure changes?

DATA QUESTIONS:
[ ] What does the input data look like?
[ ] Is the data format correct?
[ ] Are there any null or missing values?
[ ] Is the data size within limits?
[ ] Are special characters causing issues?
```

Common n8n Issues and Solutions

Workflow Execution Issues

Issue: Workflow Not Triggering

```
SYMPTOMS:
- Workflow never starts
- No executions in history
- Webhook not receiving requests

DIAGNOSTIC STEPS:
1. Check workflow is ACTIVE (green toggle)
2. Verify trigger configuration
3. Test trigger manually
4. Check n8n logs for errors

SOLUTIONS BY TRIGGER TYPE:

WEBHOOK:
[ ] URL is correct and complete
[ ] HTTP method matches (POST/GET)
[ ] n8n instance is accessible
[ ] No firewall blocking requests
[ ] Test with curl/Postman

SCHEDULE:
[ ] Cron expression is valid
[ ] Timezone is correct
[ ] Schedule hasn't passed
[ ] n8n instance running at trigger time

POLLING:
[ ] Credentials are valid
[ ] API endpoint is reachable
[ ] Rate limits not exceeded
[ ] Data exists to poll
```

Issue: Workflow Stops Mid-Execution

```
SYMPTOMS:
- Workflow starts but doesn't complete
- Execution marked as "running" indefinitely
- Some nodes execute, others don't

DIAGNOSTIC STEPS:
1. Check execution log for last successful node
2. Look for timeout indicators
3. Check error handling path
4. Review memory usage

COMMON CAUSES:
- API timeout (increase timeout setting)
- Memory exhaustion (reduce batch size)
- Infinite loop (check loop conditions)
- Credential expiry (refresh tokens)
- Rate limit hit (add delays)

SOLUTIONS:
[ ] Increase node timeout settings
[ ] Add retry logic
[ ] Implement chunked processing
[ ] Check for circular references
[ ] Add proper error handling
```

Issue: Duplicate Executions

```
SYMPTOMS:
- Same workflow runs multiple times
- Duplicate data created
- Unexpected multiple triggers

CAUSES AND FIXES:

Cause: Multiple webhook calls
Fix: Add deduplication logic
     - Track request IDs
     - Use idempotency keys

Cause: Retry on timeout
Fix: Implement idempotent operations
     - Check if record exists before creating
     - Use upsert instead of insert

Cause: Webhook testing tools retrying
Fix: Ensure quick response
     - Use respond immediately option
     - Process async if long-running
```

Node-Specific Issues

HTTP Request Node Problems

```
+--------------------------------------------------------------------+
|                     HTTP REQUEST TROUBLESHOOTING                    |
+--------------------------------------------------------------------+

ERROR: Connection Timeout
CAUSES:
- Slow target server
- Network issues
- Firewall blocking

FIXES:
[ ] Increase timeout (default 5000ms)
[ ] Check URL is correct
[ ] Test with curl outside n8n
[ ] Verify firewall rules

--------------------------------------------------------------------

ERROR: 401 Unauthorized
CAUSES:
- Invalid credentials
- Expired token
- Wrong auth method

FIXES:
[ ] Verify API key is correct
[ ] Check token hasn't expired
[ ] Confirm auth type (Basic, Bearer, API Key)
[ ] Regenerate credentials

--------------------------------------------------------------------

ERROR: 403 Forbidden
CAUSES:
- Insufficient permissions
- IP not whitelisted
- Account restrictions

FIXES:
[ ] Check API scopes/permissions
[ ] Whitelist n8n IP address
[ ] Verify account status
[ ] Contact API provider

--------------------------------------------------------------------

ERROR: 404 Not Found
CAUSES:
- Wrong URL/endpoint
- Resource deleted
- API version mismatch
```

```
FIXES:
[ ] Verify endpoint URL
[ ] Check API documentation
[ ] Confirm resource exists
[ ] Update to correct API version

--------------------------------------------------------------------

ERROR: 429 Too Many Requests
CAUSES:
- Rate limit exceeded
- Too many concurrent requests

FIXES:
[ ] Add delay between requests
[ ] Implement exponential backoff
[ ] Reduce batch size
[ ] Check rate limit headers

--------------------------------------------------------------------

ERROR: 500 Internal Server Error
CAUSES:
- Target API has bug
- Malformed request body
- Server overload

FIXES:
[ ] Validate request body format
[ ] Check API status page
[ ] Try again later
[ ] Contact API support

--------------------------------------------------------------------

ERROR: SSL Certificate Error
CAUSES:
- Expired certificate
- Self-signed certificate
- Certificate mismatch

