

Module 03 — Structured Documents: Practical Exercise Steps

Written step-by-step reference for the Exercise 1 walkthrough videos (Form Extractor & RegEx Based Extractor). UiPath IXP Self-Learning Course.

Part 1 — Form Extractor: Form 1040X

Exercise 1 – Part 1: UiPath Extraction Process Setup (Till Classification step)

Below are the step-by-step instructions to build the workflow in UiPath Studio, matching the video walkthrough.

Step 1: Install the Required Package

- Open UiPath Studio and create a new Blank Project named **“Extraction Process”**.
- Click **Manage Packages** in the top toolbar.
- Locate `UiPath.IntelligentOCR.Activities` and install it.

Step 2: Create the Data Folder Structure

- Open File Explorer and navigate to your project directory: `Documents > UiPath > Extraction Process`.
- Inside the project folder, create a new folder named **Data**.
- Inside the Data folder, create two subfolders: **Inputs** and **Outputs**.
- Place all input documents (PDFs to be processed) into the Inputs folder.

Step 3: Configure the Taxonomy Using Taxonomy Manager

- In UiPath Studio, click **Taxonomy Manager** in the top toolbar.
- Create a new **Group** and name it **IRS**.
- Under the IRS group, add a **Category** named **Forms**.
- Under the Forms category, add a **Document Type** named **Form1040X**.
- Under Form1040X, add the following Fields:
 - `FirstName` — type: Text
 - `LastName` — type: Text
 - `SocialSecurityNumber` — type: Text
 - `Address Line 1` — type: Text
 - `Apartment Number` — type: Text
 - `City Town Zip` — type: Text
 - `Return_2019, Return_2018, Return_2017, Return_2016` — type: Boolean (one entry per year)
 - `Dependents` — type: Table. Inside the Dependents table field, add columns:
 - `F_Name` (type: Text)

- SSN_Number (type: Text)
 - Signature — type: Boolean
 - Date — type: Date
- Create a second Document Type named **VehicleRegistrationForm** under a new Group called **Automobiles**, Category **Forms**. Add the following fields:
 - Registration-number — type: Text
 - Inspector-name — type: Name
 - Vehicle-license-number — type: Text
 - Driving-license-number — type: Text
 - Cost-of-Damage — type: Number

Step 4: Set Up the Main Sequence in Main.xaml

- In UiPath Studio, open **Main.xaml**.
- Add a **Multiple Assign** activity as the first activity in the Main Sequence.
- Configure the following variable assignments:
 - InputsFolder = ".\Data\Inputs"
 - OutputsFolder = ".\Data\Outputs"
 - AllFiles = Directory.GetFiles(InputsFolder)
- In the Variables panel (Data Manager tab), confirm the following variables are created:
 - AllFiles — type: String[], scope: Main
 - OutputsFolder — type: String, scope: Main
 - InputsFolder — type: String, scope: Main

Step 5: Add the Load Taxonomy Activity

- Below the Multiple Assign, add a **Taxonomy** container (scope activity).
- Inside the Taxonomy scope, add a **Log Message** activity with message: "*** Load Taxonomy **", Log Level: Info.
- Below the Log Message, add a **Load Taxonomy** activity.
 - In the Output field, assign the result to a new variable: DocTaxonomy (type: DocumentTaxonomy, scope: Main).

Step 6: Add a For Each Loop to Iterate Over All Files

- Below the Taxonomy scope, add a **For Each** activity.
 - Set **In** to AllFiles.
 - In the Properties panel, set Argument type to **String**.
- Inside the For Each Body, add a **Log Message** with the message expression: "Processing File : " + currentText, Log Level: Info.

Step 7: Add the Digitize Document Activity

- Inside the For Each Body, after the Log Message, add a **Digitize Document** activity (from Activities panel under App Integration > Document Understanding).

- Set **Document Path** to `currentText`.
- Set **OCR Engine** to **UiPath Document OCR**.
- In Output, assign:
 - Document Text → new variable `DocText` (type: String, scope: For Each `currentText`)
 - Document Object Model → new variable `DocDOM` (type: Document, scope: For Each `currentText`)

Step 8: Add the Classify Document Scope

- After the Digitize Document activity, add a **Log Message** with message `*** Classification ***`.
- Then add a **Classify Document Scope** activity.
 - Set **Document Path** to `currentText`.
 - Set **Document Text** to `DocText`.
 - Set **Document Object Model** to `DocDOM`.
 - Set **Taxonomy** to `DocTaxonomy`.
 - In Output, assign Classification Results to a new variable `DocClassificationResults` (type: `ClassificationResult[]`, scope: For Each `currentText`).
- Inside the Classify Document Scope, drop in an **Intelligent Keyword Classifier** from the Classifiers section of the Activities panel.

