Dealing with errors in workflows

# Dealing with errors in workflows#

Sometimes you build a nice workflow, but it fails when you try to execute it. Workflow executions may fail for a variety of reasons, ranging from straightforward problems with incorrectly configuring a node or a failure in a third-party service to more mysterious errors.

But don't panic. In this lesson, you'll learn how you can troubleshoot errors so you can get your workflow up and running as soon as possible.

## Checking failed workflows#

n8n tracks executions of your workflows.

When one of your workflows fails, you can check the Executions log to see what went wrong. The Executions log shows you a list of the latest execution time, status, mode, and running time of your saved workflows.

Open the Executions log by selecting Executions in the left-side panel.

To investigate a specific failed execution from the list, select the name or the View button that appears when you hover over the row of the respective execution.

This will open the workflow in read-only mode, where you can see the execution of each node. This representation can help you identify at what point the workflow ran into issues.

To toggle between viewing the execution and the editor, select the Editor | Executions button at the top of the page.

## Catching erroring workflows#

To catch failed workflows, create a separate Error Workflow with the Error Trigger node. This workflow will only execute if the main workflow execution fails.

Use additional nodes in your Error Workflow that make sense, like sending notifications about the failed workflow and its errors using email or Slack.

To receive error messages for a failed workflow, set the Error Workflow in the Workflow Settings to an Error Workflow that uses an Error Trigger node.

The only difference between a regular workflow and an Error Workflow is that the latter contains an Error Trigger node. Make sure to create this node before you set this as another workflow's designated Error Workflow.

Error workflows

• If a workflow uses the Error Trigger node, you don't have to activate the workflow.

• If a workflow contains the Error Trigger node, by default, the workflow uses itself as the error workflow.

• You can't test error workflows when running workflows manually. The Error trigger only runs when an automatic workflow errors.

• You can set the same Error Workflow for multiple workflows.

### Exercise#

In the previous chapters, you've built several small workflows. Now, pick one of them that you want to monitor and create an Error Workflow for it:

• Create a new Error Workflow.

• Add the Error Trigger node.

• Connect a node for the communication platform of your choice to the Error Trigger node, like Slack, Discord, Telegram, or even Gmail or a more generic Send Email.

• In the workflow you want to monitor, open the Workflow Settings and select the new Error Workflow you just created. Note that this workflow needs to run automatically to trigger the error workflow.

## Throwing exceptions in workflows#

Another way of troubleshooting workflows is to include a Stop and Error node in your workflow. This node throws an error. You can specify the error type:

• Error Message: returns a custom message about the error

• Error Object: returns the type of error

You can only use the Stop and Error node as the last node in a workflow.

When to throw errors

Throwing exceptions with the Stop and Error node is useful for verifying the data (or assumptions about the data) from a node and returning custom error messages.

If you are working with data from a third-party service, you may come across problems such as:

• Wrongly formatted JSON output

• Data with the wrong type (for example, numeric data that has a non-numeric value)

• Missing values

• Errors from remote servers

Though this kind of invalid data might not cause the workflow to fail right away, it could cause problems later on, and then it can become difficult to track the source error. This is why it's better to throw an error at the time you know there might be a problem.