**Manipulate PDF with Adobe developer API.txt**

**Workflow: Make OpenAI Citation for File Retrieval RAG**

**Purpose:**  
This workflow is designed to retrieve, process, and extract key citation details from PDF files. It leverages Adobe's PDF Services API to process PDF assets and uses OpenAI to extract structured citation information (e.g., invoice data, citations) from the processed content. The final output is formatted for downstream use (e.g., in Markdown or HTML) and can be appended to a data repository such as a Google Sheet.

**Key Components**

1. Trigger and Initialization

* **Manual Trigger:**
  + **Node:** *When clicking ‘Test workflow’*
  + **Function:**  
    Initiates the workflow execution manually for testing purposes.
* **Authentication for Adobe API:**
  + **Node:** *Authenticartion (get token)*
  + **Function:**  
    Retrieves an access token from Adobe’s PDF Services API using custom HTTP credentials.
  + **Configuration:**  
    Sends a POST request with form-urlencoded parameters to https://pdf-services.adobe.io/token. The access token is used in subsequent API calls.

**2. PDF Asset Handling**

* **Load Test PDF File:**
  + **Node:** *Load a test pdf file*
  + **Function:**  
    Downloads a test PDF file from Dropbox (or another storage provider) to serve as the input document.
  + **Configuration:**  
    Configured to download a file from a specified path.
* **Merge File Data and Query Parameters:**
  + **Node:** *Query + File*
  + **Function:**  
    Merges the downloaded file data with additional query parameters required for the Adobe API.
  + **Configuration:**  
    Uses the Merge node in "merge by position" mode to combine file data with query-specific information.
* **Prepare Adobe API Query:**
  + **Node:** *Adobe API Query*
  + **Function:**  
    Sets up the necessary parameters for the Adobe API call (e.g., endpoint, JSON payload). This includes specifying the transformation (e.g., extract PDF content, split pages, extract tables).
  + **Configuration:**  
    A Set node defines the target endpoint (e.g., extractpdf) and constructs a JSON payload containing parameters like renditionsToExtract and elementsToExtract.

**3. PDF Processing and File Retrieval**

* **Upload PDF Asset:**
  + **Node:** *Create Asset*
  + **Function:**  
    Uploads the PDF file to Adobe’s PDF Services API to create an asset.
  + **Configuration:**  
    Uses an HTTP POST request to https://pdf-services.adobe.io/assets with multipart-form-data containing the PDF file binary. The Adobe API token is provided in the Authorization header.
* **Process the PDF:**
  + **Node:** *Process Query*
  + **Function:**  
    Sends the merged query and file data to the Adobe API operation endpoint (e.g., extractpdf) to initiate processing.
  + **Configuration:**  
    Combines the asset ID and JSON payload and submits the request via HTTP POST.
* **Wait for Processing Completion:**
  + **Node:** *Wait 5 second*
  + **Function:**  
    Introduces a delay (e.g., 5 seconds) to allow Adobe sufficient time to process the uploaded asset.
  + **Configuration:**  
    A Wait node configured to delay execution for a fixed period.
* **Download Processed Result:**
  + **Node:** *Try to download the result*
  + **Function:**  
    Retrieves the processed result from Adobe using the URL provided in the response headers.
  + **Configuration:**  
    Uses an HTTP PUT request to the download URL with the Adobe API token for authentication.

