Document Understanding Process

Studio Template

Revision History

Date	Changes
Jun-2021	V1.0
Oct-2021	V2021.10
Feb-2022	V2022.2-preview
May-2022	V2022.4
June-2022	V2022.6-preview

Release Notes:

2022.6-preview:

New features:

- Added C# support
- Introduced Asset naming convention for better visibility of Document Understanding Assets in Orchestrator.
- Added new overriding Assets for Orchestrator Configuration.

Changes:

- Updated all dependencies to the latest stable version.

Bugfixes:

- Fixed error in reading of the Config file in **Main-Attended.xaml.**
- Fixed the Log Message from the **Classify** workflow in C#. It now displays the correct classification results.

2022.4:

New features:

- Added .NET 5.0 support

Changes:

- Removed *Windows – Legacy* support
- Updated all dependencies to the latest stable versions
- Rescoped the *extractionResults* variable in all **Main** workflows to prevent misuse

Bugfixes:

- Corrected small typos

2022.2-preview:

Changes:

- Updated dependencies for Document Understanding activities to 22.2-preview release
- **Framework\ReusableWorkflows\SetTransactionProgress.xaml** - Retry Scope within was updated with ContinueOnError = True
- **Framework\40_TrainClassifiers.xaml** – Invoke UnlockFile activity within was updated with ContinueOnError = True

Bugfixes:

- Fixed the page range logging on document exceptions in **Main-Attended.xaml**

2021.10:

New features:

- New workflows for advanced Business Rule validation for Classification and Extraction.
- New workflow for post processing Invoices based on the OOTB (out of the box) model.
- Validation Station/Action now only displays the relevant pages instead of the whole file. Removed the PDF splitting functionality.
- Support for ML Extractor Trainer to use the auto-retrain feature in AI Center.
- New assets / settings in the config file for skipping classification/extraction training.
- New assets / settings in the config file to force documents through manual validation (useful in development/UAT especially)
- New exception type: **DocumentRejectedByUserException**. This exception is thrown when a user explicitly rejects a document during validation. When setting the transaction status, a DocumentRejectedByUserException is treated the same way as a BusinessRuleException.

Changes:

- Exception handling workflows now receive the Exception as an argument instead of an exception message
- Improved logic for loading Orchestrator assets
- Improved exception messages
- Improved comments and annotations
- Improved logging