FIXES:
[ ] For testing: disable SSL verification
[ ] For production: fix the certificate
[ ] Check system time is correct
```

Code Node Problems

```
+-------------------------------------------------------------------+
|                      CODE NODE TROUBLESHOOTING                     |
+-------------------------------------------------------------------+

ERROR: "items is not defined"
CAUSE: Using old syntax
FIX: Use $input.all() instead of items

// OLD (deprecated)
for (const item of items) { ... }

// NEW (correct)
for (const item of $input.all()) { ... }

--------------------------------------------------------------------

ERROR: "Cannot read property 'X' of undefined"
CAUSE: Accessing missing data
FIX: Add null checks

// WRONG
const value = item.json.nested.property;

// RIGHT
const value = item.json?.nested?.property ?? 'default';

--------------------------------------------------------------------

ERROR: "Unexpected token"
CAUSE: Syntax error
FIX: Check for:
- Missing commas
- Missing brackets
- Invalid JSON in code
- Copy/paste encoding issues

--------------------------------------------------------------------

ERROR: "Maximum call stack size exceeded"
CAUSE: Infinite recursion
FIX: Check recursive functions have exit conditions

--------------------------------------------------------------------

ERROR: Return value issues
FIX: Always return items in correct format

// WRONG
return data;

// RIGHT
```

```
return [{ json: data }];

// OR for multiple items
return items.map(item => ({
  json: { ...item.json, newField: 'value' }
}));
```

IF Node Problems

```
COMMON ISSUES:

Problem: Wrong branch taken
Check:
[ ] Data type matches (string vs number)
[ ] Case sensitivity for strings
[ ] Whitespace in values
[ ] Expression syntax correct

Problem: Expression not evaluating
Check:
[ ] Using {{ }} correctly
[ ] Proper reference to node
[ ] Field name spelled correctly

DEBUGGING TIP:
Add a Set node before IF to output the value being compared
```

Data Issues

Issue: Data Not Passing Between Nodes

```
SYMPTOMS:
- Empty data in next node
- "No items to process" message
- Missing fields

DIAGNOSTIC STEPS:
1. Check previous node output
2. Verify expression references
3. Check for data transformation issues

COMMON CAUSES AND FIXES:

Cause: Wrong node reference
Fix: Use correct node name in expression
     {{ $('Correct Node Name').item.json.field }}

Cause: Item position mismatch
Fix: Use .first() or .all() appropriately
     {{ $('Node').first().json.field }}

Cause: Empty array returned
Fix: Check if previous node produced output
     Add error handling for empty results

Cause: Async timing issues
Fix: Ensure proper wait/queue configuration
```

Issue: Data Format Problems

```
+-------------------------------------------------------------------+
|                     DATA FORMAT SOLUTIONS                           |
+-------------------------------------------------------------------+

PROBLEM: String instead of JSON
---------------------------------
INPUT:  '{"name": "John"}'
NEEDED: {name: "John"}

FIX:
const data = JSON.parse($input.first().json.stringField);
return [{ json: data }];

-------------------------------------------------------------------

PROBLEM: JSON instead of String
---------------------------------
INPUT:  {name: "John"}
NEEDED: '{"name": "John"}'

FIX:
const str = JSON.stringify($input.first().json);
return [{ json: { data: str } }];

-------------------------------------------------------------------

PROBLEM: Date format wrong
---------------------------------
INPUT:  "2024-01-15T10:30:00Z"
NEEDED: "January 15, 2024"

FIX (Code node):
const date = new Date($input.first().json.date);
const formatted = date.toLocaleDateString('en-US', {
  year: 'numeric',
  month: 'long',
  day: 'numeric'
});
return [{ json: { date: formatted } }];

-------------------------------------------------------------------

PROBLEM: Nested data flattening
---------------------------------
INPUT:  {user: {name: "John", email: "j@test.com"}}
NEEDED: {userName: "John", userEmail: "j@test.com"}

FIX (Set node):
userName: {{ $json.user.name }}
userEmail: {{ $json.user.email }}
```

```
--------------------------------------------------------------------

PROBLEM: Array needs to be items
---------------------------------
INPUT:  {items: [{...}, {...}]}
NEEDED: Each array element as separate item

FIX (Split Out node):
Field to Split: items
Include: No (to not include parent)
```

Integration-Specific Troubleshooting

Webhook Issues

```
+======================================================================+
|                    WEBHOOK TROUBLESHOOTING                           |
+======================================================================+

NOT RECEIVING WEBHOOKS:
-----------------------
[ ] Workflow is active
[ ] URL is correct (check for typos)
[ ] Using correct HTTP method
[ ] n8n instance is publicly accessible
[ ] No SSL/TLS issues
[ ] Firewall allows incoming connections
[ ] Sender service can reach your n8n

