**1. File Upload & Initial Ingestion**

**1.1 User or System Upload**

* **Supported File Types**: Images, PDFs, Excel/CSV.
* **Interface**: A drag-and-drop UI or an automated folder watch / cloud upload.
* **User Command**: For example:
  + “Apply these attributes to Brand = Astral, Catalog = Aquarius.”
  + “Extract product details from this PDF for future review.”
  + “Attach this file to all relevant products in my PIM.”

**1.2 Metadata Setup**

* The system records the **file name**, **upload date**, **user** or **automation** that triggered it, and the **command** or **intention** (bulk attribute update, attach only, etc.).
* This is saved in a “batch job” or “import session” table for reference and auditing.

**2. Parsing & Extraction**

**2.1 Determine Parsing Strategy**

* **Image** → run **OCR** (Tesseract or equivalent) to get text.
* **PDF** → if digital, parse with pdfplumber/Tabula/Camelot to extract text or tables directly; if scanned, do OCR first.
* **Excel/CSV** → read each row/column natively (e.g., Python’s pandas).

**2.2 Chunking & Table Detection (For PDFs)**

* Break large PDFs into sections or “chunks”:
  + Each chunk might be one table or a relevant block of text.
  + Store page numbers, headings, or subheadings that appear near that chunk.

**2.3 Basic Cleaning & Formatting**

* **Remove noise**: repeated headers, footers, page numbers.
* **Coalesce** multiline cells if table lines wrap.
* **Convert** extracted text into a raw CSV or JSON structure, e.g.:

json

CopyEdit

[

{

"table\_id": "table\_1\_page\_3",

"heading": "FlowGuard CPVC",

"columns": ["Property", "Value (PSI)", "Value (N/mm²)", ...],

"rows": [

["Tensile Strength @ 23°C", "8000", "55", ...],

...

]

},

...

]

**3. Data Labeling & Product Context Assignment**

*(Key for dealing with multiple tables that share the same headers.)*

**3.1 Labeling Tool or AI Classifier**

* **Label Studio**, **Doccano**, or custom tool to show each “chunk” or “table” with its surrounding text snippet (heading/caption).
* **Goal**: Identify or confirm the **product line** (e.g., “Astral Aquarius,” “UPVC 15–50mm range,” “Hot Rolled Coil,” etc.) each table belongs to.
  + If each row references a distinct product, label them row by row.

**3.2 Assign Standardized Column Names**

* The labeling process or an AI model also normalizes the columns:
  + “Tensile Str.” or “TS @ 23C” → “tensile\_strength\_23c”
  + “Thickness (mm)” → “thickness\_mm”
* If you see synonyms or slightly different column names across tables, the labeling ensures they unify under the same canonical name.

**3.3 Export of Labeled Data**

* After labeling, we produce a structure where each row is:

json

CopyEdit

{

"identified\_product": "UPVC brand 'X' 15mm pipe",

"attribute\_key": "tensile\_strength\_23c",

"value\_english": "8000 psi",

"value\_si": "55 N/mm²",

"test\_method": "ASTM D638"

}

* If the labeling is done by an AI classifier, a **confidence** score is added. Low-confidence items can be reviewed manually.

**4. Handling Unknown Attributes & Brand/Catalog Scopes**

**4.1 Compare Extracted Attributes to PIM**

* For each labeled “attribute\_key,” check your existing PIM fields:
  + If it exists (like “tensile\_strength\_23c”), plan to update it.
  + If it’s new (e.g., “impact\_strength\_23c”), mark as **unknown**.

**4.2 Brand/Catalog Scope**

* If the user command says: “Apply to brand Astral, catalog Aquarius,” the system fetches the list of product IDs or SKUs that match.
* If each row is specifically labeled for a single product, that overrides the brand+catalog approach. Otherwise, by default, everything is applied to the entire brand’s product set.

**4.3 Data Governance Steps for New Attributes**

* **Manual Approval**: If “impact\_strength\_23c” doesn’t exist, show it on a “new attribute creation” queue. An admin can confirm the name, data type, etc.
* **Auto-Create**: If your system is set to auto-create attributes, it does so with a standard naming or a user-specified name.

**5. Review & Validation UI**

**5.1 Summary Preview**

* The user sees how many **products** will be updated, how many **attributes** are recognized, how many are “new,” etc.
* If the parse is from an **Excel** file, columns might map easily to existing PIM fields. If from a PDF, labeling might require more review.

**5.2 Edit or Confirm**

* The user can rename or skip certain attributes:
  + For instance, rename “impact\_strength\_23c” → “charpy\_impact\_23c.”
* The user can pick whether to create new attributes or discard them if they’re irrelevant.

**5.3 Confidence Handling**

* Items flagged as “low confidence” (maybe the LLM was uncertain) are displayed with a highlight. The user must confirm or correct them.

**6. Bulk Update & File Attachment in PIM**

**6.1 Attach File to Products (if commanded)**

* If user said: “Attach this PDF to all Astral/Aquarius products,” the system:
  + Creates or references a digital asset record in the PIM for that PDF.
  + **Links** it to each relevant product ID (brand + catalog or an explicitly chosen subset).