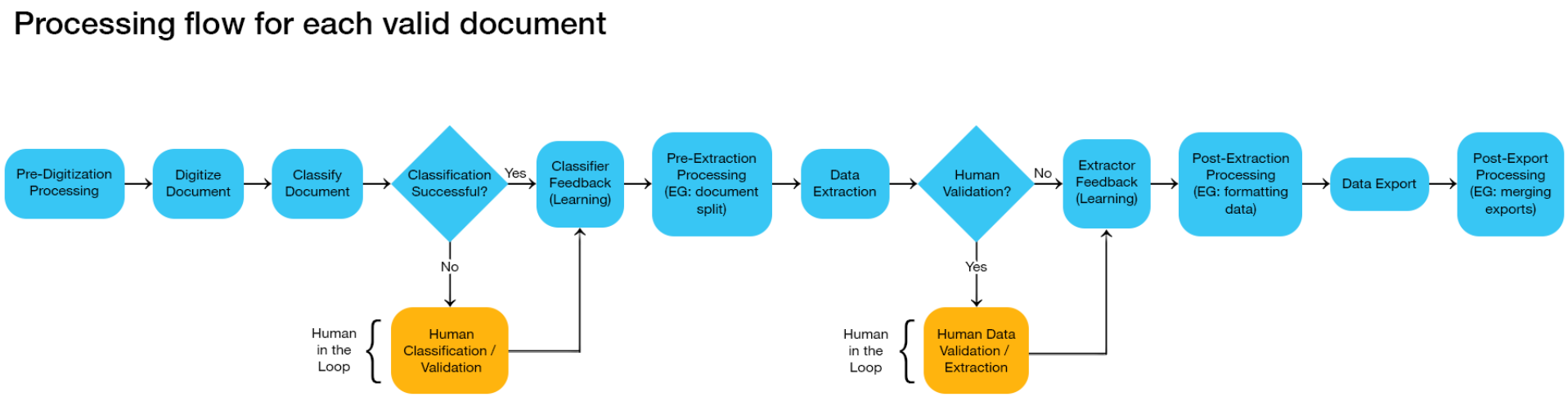
Overview

As the number of Document Understanding engagements keeps increasing in complexity, the need for a common implementation/RPA approach becomes evident. Document Understanding processes have a particular logical flow and requirements that are better suited for a dedicated approach of their own, instead of relying on the existing RE-Framework.

Key Features

- Seamless getting started process with new Document Understanding projects.
- Suitable for all use cases: from quick demos to scalable implementation projects
- Production-ready. Implements built-in logging, exception handling, and retry mechanisms.
- Common architecture for both Attended and Unattended (plus Action Center) implementations. Simple to switch between solutions.
- 2021.10: Ability to post-process Invoices to minimize the number of hits to Action Center
- Designed to make development, testing, deployment, debugging, and scaling easy.
- Follows the best practices pertaining to RPA, Document Understanding, Orchestration Processes, and Long-running workflows.

Generic Document Processing Flow



Keep in mind that the diagram above shows that the most detailed logical flow split into the smallest possible modules. In practice, it is to be expected that some parts could be merged or might be completely excluded, as they are not required in a particular implementation.

Pre-Digitization Processing

Any processing that needs to take place before digitizing the document. E.g., applying grayscale or a skew correction.

Digitize Document

A digital version of the document is obtained. In case the document is scanned, OCR (Optical Character Recognition) is required.

Classify Document

The document is classified.

Classification Successful?

The logic required to determine whether the classification was successful or not. Based on this decision, classification validation might be needed.

Note: Deciding whether human validation is needed or not is a business decision.

Human Classification / Validation

If the automatic classification was unsuccessful (low confidence, business rules are not met, etc.), send the document to a human operator for manual classification/validation or rejection.

Classifier Feedback (Learning)

Integration of the learning mechanisms, if any, for classifier(s). It is recommended to only train with human-validated data, but not mandatory.

Pre-Extraction Processing

Any processing that needs to take place before the actual data extraction begins.

Data Extraction

Extract the data using the appropriate extractor(s).

Human Validation?

The logic required to determine whether the human validation of the extracted data (validating the extracted data against pre-defined business rules, checking confidence levels, OCR confidence, missing data, etc.) is needed or not.

Note: Deciding whether human validation is needed or not is a business decision.

Human Data Validation / Extraction

Human-in-the-loop step where the human operator can correct automatically extracted data or manually extract the missing data.

Extractor Feedback (Learning)

Integration of the learning mechanisms, if any, of the used extractor(s). It is recommended to only train with human-validated data, but it is not mandatory.

Post-Extraction Processing

Any processing that should be done to the extracted data before exporting it. E.g., formatting the data in a specific manner.

Data Export

Exporting the data to a persistent location so that it can be used by other processes or by business users. Common examples: exporting to the Data Service, updating a Database, or writing the results to an Excel, CSV, or JSON file.

Post-Export Processing

Any processing that should be done after the data is exported. E.g., merging data extracted from multiple documents into a single result or sending a notification email.

Solution Architecture

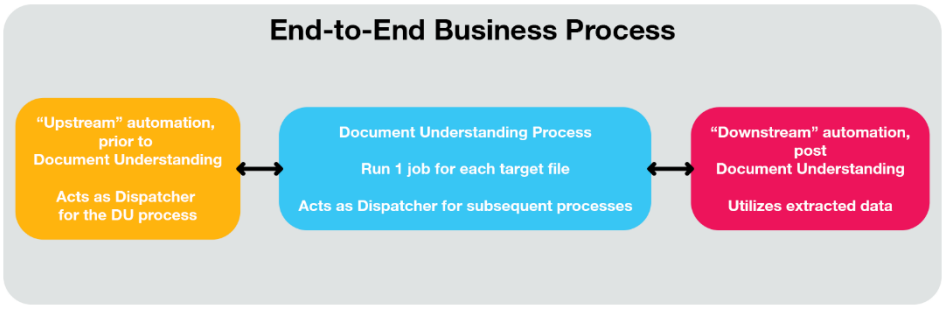
Document Understanding, Part of End-to-End Business Processes

Document Understanding processes are not standalone. They are part of bigger business processes to be automated.