DEBUG STEPS:
1. Get webhook URL from n8n
2. Test with curl:
   curl -X POST https://your-n8n.com/webhook/xxx \
     -H "Content-Type: application/json" \
     -d '{"test": "data"}'
3. Check n8n executions for the request
4. If no execution, check n8n logs

--------------------------------------------------------------------

WEBHOOK RETURNING WRONG DATA:
-----------------------------
[ ] Response node configured correctly
[ ] Respond to Webhook node at end of flow
[ ] Data being returned exists
[ ] Content-Type header correct

--------------------------------------------------------------------

WEBHOOK TIMING OUT:
-------------------
Cause: Long-running process

Fix 1: Use "Respond Immediately"
- Add response node right after webhook trigger
- Process rest of workflow async

Fix 2: Queue-based architecture
- Webhook accepts and queues
- Separate workflow processes queue

--------------------------------------------------------------------

WEBHOOK SECURITY ISSUES:
------------------------
[ ] HTTPS enabled (not HTTP)
```

```
  [ ] Authentication configured
  [ ] IP whitelist if possible
  [ ] Rate limiting in place
  [ ] Input validation implemented
```

API Authentication Issues

```
+======================================================================+
|                  AUTHENTICATION TROUBLESHOOTING                      |
+======================================================================+

OAUTH2 TOKEN EXPIRED:
---------------------
Symptoms: Previously working suddenly fails with 401

Solutions:
1. Reconnect credential in n8n
2. Re-authorize OAuth flow
3. Check if refresh token is working
4. Verify OAuth app still active

--------------------------------------------------------------------

API KEY NOT WORKING:
--------------------
Check:
[ ] Key is active in provider dashboard
[ ] Key has correct permissions/scopes
[ ] Key isn't rate limited
[ ] Key format is correct (no extra spaces)
[ ] Using correct header/param name

--------------------------------------------------------------------

CREDENTIAL ERRORS:
------------------
"Could not find credential"
-> Credential was deleted or renamed
-> Fix: Update credential reference

"Could not decrypt credentials"
-> n8n encryption key changed
-> Fix: Re-enter credential values

"Invalid credentials"
-> Credentials are wrong
-> Fix: Verify and update credentials

--------------------------------------------------------------------

SERVICE ACCOUNT ISSUES:
-----------------------
Check:
[ ] Service account enabled
[ ] Correct permissions assigned
[ ] Key file format correct
[ ] Account not suspended
[ ] Quota not exceeded
```

Email Integration Issues

```
SMTP NOT SENDING:
-----------------
Check:
[ ] SMTP host and port correct
[ ] SSL/TLS setting matches port
[ ] Username and password correct
[ ] Sender email allowed
[ ] Not blocked by spam filters

Common Port Configurations:
- Port 25: Plain (often blocked)
- Port 465: SSL/TLS
- Port 587: STARTTLS (recommended)

--------------------------------------------------------------------

EMAILS GOING TO SPAM:
---------------------
Improve deliverability:
[ ] Set up SPF record
[ ] Set up DKIM
[ ] Set up DMARC
[ ] Use verified sending domain
[ ] Avoid spam trigger words
[ ] Include unsubscribe option

--------------------------------------------------------------------

GMAIL SPECIFIC:
---------------
[ ] Less secure apps enabled OR
[ ] App password created (if 2FA)
[ ] Daily sending limit not exceeded (500/day personal)
```

Database Integration Issues

```
+======================================================================+
|                    DATABASE TROUBLESHOOTING                          |
+======================================================================+

CONNECTION REFUSED:
-------------------
Causes:
- Wrong host/port
- Database not running
- Firewall blocking
- Wrong credentials

Fixes:
[ ] Verify connection string
[ ] Test with external client
[ ] Check database service status
[ ] Verify firewall rules
[ ] Check IP whitelist

--------------------------------------------------------------------

QUERY TIMEOUT:
--------------
Causes:
- Query too complex
- Missing indexes
- Large result set
- Database overloaded

Fixes:
[ ] Optimize query
[ ] Add appropriate indexes
[ ] Limit results (LIMIT clause)
[ ] Increase timeout setting
[ ] Run during off-peak hours

--------------------------------------------------------------------

PERMISSION DENIED:
------------------
Causes:
- User lacks permissions
- Wrong database selected
- Table doesn't exist

Fixes:
[ ] GRANT necessary permissions
[ ] Verify database name
[ ] Check table exists
[ ] Use correct schema
```

