Using the "Execute Workflow" Node in n8n for Efficient Modular Automation

---

Overview:

This SOP demonstrates how to use the "Execute Workflow" node in n8n to build modular, efficient, and reusable workflows. The key focus is toggling the `Wait for Completion` option, which can drastically cut execution time—from 36 minutes to 6 minutes—by allowing parallel processing. The guide also covers designing sub-workflows for tasks like image generation, post creation, and publishing to Facebook, ensuring each component can be reused in different automations.

Step-by-Step Guide:

1. **Create a Modular Workflow to Execute a Sub-Workflow**
    a. Open n8n and create a new workflow.
    b. Add an `Execute Workflow` node.
    c. Select the target sub-workflow you want to execute.
2. **Configure Execution Behavior**
    a. In the `Execute Workflow` node, set `Run Once for Each Item` if processing multiple items individually.
    b. Toggle `Wait for Completion` based on your needs:
        - Turn **on** if you need the result from the sub-workflow before proceeding.
        - Turn **off** for parallel execution to save time (ensure API limits are considered).
3. **Build a Sub-Workflow for Reuse**
    a. Create a sub-workflow (e.g., “Cancel Subscription” or “Image Generation”).
    b. Define inputs using the input schema (e.g., `email`, `image_description`).

c. Process the data and return results via the last node.

4. **Use Modular Workflows in Practice**
   a. In a parent workflow, use multiple `Execute Workflow` nodes to call sub-workflows (e.g., cancel subscription, generate Facebook image, publish post).
   b. Pass specific fields into each sub-workflow via the input schema.
   c. Map outputs where necessary using expressions like:

```
{{ $json["field_name"] }}
```

5. **Optimize with Conditional Execution**
   a. Use expressions and rate-limiting strategies:
      - Use `Wait` node between executions when API rate limits apply.
      - Set `Wait` time to 5–60 seconds depending on the external service.
6. **Aggregate and Use Output Data**
   a. Merge outputs from multiple sub-workflows using `Merge` node.
   b. Use the combined data (e.g., binary file + post text) in a final publishing workflow.
7. **Build and Use a Publishing Sub-Workflow**
   a. Create a new sub-workflow called `Publisher Sub`.
   b. Accept all data and use HTTP Request node to send data (e.g., to Facebook API).
   c. Send binary file (`data`) and text fields (`message`) in request parameters.

Warnings and Notes:

- **Do Not Disable** `Wait for Completion` when interacting with APIs that have strict rate limits, like Google Maps.
- **Always Define Input Schemas** in sub-workflows to avoid confusion when mapping data.
- **Monitor Execution Logs** in n8n to quickly identify and debug failed sub-workflow executions.

- **Binary Files Need Special Handling**: Ensure the downstream nodes can process the binary data passed from sub-workflows.
- **Avoid Hardcoded Delays**: Use dynamic waits or conditional execution when dealing with rate-limited APIs for better efficiency.

Visual Diagram:

```
flowchart TD
    A[Start Main Workflow] --> B[Loop Through Items]
    B --> C[Execute Workflow Node]
    C -->|Wait for Completion On| D[Process One Item Then Next]
    C -->|Wait for Completion Off| E[Process All in Parallel]
    E --> F[Multiple Executions in Parallel]
    D --> G[Collect Output]
    F --> G
    G --> H[Merge Outputs]
    H --> I[Execute Publisher Workflow]
    I --> J[Post to Facebook]
    J --> K[End]
```