**6.2 Bulk Attribute Update**

* For each extracted row or each attribute-value pair, the system:
  1. Determines the **target product** (by brand scope or specific labeling).
  2. Locates or creates the PIM attribute.
  3. **Updates** the product with the new value. Possibly store English + SI in separate fields or appended data.

**6.3 Logging & Completion**

* After the updates, the system logs:
  + How many **products** got updated.
  + How many **attributes** changed.
  + How many new attributes got created.
  + Any errors or warnings (e.g., product not found, attribute creation not allowed).

**7. Post-Processing & Governance**

**7.1 Final Confirmation**

* The user sees a final success message: “Applied 10 new mechanical properties to 50 products; attached 1 PDF to 50 products. Created 2 new attributes in the PIM.”

**7.2 Auditing & Version Control**

* The system stores the **batch job** details: who triggered it, the file used, labeling details, new attributes created.
* If needed, you can revert or compare “before vs. after.”

**7.3 AI Model Refinement**

* If an AI model is involved in parsing or classification, the corrections from the labeling or final user edits can be used to retrain or fine-tune the model for better performance next time.

**Putting It All Together: Example Step-by-Step**

1. **User Uploads** “Astral\_Aquarius\_Brochure.pdf” & commands “Apply these attributes to brand Astral, catalog Aquarius.”
2. **System Parses** the PDF:
   * Finds multiple tables with repeated headers (Thickness, Tensile Strength, etc.).
   * Splits them into separate table chunks.
3. **Labeling**:
   * A user or an AI model with a labeling UI assigns each chunk to a product line. For example: Table A → “UPVC 15mm – 50mm,” Table B → “CPVC 20mm – 50mm.”
   * They unify column synonyms into standard attribute keys.
4. **Check PIM Attributes**:
   * “Tensile Strength @ 23°C” is recognized or created.
   * “Impact Strength (Notched)” is new, so the user is asked to confirm or rename.
5. **Review & Validate**:
   * The system shows a preview: “We found 8 attributes for 30 products. 3 new attributes are proposed.”
   * The user checks or edits them.
6. **Bulk Update**:
   * The system calls the PIM’s API (or import) to set these attributes on each product within brand Astral/Aquarius.
   * If the user also said “Attach the PDF,” the system links the PDF to all 30 products.
7. **Completion**:
   * The pipeline logs “Updated 30 products, created 3 new attributes, attached 1 PDF,” with no errors reported.

This final concept covers **multiple file types** (through OCR or direct parsing), **data labeling** for correct product assignment, **unknown attribute handling**, a robust **review** process, and the subsequent **bulk PIM update** (and file attachment). It ensures a **scalable**, **accurate** method to keep your PIM data fresh—whether you’re dealing with new mechanical property tables or simply adding marketing PDFs to your product records.

**Extraction Mechanisms**  
Below is a **concise but comprehensive** explanation of **how extraction works** from **catalogs** (e.g., product brochures, specification sheets) to ultimately **structure** that data for your PIM. This includes the **basic logic**, the **optional use of AI**, how we **identify and extract** relevant data fields, and how everything ties together.

**1. Catalog Input & Preprocessing**

**1.1 Ingesting the Catalog**

* **Files**: Typically PDF, Word, Excel, or even images.
* **Sources**: These might come from a marketing team, a vendor, or an in-house technical department.

**1.2 Determining File Type & Strategy**

* **Digital PDF**: If the text is selectable, we can parse tables or text directly.
* **Scanned PDF or Images**: We must run OCR (e.g., Tesseract) to convert images to text.
* **Excel**: We can read rows and columns natively (via Python’s pandas, for example).

**2. Table Detection & Text Chunking**

**2.1 Table Detection**

* Tools like **Camelot**, **Tabula**, or **pdfplumber** can find the bounding boxes of tables in a PDF’s layout.
* If the file is **image-based**, we might first do an OCR layout analysis (like **Tesseract** with hOCR output) or use specialized “table detection” neural networks (like **LayoutParser**).

**2.2 Splitting into Chunks**

* Each table is extracted into a **structured** format (rows/columns).
* Non-tabular text (headings, paragraphs) can also be chunked out. For instance, we might want the product name from a heading if the table itself only has numeric data.

**3. Data Extraction Logic**

**3.1 Headings & Context**

* We gather **surrounding** lines near each table to determine **product context**. For example:
  + “FlowGuard CPVC piping” might appear right before a table of specs.
  + We store that “FlowGuard CPVC piping” is the **context** for this table.

**3.2 Column Headers**

* We read **header cells** in each table row to identify attributes (like “Thickness,” “Tensile Strength,” “Elongation,” etc.).
* If repeated or multi-line headers exist, we unify them into a single, clean header.

**3.3 Row Values**

* Each subsequent row is **key-value** pairs:
  + “Thickness = 1.4–25.4 mm”
  + “Yield Strength = 270–460 MPa”
  + “Tensile Strength = 400–550 MPa”
* If the table is for a single product family, all rows relate to that context. If each row is for a different SKU, we might parse a “Product Name” column.