```
--------------------------------------------------------------------

DATA TYPE MISMATCH:
-------------------
Symptoms: Insert/update fails

Fixes:
[ ] Check column data types
[ ] Convert data before insert
[ ] Handle NULL values
[ ] Match date/time formats
```

Performance Issues and Optimization

Diagnosing Performance Problems

```
PERFORMANCE METRICS TO MONITOR:
-------------------------------
- Execution time (total and per-node)
- Memory usage
- API response times
- Queue depth (if applicable)
- Error rate percentage

SLOW WORKFLOW INVESTIGATION:
1. Open execution details
2. Check time spent per node
3. Identify slowest nodes
4. Analyze why they're slow
5. Optimize or parallelize
```

Common Performance Problems

```
+======================================================================+
|                     PERFORMANCE OPTIMIZATION                          |
+======================================================================+

PROBLEM: Loop processing too slow
---------------------------------
Cause: Sequential processing of many items
Solution: Batch processing

Before (slow):
Loop -> HTTP Request (one at a time)

After (faster):
Split in batches -> HTTP Request (batch)

--------------------------------------------------------------------

PROBLEM: Too many API calls
---------------------------
Cause: N+1 query pattern
Solution: Bulk APIs when available

Before (N+1):
Get list -> Loop -> Get details for each

After (bulk):
Get list -> Bulk get all details

--------------------------------------------------------------------

PROBLEM: Large data sets crashing
---------------------------------
Cause: Memory exhaustion
Solutions:
[ ] Process in chunks
[ ] Stream data instead of loading all
[ ] Increase memory limit (self-hosted)
[ ] Use pagination

--------------------------------------------------------------------

PROBLEM: Webhook response timeout
---------------------------------
Cause: Long processing time
Solutions:
[ ] Respond immediately, process async
[ ] Queue-based architecture
[ ] Optimize slow operations

--------------------------------------------------------------------
```

```
PROBLEM: High execution costs
-----------------------------
Causes:
- Too many unnecessary executions
- Over-processing data
- Redundant API calls

Solutions:
[ ] Add filters early in workflow
[ ] Cache frequently accessed data
[ ] Deduplicate before processing
[ ] Use cheaper AI models where appropriate
```

Optimization Checklist

```
WORKFLOW OPTIMIZATION:
[ ] Filter data as early as possible
[ ] Use batch operations when available
[ ] Implement caching for repeated lookups
[ ] Parallelize independent operations
[ ] Remove unnecessary nodes
[ ] Optimize expressions for readability and performance

API CALL OPTIMIZATION:
[ ] Minimize number of API calls
[ ] Use bulk endpoints when available
[ ] Implement rate limiting to avoid 429s
[ ] Cache responses when appropriate
[ ] Use webhooks instead of polling where possible

DATA HANDLING OPTIMIZATION:
[ ] Limit fields retrieved (select only needed)
[ ] Use pagination for large datasets
[ ] Process in batches to avoid memory issues
[ ] Clean up temporary data
```

Error Message Reference and Solutions

n8n System Errors

```
+======================================================================+
|                       ERROR REFERENCE TABLE                          |
+======================================================================+

ERROR: "ECONNREFUSED"
---------------------
Meaning: Connection refused by target server
Common causes:
- Server not running
- Wrong port
- Firewall blocking
Solution: Verify server is accessible

--------------------------------------------------------------------

ERROR: "ETIMEDOUT"
------------------
Meaning: Connection timed out
Common causes:
- Slow network
- Server overloaded
- Firewall dropping packets
Solution: Increase timeout, check connectivity

--------------------------------------------------------------------

ERROR: "ENOTFOUND"
------------------
Meaning: DNS lookup failed
Common causes:
- Typo in hostname
- DNS server issues
- Hostname doesn't exist
Solution: Verify hostname is correct

--------------------------------------------------------------------

ERROR: "Execution timed out"
----------------------------
Meaning: Workflow exceeded time limit
Common causes:
- Infinite loop
- Very slow operations
- Waiting for response
Solution: Optimize workflow, increase timeout

--------------------------------------------------------------------

ERROR: "Out of memory"
----------------------
Meaning: Node.js ran out of memory
```

```
Common causes:
- Processing too much data
- Memory leak
- Recursive operations
Solution: Process in batches, increase memory

--------------------------------------------------------------------

ERROR: "Cannot find credential"
-------------------------------
Meaning: Referenced credential missing
Common causes:
- Credential deleted
- Credential renamed
- Wrong environment
Solution: Recreate or update credential reference

--------------------------------------------------------------------

ERROR: "Workflow could not be started"
--------------------------------------
Meaning: Trigger failed to initialize
Common causes:
- Invalid trigger configuration
- Credential issues
- Missing dependencies
Solution: Check trigger settings and credentials
```

Integration-Specific Errors

