Node Guide: Condition

Overview

The **Condition** node helps in controlling the flow of your workflow based on logical checks—just like an **if-else block** in programming. It evaluates one or more variables against specific conditions and decides which path the workflow should follow next.

This allows you to build dynamic flows where different outcomes depend on variable values.

How It Works

When this node runs:

- It checks values stored in variables based on the condition(s) you define.
- It decides the direction the flow should follow—one for when the condition **passes**, another for when it **fails**.
- You can connect multiple paths and name them clearly.
- The node itself does not return output data—it simply controls **which node gets executed next**.

Configuration Details

1. Choose Condition Logic

Select whether multiple conditions should be evaluated with:

- **AND** – All conditions must be true
- **OR** – At least one condition must be true

2. Define Conditions

You can compare:

- Variable → to a fixed value
- Variable → to another variable

Example:

- `status == "approved"`
- `score >= 75`
- `userType == userGroup`

3. Define Edges (Paths)

When you connect this node to other nodes, you will define two (or more) edges:

- **Success Edge** – The path taken when the condition is met
- **Failure Edge** – The path taken when the condition is not met

You can label these edges for clarity (e.g., "True Path", "False Path", "Else Condition").

Inputs

- **Variables**: The only input this node accepts is one or more variables whose values will be checked.

Outputs

- **Flow Control Only**: This node does not return any variable or data—it only decides which connected node will execute next.

When to Use

Use the Condition node when you want your workflow to:

- Take different paths based on user input or system responses
- Check results from previous steps before continuing
- Branch logic similar to `if`, `else if`, and `else` in programming
- Run different actions based on status, type, amount, or any condition

---

Example Flow: Check Ticket Priority Before Escalation

Scenario

After a ticket is created, you want to check its priority and handle it differently based on whether it is "High" or not.

Flow Steps

1. **Create Ticket**
   Ticket details are submitted and stored in variables, including `priority`.
2. **Condition Node**
   - Condition: `priority == "High"`
   - Success Edge → **Send Escalation Email**
   - Failure Edge → **Log Ticket in System**
3. **Send Escalation Email**
   If priority is high, notify the escalation team.
4. **Log Ticket in System**
   If priority is not high, continue with the normal process.

Summary of the Flow

- The system evaluates the value of a variable (`priority`)

- Based on the result, the workflow follows a different direction
- This adds dynamic decision-making to your automation