Step 9: Configure the Intelligent Keyword Classifier – Manage Learning

- Click on the **Intelligent Keyword Classifier** activity in the canvas.
 - In Properties, set **LearningFilePath**: point to a file in the DocumentProcessing folder (e.g., `DocumentProcessing\LearningFilePath`) — save this file via the file browser dialog.
- **Note:** Create a blank new file `LearningFilePath.json` in the DocumentProcessing folder.
- Click **Manage Learning** inside the activity.
 - In the Manage Learning dialog, you will see Document Types: **Form1040X** and **VehicleRegistrationForm**, each showing “No training for this document type.”
- For **Form1040X**, click **Start Training...**:
 - Select 3–4 PDF files from `Data\Inputs`.
 - OCR Engine is pre-populated: UiPath Document OCR, with the endpoint and API key already set.
 - Click **Start Training** and wait for the digitization and training progress bar to complete (100%).
- For **VehicleRegistrationForm**, click **Start Training...**:
 - Select 3–4 PDF files from `Data\Inputs`.
 - Click **Start Training** and wait for completion.
- After training completes, both document types will show their trained word vectors in the Manage Learning dialog.
- Click **Save** and close the Manage Learning dialog.

Step 10: Add the Present Classification Station

- After the Classify Document Scope, add a **Log Message** with message `*** Classification Validation ***`.
- Add a **Present Classification Station** activity.

- Set **Document Path** to currentText.
- Set **Document Text** to DocText.
- Set **Document Object Model** to DocDOM.
- Set **Taxonomy** to DocTaxonomy.
- Set **Automatic Classification Results** to DocClassificationResults.
- In Output, assign Validated Classification Results to a new variable DocClassificationResults_H (type: ClassificationResult[]).

Step 11: Add the Train Classifiers Scope

- After the Present Classification Station, add a **Log Message** with message "*** Classification Trainer ***".
- Add a **Train Classifiers Scope** activity.
 - Set **Document Path** to currentText.
 - Set **Document Text** to DocText.
 - Set **Document Object Model** to DocDOM.
 - Set **Taxonomy** to DocTaxonomy.
 - Set **Human Validated Classification Data** to DocClassificationResults_H.
- Inside the Train Classifiers Scope, drop in the **Intelligent Keyword Classifier Trainer** activity.
 - Set **Learning File Path** to "DocumentProcessing\..." (same path used in Step 9).
 - Click **Configure Classifiers** to verify the classifier is bound correctly.

Step 12: Run and Validate the Process

- Click **Debug** (or Run) in the top toolbar to execute the automation.
- Monitor the Output panel to verify the process logs:
 - ** Load Taxonomy **
 - Processing File : .\Data\Inputs\<filename>
 - ** Classification **
 - ** Classification Validation **
 - ** Classification Trainer **
- Confirm that each file in the Inputs folder is processed through Digitize → Classify → Validate → Train in sequence.
- The automation processes all files in AllFiles (Form1040X PDFs and VehicleDamageReport PDFs) iteratively, training the classifier incrementally with human-validated results.

Part 2 — RegEx Based Extractor: Vehicle Damage Report

Exercise 1 – Part 2: UiPath Extraction Process – Data Extraction, Validation & Export

Continued from previous exercise... Below are the step-by-step instructions to build the workflow in UiPath Studio.

Step 1: Add a For Each Loop for Classification Results

- After the Train Classifiers Scope activity in Main.xaml, add a **For Each** activity.
 - Set **In** to DocClassificationResults_H.
 - Rename the iterator variable to currentClassificationResult.
 - In the Properties panel, set Item name to currentClassificationResult and Argument type to UiPath.DocumentProcessing.Contracts.Results.ClassificationResult.
- In the Variables panel, confirm a new variable Counter is created (type: Int32, scope: For Each currentText).

Step 2: Add the Data Extraction Scope

- Inside the For Each currentClassificationResult Body, add a **Log Message** with the message "*** Extract Data from Document **", Log Level: Info.
- Below it, add a **Data Extraction Scope** activity (from Activities > App Integration > Document Understanding).
 - Configure the following inputs in the Properties panel:
 - **ClassificationResult:** currentClassificationResult
 - **DocumentObjectModel:** DocDOM
 - **DocumentPath:** currentText
 - **DocumentText:** DocText
 - **Taxonomy:** DocTaxonomy
 - In Output, assign Extraction Results to a new variable DocExtractionResults (type: ExtractionResult, scope: For Each currentClassificationResult).
- Inside the Data Extraction Scope, drop in a **Form Extractor** activity (from Activities > Extractors).