```
GOOGLE APIS:
------------
"Invalid grant" -> OAuth expired, reconnect
"Quota exceeded" -> API quota hit, wait or upgrade
"Access denied" -> Permission issue, check scopes

SLACK:
------
"channel_not_found" -> Wrong channel ID or bot not in channel
"not_in_channel" -> Add bot to channel first
"rate_limited" -> Too many requests, add delay

AIRTABLE:
---------
"INVALID_REQUEST_UNKNOWN" -> Check field names and types
"TABLE_NOT_FOUND" -> Verify table name/ID
"VIEW_NOT_FOUND" -> Verify view name/ID

NOTION:
-------
"object_not_found" -> Page/database doesn't exist or no access
"validation_error" -> Check property names and formats
"rate_limited" -> Slow down requests

HUBSPOT:
--------
"RATE_LIMIT" -> Too many requests, implement backoff
"INVALID_PROPERTY" -> Check property internal name
"OBJECT_NOT_FOUND" -> Verify record ID exists
```

Debugging Techniques

Using n8n's Built-in Tools

```
EXECUTION PREVIEW (Cmd/Ctrl + Enter):
-------------------------------------
- Test individual nodes
- See output before running full workflow
- Debug expressions

EXPRESSION EDITOR:
------------------
- Test expressions with real data
- See available fields
- Debug JavaScript snippets

PINNING DATA:
-------------
- Pin output from a node
- Use pinned data for testing
- Avoid hitting APIs repeatedly

EXECUTION LOG:
--------------
- See all historical executions
- Filter by status (success/error)
- View detailed input/output
```

Debugging Strategies

```
STRATEGY 1: BINARY SEARCH
-------------------------
1. Disable half the workflow
2. Run and check if error occurs
3. If error: problem in active half
4. If no error: problem in disabled half
5. Repeat until isolated

STRATEGY 2: TRACE LOGGING
-------------------------
Add Set nodes to log intermediate values:

Set Node: "Debug - After Fetch"
- execution_id: {{ $execution.id }}
- data_count: {{ $input.all().length }}
- first_item: {{ JSON.stringify($json) }}

STRATEGY 3: ERROR ISOLATION
---------------------------
Wrap suspicious sections in error branches:

[Risky Operation]
       |
       v
[Error Trigger] --> [Log Error Details]
       |
       v
[Continue/Retry]

STRATEGY 4: MINIMAL REPRODUCTION
--------------------------------
1. Export problematic workflow
2. Remove unrelated nodes
3. Simplify until smallest case that fails
4. Debug that minimal case
```

Code Node Debugging

```javascript
// Add debug output to Code nodes

// Log input data
console.log('Input:', JSON.stringify($input.all(), null, 2));

// Log intermediate values
const data = processData($input.first().json);
console.log('Processed:', data);

// Log before return
const result = { json: data };
console.log('Returning:', result);

return [result];

// Note: console.log outputs appear in node output
// when using "Execute Node" or in execution logs
```

Log Analysis Procedures

n8n Execution Logs

```
ACCESSING LOGS:
---------------
n8n Cloud: Executions tab in workflow editor
Self-hosted: /var/log/n8n/ or Docker logs

KEY INFORMATION IN LOGS:
- Timestamp
- Execution ID
- Workflow ID
- Node that executed
- Input data
- Output data
- Error details

LOG ANALYSIS STEPS:
1. Identify the failed execution
2. Find the failing node
3. Examine input data to that node
4. Check error message
5. Compare with successful executions
```

Creating a Logging System

```
EXECUTION LOGGING SETUP:
------------------------

Create a workflow that logs to Google Sheets or database:

Columns to track:
- timestamp: ISO date/time
- execution_id: n8n execution ID
- workflow_name: which workflow
- status: success/error
- input_summary: first 500 chars
- output_summary: first 500 chars
- error_message: if applicable
- duration_ms: execution time
- tokens_used: for AI nodes

Example Code Node for logging:

const logEntry = {
  timestamp: new Date().toISOString(),
  execution_id: $execution.id,
  workflow_name: $workflow.name,
  status: 'success',
  input_summary: JSON.stringify($input.first().json).substring(0, 500),
  duration_ms: Date.now() - $execution.startTime
};

return [{ json: logEntry }];
```

Log Patterns to Watch

```
WARNING PATTERNS:
-----------------
Pattern: Increasing error rate
Action: Investigate root cause

Pattern: Specific time-based failures
Action: Check scheduled maintenance, rate limits