**4. Handling Multiple Tables & Repetitive Headers**

**4.1 Merge or Distinguish**

* If there are **multiple** tables with similar headers (e.g., “Thickness,” “Yield Strength”), we note the different contexts:
  + Table A: “UPVC pipes”
  + Table B: “CPVC pipes”
* This ensures we **separate** the attributes or apply them to the **right product** line in the PIM.

**4.2 Data Labeling / Manual Tagging**

* If it’s ambiguous which product line a table is for, a **labeling** or **review** step can confirm.
* That labeling ensures extracted data is assigned to the correct product (or brand) in the PIM.

**5. Optional AI / NLP Steps**

**5.1 Named Entity Recognition**

* A custom or **Generative AI** model can detect references to product names, attributes, or test methods in the text.
* E.g., “Tensile Strength @ 23°C” is recognized as an attribute, “ASTM D638” recognized as a test standard.

**5.2 Confidence Scores & Human Review**

* AI might produce a **confidence** measure for each extracted attribute.
* Low confidence or suspicious values get routed to a **human** check.

**5.3 Synonym & Standardization**

* If the table says “Tens. Str.” in one place, “Ultimate Tensile Strength” in another, AI can unify them under one canonical key.

**6. Data Formatting & Normalization**

**6.1 Numeric & Unit Parsing**

* E.g., “55 N/mm²” might be stored as (value=55, unit='N/mm²').
* If the catalog has English/Imperial units, we might store “8000 psi” or convert to MPa as needed.

**6.2 Handling Multiple Columns**

* Some catalogs list both English and SI units. We might store each in separate fields or unify them.

**6.3 Staging in a Structured Format**

* Typically, we produce a JSON or CSV with rows like:

json

CopyEdit

{

"product\_family": "CPVC Pipe",

"attribute": "Tensile Strength",

"value": "4000 psi",

"unit": "psi"

}

**7. Linking to the PIM**

**7.1 Product Identification**

* If the table references “CPVC 15–50 mm,” we match it to one or more **SKUs** in the PIM (e.g., brand = “Astral,” line = “CPVC,” diameter range = 15–50 mm).
* Alternatively, if the row mentions a specific product code (like “SKU #12345”), we do a direct match.

**7.2 Existing vs. New Attributes**

* We see if “Tensile Strength @ 23°C” is already defined in the PIM.
* If not, we either auto-create or prompt an admin to confirm the new attribute name.

**7.3 Bulk Update**

* Once we have the final mapping of (PIM product, PIM attribute, new value), we run a **bulk update**.
* For example:

python

CopyEdit

for row in extracted\_data:

product\_id = find\_or\_guess\_pim\_product(row["product\_family"])

attribute\_key = unify\_attribute\_name(row["attribute"])

pim.update\_attribute(product\_id, attribute\_key, row["value"], row["unit"])

**8. Review & Confirmation**

**8.1 Summary Page**

* The user sees how many products are updated, how many attributes are recognized, and if any new attributes are created.

**8.2 Manual Corrections**

* If the system is uncertain or if user sees a mismatch, they can override or fix the attribute or product mapping.

**9. Attachments (Optional)**

* If the user wants the **PDF** or **Excel** file itself attached to those product records, the system:
  + Creates or references a digital asset in the PIM’s **Documents/Assets** section.
  + Links that asset to all relevant products.

**Putting It All Together: Example**

1. **User** uploads “UPVC\_Catalog.pdf.”
2. **System** sees it’s a PDF → uses **pdfplumber** to detect 3 tables.
3. Each table is extracted to CSV format. Surrounding headings say “UPVC 20–50 mm,” “UPVC 63–110 mm,” etc.
4. **AI** or labeling confirms: Table 1 → “UPVC small diameters,” Table 2 → “UPVC larger diameters.”
5. Each table row: “Thickness,” “Max Pressure,” “Tensile Strength.” The system merges synonyms.
6. The user’s command: “Apply these new properties to brand Astral, catalog Aquarius.” The system finds relevant product IDs.
7. The system sees “Max Pressure” is new → prompts to create the attribute.
8. **Review**: The user sees a list: “We’ll update 20 products. 1 new attribute. 3 existing attributes.” They confirm.
9. **Bulk Update**: The pipeline calls your PIM’s API or import process, sets each attribute value.
10. A final message: “Catalog processed. 20 products updated, 1 new attribute created, no errors.”

**Conclusion**

**Extraction from catalogs** typically involves:

1. **Parsing** text/tables from PDF, Excel, or images (via OCR).
2. **Identifying** or labeling which product each table or row pertains to.
3. **Mapping** the extracted attribute names/values into a **consistent** data structure.
4. **Updating** your PIM with these attributes (creating new ones if needed).
5. **Optionally** attaching the source file to the relevant product(s).

This approach yields a **scalable, repeatable** pipeline for managing **specifications** or **technical data** from a wide range of product catalogs or brochures.