

## ✓What is Error Handling in Zapier?

Error handling means deciding **what should happen if a step in your Zap fails**.

By default, if any step fails, the Zap **stops** and the task is marked as an error.

But Zapier gives you tools to **control and manage errors gracefully**.

---

## ✓Why is it important?

- Prevent data loss.
  - Avoid breaking your automation.
  - Handle exceptions (e.g., missing fields, API errors) without manual intervention.
- 

## ✓Ways to Handle Errors in Zapier

Zapier offers several built-in options:

---

### 1. Default Behavior

- If a step fails, Zap stops.
  - You can see the error in **Task History** and retry manually.
- 

### 2. Use “Error Handling” Settings

Zapier has an **Error Handling** section for each step:

- **Stop Zap** (default): Zap ends if this step fails.
  - **Continue Zap**: Zap ignores the error and moves to the next step.
  - **Custom Error Handling**: Combine with filters or paths for advanced logic.
- 

### 3. Add a Fallback Path

- Use **Paths by Zapier**:
    - Main Path → Normal flow.
    - Fallback Path → Runs if something goes wrong (e.g., log error to Google Sheets or send an alert to Slack).
- 

### 4. Use “Filter” or “Exists” Checks

- Before a step, add a **Filter**:
  - Example: Only continue if email exists.

- This prevents steps from failing due to missing data.
- 

### 5. Use “Catch Hook” or “Catch Error”

- For advanced users:
    - Use **Webhooks** or **Code by Zapier** to catch and handle errors programmatically.
- 

### 6. Notifications

- Add a step at the end:
    - Send an email or Slack message if something fails.
    - Helps you monitor issues quickly.
- 

### ✓Example Scenario

You have a Zap:

- Trigger: New order.
- Step 1: Add order to CRM.
- Step 2: Send confirmation email.

**Problem:** If CRM API fails, email never goes out.

**Solution:**

- Set CRM step to “**Continue Zap if error**”.
  - Add a **Fallback Path**:
    - If CRM fails → Log to Google Sheet + Notify team on Slack.
- 

### ✓Best Practices

- Always add **error notifications** for critical Zaps.
  - Use **Fallback Paths** for important workflows.
  - Validate data early with **Filters**.
  - Check **Task History** regularly for errors.
- 

### ✓Limitations

- Zapier doesn’t have full “try/catch” like programming languages.
- Complex error handling may require **Code by Zapier** or external tools.