Pattern: Same error repeating
Action: Fix systematic issue

Pattern: Errors after deployment
Action: Rollback or quick fix

Pattern: Memory growth over time
Action: Check for leaks, restart

Pattern: Increasing latency
Action: Performance investigation

USEFUL LOG SEARCHES:
--------------------
Find all errors: status:error
Find specific workflow: workflow_name:"My Workflow"
Find recent: timestamp > yesterday
Find slow executions: duration_ms > 30000
```

When to Escalate vs. Self-Solve

Escalation Decision Matrix

```
+======================================================================+
|                      ESCALATION DECISION MATRIX                      |
+======================================================================+

SELF-SOLVE (Don't Escalate):
----------------------------
[ ] Configuration errors
[ ] Credential issues
[ ] Data format problems
[ ] Simple logic bugs
[ ] Documentation gaps
[ ] Rate limit issues
[ ] Timeout adjustments

CONSIDER ESCALATING:
--------------------
[ ] Issue persists after multiple attempts
[ ] Affecting multiple clients
[ ] Requires vendor support
[ ] Security implications
[ ] Data integrity at risk
[ ] Beyond your expertise

MUST ESCALATE:
--------------
[ ] Data breach suspected
[ ] System completely down
[ ] Affecting production revenue
[ ] Legal/compliance issue
[ ] Requires infrastructure access you don't have
```

Self-Solve Time Limits

```
TIME-BOXING TROUBLESHOOTING:
----------------------------

Minor Issue (doesn't affect core function):
- Self-solve time: 1-2 hours
- Then: Document and move on

Moderate Issue (affects some functionality):
- Self-solve time: 2-4 hours
- Then: Escalate or get help

Critical Issue (system down):
- Self-solve time: 30 minutes
- Then: Immediate escalation

BEFORE ESCALATING, HAVE:
[ ] Clear problem description
[ ] Steps to reproduce
[ ] What you've tried
[ ] Relevant logs/screenshots
[ ] Impact assessment
```

Escalation Paths

```
ESCALATION HIERARCHY:
---------------------

Level 1: Self-Research
- n8n documentation
- Community forum
- Stack Overflow
- Past similar issues

Level 2: Team/Peer Help
- Team Slack/chat
- Senior developer
- Technical lead

Level 3: Vendor Support
- n8n support (if on paid plan)
- Integration vendor support
- Cloud provider support

Level 4: Emergency
- Direct phone to vendor
- Consultant/expert network
- Rollback to last working state
```

Client Communication During Issues

Communication Templates

```
INITIAL NOTIFICATION:
---------------------
Subject: [AWARENESS] Issue Detected - [Brief Description]

Hi [Name],

I've identified an issue with [workflow/system] and wanted to
let you know I'm actively working on it.

WHAT'S HAPPENING:
[Brief, non-technical description]

IMPACT:
[What's affected, what's working]

CURRENT STATUS:
Investigating the root cause. I'll update you within [timeframe].

QUESTIONS?
Reply to this email or call [number].

--------------------------------------------------------------------

PROGRESS UPDATE:
----------------
Subject: [UPDATE] [Issue] - Progress Report

Hi [Name],

Update on the issue I reported earlier:

STATUS: [Investigating / Identified cause / Implementing fix]

WHAT WE KNOW:
[Technical details in plain language]

WHAT'S BEING DONE:
[Current action]

EXPECTED RESOLUTION:
[Timeframe if known]

I'll keep you posted.

--------------------------------------------------------------------

RESOLUTION NOTIFICATION:
------------------------
Subject: [RESOLVED] [Issue] - Fixed

Hi [Name],
Good news - the issue is now resolved.

WHAT HAPPENED:
[Root cause explanation]

WHAT WE DID:
[Solution implemented]

PREVENTION:
[Steps taken to prevent recurrence]

Please test and confirm everything is working. Let me know
if you notice any issues.
```

Communication Principles

```
DOS:
----
[X] Notify proactively (don't wait for client to notice)
[X] Use plain language (avoid jargon)
[X] Give realistic timelines
[X] Own the problem (no blame)
[X] Provide regular updates
[X] Confirm resolution

DON'TS:
-------
[ ] Ignore the problem
[ ] Over-promise quick fixes
[ ] Use technical jargon
[ ] Blame third parties excessively
[ ] Go silent during investigation
[ ] Close without confirming fix
```

Severity-Based Communication

```
+--------------------------------------------------------------------+
|                   COMMUNICATION BY SEVERITY                          |
+--------------------------------------------------------------------+

CRITICAL (System Down):
-----------------------
Notify: Immediately
Channel: Phone + Email
Updates: Every 30 minutes
Tone: Urgent but calm

HIGH (Major Feature Broken):
----------------------------
Notify: Within 1 hour
Channel: Email + Slack
Updates: Every 2 hours
Tone: Professional urgency