**4. Data Extraction and AI Processing**

* **Retrieve Parsed Data:**
  + **Node:** *Get Parsed Invoice Data*
  + **Function:**  
    Fetches the result of the PDF processing from Adobe’s API, typically in a parsed format (e.g., Markdown).
  + **Configuration:**  
    Sends an HTTP GET request to an endpoint like https://api.cloud.llamaindex.ai/api/parsing/job/{job\_id}/result/markdown.
* **Regularize and Map Output:**
  + **Node:** *Regularize Output*
  + **Function:**  
    Consolidates key information (e.g., file ID, file name, extracted text) into a standardized format for further processing.
  + **Configuration:**  
    A Set node formats and prepares the data.
* **Extract Citation Data with AI:**
  + **Node:** *Apply Data Extraction Rules*
  + **Function:**  
    Uses an OpenAI assistant (via a Chain LLM node) to extract structured citation details from the parsed PDF content.
  + **Configuration:**  
    A detailed prompt instructs the AI to extract fields such as Invoice date, invoice number, supplier details, pricing, etc. The expected output is structured in JSON.
* **Optional Markdown to HTML Conversion:**
  + **Node:** *Optional Markdown to HTML*
  + **Function:**  
    Converts Markdown-formatted output into HTML if needed.
  + **Configuration:**  
    This node is disabled by default and can be enabled as per output requirements.
* **Final Output Formatting:**
  + **Node:** *Finnaly format the output*
  + **Function:**  
    Finalizes the output by substituting placeholders (e.g., citation markers) with actual file names or links, using Markdown or HTML formatting.
  + **Configuration:**  
    Uses a Code node that iterates over aggregated citation data and performs replacements to generate a final output string.

**5. Data Integration and Downstream Processing**

* **Append to Google Sheet:**
  + **Node:** *Append to Reconciliation Sheet*
  + **Function:**  
    Writes the final extracted and formatted citation data into a Google Sheet for record-keeping or further processing.
  + **Configuration:**  
    Configured with the Google Sheets node to append data to a specific document and sheet.
* **Execute Workflow Trigger (Optional):**
  + **Node:** *Execute Workflow Trigger*
  + **Function:**  
    Optionally triggers a downstream workflow, such as sending notifications or further processing the output.
  + **Configuration:**  
    Passes the final results to another workflow for additional actions.

**Data Flow Overview**

1. **Trigger & Input Acquisition:**
   * The workflow is triggered manually using the "Test workflow" node.
   * A PDF file is loaded from Dropbox and merged with query parameters.
   * Adobe API token is retrieved for authentication.
2. **PDF Processing:**
   * The PDF is uploaded to Adobe’s PDF Services API, creating an asset.
   * A processing request is sent, and the system waits for Adobe to complete the processing.
   * The processed file is then downloaded using the URL provided by Adobe.
3. **Data Extraction with AI:**
   * The parsed PDF data is retrieved from Adobe.
   * The output is regularized and mapped into a standard format.
   * An AI assistant extracts structured citation data using a detailed prompt.
   * Optionally, Markdown output can be converted to HTML.
4. **Output Delivery & Integration:**
   * The final formatted output is appended to a Google Sheet.
   * Optionally, downstream workflows are triggered for further processing or notifications.

**Customization and Configuration**

* **API Credentials:**
  + Ensure all API credentials (Adobe, OpenAI, Dropbox, Google Sheets) are properly set up and configured.
  + Adjust authentication methods and tokens as needed.
* **Prompt Customization:**
  + Modify the AI prompt in the "Apply Data Extraction Rules" node to extract the desired fields and format the output.
  + Tweak model parameters (e.g., temperature, max tokens) in the OpenAI nodes for optimal performance.
* **Output Formatting:**
  + Enable the "Optional Markdown to HTML" node if HTML output is preferred.
  + Customize the final Code node to alter how citation markers are replaced or formatted.
* **Error Handling and Wait Time:**
  + Adjust the wait time in the "Wait 5 second" node to accommodate processing times based on your workload.
  + Implement error handling in HTTP request nodes to manage transient errors.

**Troubleshooting & Best Practices**

* **Testing:**
  + Use the manual trigger to thoroughly test the workflow before deploying it in production.
  + Verify that each node (file download, upload, processing, AI extraction) produces the expected output.
* **Error Handling:**
  + Configure nodes with appropriate error handling (e.g., "continue on error") to prevent workflow interruptions.
  + Monitor API usage and response times to ensure efficient processing.
* **Monitoring and Logging:**
  + Enable logging on key nodes (e.g., HTTP requests, Code node) to capture detailed execution data.
  + Use Google Sheets or other reporting tools to monitor processed outputs.
* **Performance:**
  + For large volumes, consider increasing wait times and optimizing API calls to remain within service limits.
  + Regularly review and update the AI extraction prompts to ensure accuracy and relevancy of the extracted data.