Step 3: Configure the Form Extractor Template (Template Manager)

- Click on the **Form Extractor** activity, then click **Manage Templates** (or open Template Manager from the activity).
- In the Template Manager dialog, click **Create Template**.
- In the Create a new template dialog:
 - Set **Document Type** to **Form1040X**.
 - Set **Template name** to **Template_1040X**.
 - Select one file from Data\Inputs folder (preferably a native document).
 - Set **OCR Engine** to **UiPath Document OCR**.
 - Pre-fill Endpoint, Api Key, and Timeout fields with the same credentials used previously.
 - Leave **Apply OCR on PDF** as **Auto**.
 - Click **Configure** — the system digitizes the document and opens the Template Editor.
- In the Template Editor, the document opens showing the Form 1040-X (two pages). Map the fields as described in the video.
- After all fields are mapped, click **Submit** to save the template.

Step 4: Configure the Regex Based Extractor

- In the Data Extraction Scope, also drop in a **Regex Based Extractor** activity (from Activities > Extractors).

- Click **Configure Regular Expressions** inside the activity.

Field	Regex
Registration Number	registration\s+number:\D*(\d{2})(-)(\d{7})
Inspector Name	report\s+prepared\s+by:\s*(.+?)\s*designation
Vehicle License Number	vehicle\s+license\s+number:\s*(.+?)\s*drive
Driving License Number	driving\s+license\s+number:\s*(?<dlicense>.+?)\s*brief nature
Cost Of Damage	cost\s+of\s+damage.+?(\d+)

- Click **Save** in the Configure Regular Expressions dialog.

Step 5: Configure Extractors via Configure Extractors Dialog

- Inside the Data Extraction Scope, click **Configure Extractors**.
- In the Configure Extractors dialog:
 - For **Form1040X**: check the **Form Extractor** checkbox (blue check) to activate it.
 - For **VehicleRegistrationForm**: leave Form Extractor unchecked; check **Regex Based Extractor**.
- Click **Save**.

Step 6: Add the Present Validation Station

- After the Data Extraction Scope, add a **Log Message** with the message "*** Validate Extracted Data from Document ***", Log Level: Info.
- Below it, add a **Present Validation Station** activity.
 - Configure inputs:
 - **Document Path**: currentText
 - **Document Text**: DocText
 - **Document Object Model**: DocDOM
 - **Taxonomy**: DocTaxonomy
 - **Automatic Extraction Results**: DocExtractionResults
 - In Output, assign Validated Extraction Results to a new variable DocExtractionsResults_H (type: ExtractionResult, scope: For Each currentClassificationResult).

Step 7: Build the Results File Path

- After the Present Validation Station, add a **Sequence** container named **Get ResultsFilePath**.
- Inside the sequence, add a **Log Message** with the message "*** Create Path for Results File ***", Log Level: Info.
- Below it, add a **Multiple Assign** activity with the following assignments:
 - First row: `FileName = Path.GetFileNameWithoutExtension(currentText)` — create a new variable FileName (type: String, scope: For Each currentClassificationResult).

- Second row: `ResultsFilePath = Path.Combine(OutputsFolder, FileName + "-" + Counter.ToString("000") + ".xlsx")` — create a new variable `ResultsFilePath` (type: String).

Step 8: Add Export Extraction Results and Output Loop

- After the `Get ResultsFilePath` sequence, add a **Log Message** with the message `*** Extract Data/Outputs ***`, Log Level: Info.
- Below it, add an **Export Extraction Results** activity.
 - Set **Extraction Results** input to `DocExtractionsResults_H`.
 - In Output, assign `DataSet` to a new variable `DocExtractionResultsDS` (type: `DataSet`).
- Below it, add a **For Each** activity named **For Each currentDataTable**:
 - Set **In** to `DocExtractionResultsDS.Tables`.
 - Rename iterator to `currentDataTable`.
 - Add a **Write Range Workbook** activity:
 - `FilePath = ResultsFilePath`
 - `Table Name = currentDataTable.TableName`
 - `Table = currentDataTable`

Step 9: Save and Verify the Final Workflow

- Press **Ctrl+S** to save `Main.xaml`.
- The overall Main Sequence flow is now: `Multiple Assign → Taxonomy (Load Taxonomy) → For Each currentText [Digitize → Classify → Present Classification Station → Train Classifiers → For Each currentClassificationResult [Extract → Validate → Build Path → Export]]`.
- Click **Debug** to run the full process and confirm all steps execute without errors and the output is stored in an Excel file in the `Outputs` folder.