MEDIUM (Partial Issue):
-----------------------
Notify: Same business day
Channel: Email
Updates: Daily
Tone: Informative

LOW (Minor Issue):
------------------
Notify: At next regular touchpoint
Channel: Email or meeting
Updates: With resolution
Tone: Casual
```

Prevention Strategies

Proactive Monitoring

```
IMPLEMENT THESE MONITORS:
-------------------------

1. EXECUTION ERROR RATE
   Alert when: Error rate > 10%
   Check: Daily

2. CONSECUTIVE FAILURES
   Alert when: 3+ failures in a row
   Check: Real-time

3. EXECUTION TIME ANOMALY
   Alert when: 2x normal duration
   Check: Per execution

4. API QUOTA USAGE
   Alert when: > 80% quota used
   Check: Daily

5. CREDENTIAL EXPIRY
   Alert when: Token expires in < 7 days
   Check: Weekly

6. WORKFLOW INACTIVITY
   Alert when: No executions in expected period
   Check: Daily
```

Error Prevention Checklist

```
DURING DEVELOPMENT:
[ ] Add error handling to all nodes
[ ] Validate all inputs
[ ] Test with edge cases
[ ] Add timeout handling
[ ] Implement retry logic
[ ] Log important events

BEFORE DEPLOYMENT:
[ ] Full end-to-end testing
[ ] Load testing if high volume
[ ] Credential verification
[ ] Rollback plan documented
[ ] Monitoring configured

AFTER DEPLOYMENT:
[ ] Watch first few executions
[ ] Check for errors
[ ] Verify performance
[ ] Confirm expected behavior
```

Common Pitfall Prevention

```
+=====================================================================+
|                        COMMON PITFALLS                              |
+=====================================================================+

PITFALL: Hardcoded values
-------------------------
Problem: Values that change break workflow
Prevention: Use variables and configuration nodes

PITFALL: No error handling
--------------------------
Problem: Failures cascade or go unnoticed
Prevention: Add error branches to all critical nodes

PITFALL: Missing null checks
----------------------------
Problem: Crashes on unexpected missing data
Prevention: Always use optional chaining (?.)

PITFALL: Assuming API availability
----------------------------------
Problem: Workflow fails when API is down
Prevention: Implement retries and fallbacks

PITFALL: No input validation
----------------------------
Problem: Bad data causes downstream failures
Prevention: Validate all external inputs early

PITFALL: Unhandled rate limits
------------------------------
Problem: Burst of 429 errors
Prevention: Implement delays and backoff

PITFALL: Missing timeouts
-------------------------
Problem: Workflow hangs indefinitely
Prevention: Set appropriate timeouts on all operations
```

Monitoring and Alerting Setup

Basic Monitoring Workflow

```
MONITORING WORKFLOW STRUCTURE:
------------------------------

[Schedule Trigger: Every 5 min]
         |
         v
[Check Target Workflow Status]
         |
    +----+----+
    |         |
  ERROR      OK
    |         |
    v         v
[Send Alert] [Log Success]
```

Alert Configuration

```
ALERT CHANNELS:
---------------
- Email (for non-urgent)
- Slack/Teams (for urgent)
- SMS (for critical)
- PagerDuty (for on-call)

ALERT CONTENT TEMPLATE:
-----------------------
Workflow: [Name]
Status: [ERROR/WARNING]
Time: [Timestamp]
Error: [Message]
Execution: [Link to execution]
Action: [What to do]

ALERT FATIGUE PREVENTION:
-------------------------
[ ] Set sensible thresholds
[ ] Group similar alerts
[ ] Implement cooldown periods
[ ] Distinguish severity levels
[ ] Allow alert acknowledgment
```

Monitoring Dashboard Setup

```
RECOMMENDED METRICS TO DISPLAY:
-------------------------------

Real-Time:
- Active workflows count
- Current executions
- Error count (last hour)
- Queue depth

Historical:
- Executions over time (chart)
- Success rate (chart)
- Error rate by workflow (table)
- Average execution time (chart)

Alerts:
- Open issues
- Recent alerts
- Alert history
```

Useful n8n Tips and Shortcuts

Keyboard Shortcuts

```
+======================================================================+
|                      N8N KEYBOARD SHORTCUTS                          |
+======================================================================+

GENERAL:
--------
Ctrl/Cmd + S        Save workflow
Ctrl/Cmd + Enter    Execute workflow
Ctrl/Cmd + A        Select all nodes
Ctrl/Cmd + C        Copy selected
Ctrl/Cmd + V        Paste
Ctrl/Cmd + Z        Undo
Ctrl/Cmd + Shift+Z  Redo
Ctrl/Cmd + D        Duplicate selection
Delete/Backspace    Delete selected

