Sub-Nodes

Trigger Node

1. **Webhook Listener**
    - Listens for incoming HTTP requests from external services.
    - Can be configured with authentication and request validation.
2. **Event Listener**
    - Subscribes to internal or external event sources (e.g., message queues, database updates).
    - Supports event filtering and processing.
3. **Conditional Trigger**
    - Enables triggering workflows only when specific conditions are met.
    - Can check payload contents, headers, or metadata.

Input/Data Source Node

1. **API Connector**
    - Fetches data from external REST or GraphQL APIs.
    - Supports authentication (OAuth, API keys, etc.).
    - Handles pagination and rate limiting.
2. **Database Connector**
    - Connects to databases (SQL, NoSQL, etc.) to retrieve structured data.
    - Supports query customization and indexing for optimization.
3. **File Ingestor**
    - Reads data from CSV, JSON, or XML files from cloud storage (S3, Google Drive).
    - Supports file format transformations.
4. **Streaming Data Listener**
    - Listens to real-time data streams (Kafka, WebSockets, MQTT, etc.).
    - Buffers or batches incoming messages for processing.
5. **Caching Layer**
    - Stores recently fetched data for improved performance.
    - Avoids redundant API/database calls using TTL (Time-To-Live).

Data Mapping/Transformation Node

1. **Field Mapper**
   - Maps source fields to target fields using predefined templates.
   - Supports aliasing, renaming, and nesting/un-nesting of fields.
2. **Data Type Converter**
   - Converts data types (e.g., string to integer, timestamp to date).
   - Ensures consistency across multiple data sources.
3. **Data Formatter**
   - Formats fields (e.g., date normalization, currency conversion).
   - Applies locale-based adjustments when needed.
4. **Data Validator**
   - Ensures data integrity by checking required fields and formats.
   - Flags missing, null, or inconsistent values.
5. **Conditional Transformer**
   - Applies transformations based on conditional rules.
   - Example: If a country code is "US," format phone numbers differently.
6. **Aggregation & Calculation Node**
   - Computes derived fields (e.g., total sales, averages, percentages).
   - Supports sum, count, min/max, and other statistical functions.
7. **Custom Scripting Node**
   - Allows execution of JavaScript or Python transformation logic.
   - Useful for complex custom rules that standard transformations can't handle.

Conditional/Decision Node

1. **Expression Evaluator**
   - Parses and evaluates logical expressions (e.g., `data.score > 100`).
   - Supports multiple operators (`>`, `<`, `==`, `!=`, `&&`, `||`).
   - Can reference external variables or computed values.
2. **Pattern Matcher**
   - Uses regex or predefined patterns to classify incoming data.
   - Example: Route emails based on domain (`@gmail.com` → personal, `@company.com` → business).

Loop/Iteration Node

1. **Iterator Controller**
   - Manages loop execution by iterating over a collection.
   - Handles different iteration modes (sequential, parallel).
2. **Item Variable Mapper**
   - Maps each element in the collection to an iteration variable.
   - Example: Assigns `item` as the current element in `data.items`.

Action/Integration Node

1. **API Request Handler**
    - Sends HTTP requests (GET, POST, PUT, DELETE) to external APIs.
    - Supports authentication methods (OAuth, API Key, JWT).
2. **Payload Formatter**
    - Structures data into required JSON, XML, or FormData formats.
    - Supports dynamic field mappings and template-based transformations.
3. **Rate Limiter**
    - Controls request frequency to avoid API throttling.
    - Implements exponential backoff and request queuing.
4. **Batch Processor**
    - Groups multiple actions into a single batch request if supported by the API.
    - Example: Sending 100 WhatsApp messages in one API call.
5. **Response Validator**
    - Parses and verifies API responses to check for success/failure.
    - Supports error handling strategies like retries or fallbacks.

Output/Result Node

1. **Notification Dispatcher**
    - Sends notifications via email, SMS, Slack, or push notifications.
    - Supports templating for personalized messages.
2. **Dashboard Updater**
    - Pushes processed data to analytics dashboards or reporting tools.
    - Example: Updates charts, metrics, or KPIs dynamically.
3. **Data Storage Writer**
    - Saves final workflow results to a database, data warehouse, or file storage.
    - Supports structured (SQL, NoSQL) and unstructured (JSON, CSV) formats.

Scheduler/Timer Node

1. **Cron Scheduler**
    - Executes actions at specific time intervals (e.g., `"0 0 * * *"` for daily execution).
    - Supports complex scheduling patterns and time zones.
2. **Fixed Delay Timer**
    - Introduces a fixed delay between workflow steps.
    - Example: Waits 5 seconds before triggering the next action.
3. **Dynamic Timer**
    - Calculates delay dynamically based on workflow conditions.
    - Example: Delay depends on user input or API response times.
4. **Recurring Timer**
    - Triggers actions at repeating intervals (e.g., every 15 minutes).
    - Supports exponential backoff for retry scenarios.
5. **Event-Based Timer**
    - Waits for an event (e.g., user action, API callback) before proceeding.
    - Example: Pauses until a user approves an email confirmation.
6. **Timeout Handler**
    - Cancels workflow execution if an action exceeds a specified time limit.
    - Example: If an API response takes longer than 10 seconds, trigger a fallback.
7. **Time Zone Converter**
    - Adjusts schedules based on different time zones.
    - Example: Ensures global users receive notifications at local time.

Common Sub-Nodes for each Node:

1. **Retry & Fallback Mechanism**
    - Retries failed API requests based on predefined rules.
    - Falls back to an alternative integration in case of repeated failures.
2. **Error Handling Node**
    - Handles errors of each particular node.
3. **Cleanup Handler**
    - Cleans up any residual data.