NAVIGATION:
-----------
+/-                 Zoom in/out
Ctrl/Cmd + 0        Reset zoom
Space + Drag        Pan canvas
Ctrl/Cmd + F        Search nodes

NODE OPERATIONS:
----------------
Tab                 Open node selector
Enter               Open selected node
Escape              Close node panel
D                   Disable/Enable node
P                   Pin node output
```

Pro Tips

```
+======================================================================+
|                          PRO TIPS                                    |
+======================================================================+

TIP 1: USE STICKY NOTES
-----------------------
Add Sticky Notes to document:
- What each section does
- Configuration requirements
- Known limitations
- Contact info for issues

TIP 2: COLOR CODE NODES
-----------------------
Use consistent colors:
- Blue: Data input/triggers
- Green: Processing
- Yellow: Conditionals
- Red: Error handling
- Purple: AI operations
- Gray: Utilities

TIP 3: NAME NODES DESCRIPTIVELY
-------------------------------
Bad:  HTTP Request, Code, Set
Good: Fetch Customer Data, Transform Response, Set Output Fields

TIP 4: USE SUBWORKFLOWS
-----------------------
Extract repeated patterns into subworkflows:
- Error handling
- Logging
- Common API calls
- Standard transforms

TIP 5: PIN DATA FOR TESTING
---------------------------
Pin output from expensive nodes (API calls, AI):
- Saves time during development
- Reduces API costs
- Enables offline testing

TIP 6: EXPRESSION SHORTCUTS
---------------------------
{{ $json.field }}           Current item's field
{{ $('Node').item }}        Reference other node
{{ $input.first() }}        First input item
{{ $input.all() }}          All input items
{{ $execution.id }}         Current execution ID
{{ $now }}                  Current timestamp
{{ $today }}                Today's date
```

```
TIP 7: DEBUG WITH SET NODES
---------------------------
Add Set nodes to:
- See intermediate values
- Store debug info
- Track execution flow
```

Common Patterns

```
PATTERN: RETRY WITH BACKOFF
---------------------------
[API Call]
    |
[IF Error?]
    |yes
    v
[Wait (exponential)]
    |
[Retry Counter < Max?]
    |yes
    v
[Loop back to API Call]

---------------------------------------------------------------------

PATTERN: DEDUPLICATION
----------------------
[Incoming Items]
    |
    v
[Code: Check seen IDs]
    |
    v
[IF New?]
    |yes
    v
[Process]
    |
    v
[Store ID as seen]

---------------------------------------------------------------------

PATTERN: ERROR AGGREGATION
--------------------------
[Multiple Operations]
    |
[Collect Errors]
    |
[IF Any Errors?]
    |yes
    v
[Send Error Summary]

---------------------------------------------------------------------

PATTERN: RATE LIMIT HANDLING
----------------------------
[Batch Items]
    |
    v
[Loop with Delay]
    |
    v
[API Call]
    |
[IF 429?] --> [Wait] --> [Retry]
    |no
    v
[Continue]
```

Quick Reference Card

```
+======================================================================+
|                    TROUBLESHOOTING QUICK REFERENCE                   |
+======================================================================+

FIRST STEPS:
1. Check if workflow is active
2. Look at execution logs
3. Identify the failing node
4. Check error message
5. Verify credentials

COMMON FIXES:
- Credential issue -> Reconnect/refresh
- Timeout -> Increase timeout setting
- Rate limit -> Add delays
- Data format -> Check and transform
- Missing data -> Add null checks

ESCALATE IF:
- Persists after 2+ hours
- Affects production revenue
- Security concern
- Beyond your expertise

COMMUNICATE:
- Proactively notify client
- Use plain language
- Give realistic timelines
- Provide regular updates

LOG EVERYTHING:
- What you tried
- What worked/didn't
- Root cause found
- Solution implemented
```

Troubleshooting Checklist Summary

```
INITIAL DIAGNOSIS:
[ ] Identify the symptoms clearly
[ ] Check if workflow is active
[ ] Review execution logs
[ ] Identify the failing node
[ ] Read the error message carefully

INVESTIGATION:
[ ] Check recent changes
[ ] Verify credentials
[ ] Test with known-good input
[ ] Check integration status pages
[ ] Review rate limits

RESOLUTION:
[ ] Implement fix in test
[ ] Verify fix works
[ ] Deploy to production
[ ] Monitor for recurrence
[ ] Document the solution

POST-MORTEM:
[ ] Document root cause
[ ] Update runbooks
[ ] Implement prevention
[ ] Share learnings
[ ] Update monitoring
```

Next: See `12-advanced-patterns.md` for complex workflow patterns.