Benefits:

- Prevents external issues from impacting Document Understanding processes and from causing unneeded consumption of license pages through re-execution.
- Better overview of workload and robot utilization.
- Easier to scale.

The architecture for an end-to-end Business Process involving Document Understanding consists of:



1. **Upstream automation** which handles all the business logic that should be executed prior. It consists of one or several RPA processes and acts as a Dispatcher for Document Understanding jobs.
2. **Document Understanding process** - acts as a Performer for "Upstream" and as a Dispatcher for "Downstream" jobs.
3. **Downstream automation** which handles all the business logic that makes use of the extracted data. It consists of one or several RPA processes and acts as a Performer for Document Understanding jobs.

The One-Job-Per-File Approach

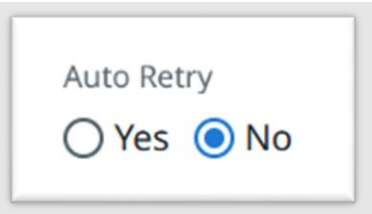
Document Understanding processes will not run as batch jobs. Instead, an individual job starts for each file to be processed. This approach is used for both Attended and Unattended implementations.

Document Understanding and Queues

Orchestrator v20.10.8 or Newer

Orchestrator version 20.10.8 introduced queue support for Persistence activities enabling queues to be used.

Note: To avoid Document Understanding license consumption through re-executions, the **Auto Retry** functionality of the queue should be **disabled**.



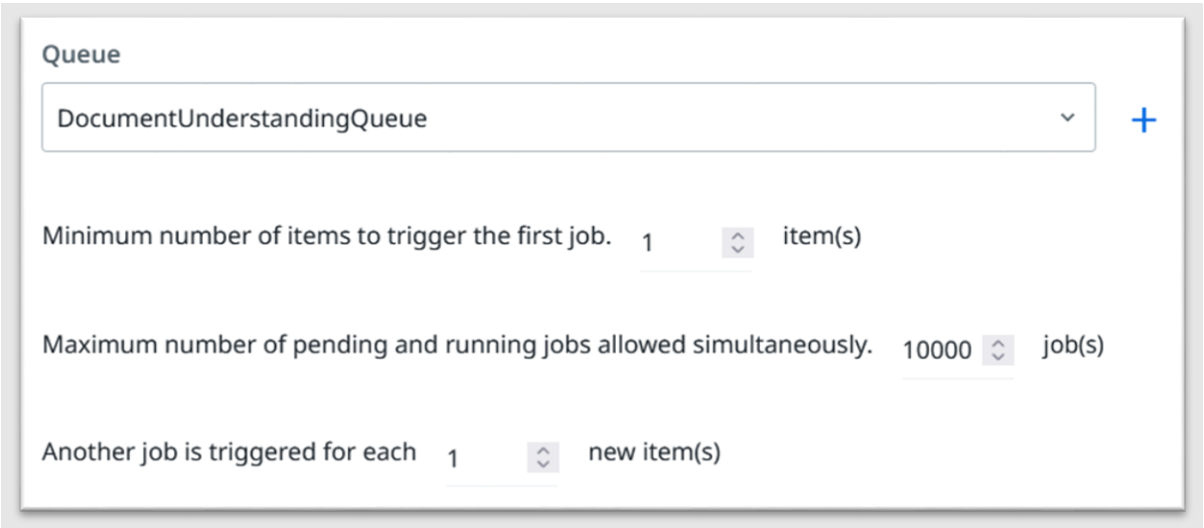
Orchestrator Versions Prior to 20.10.8

Queues should not be used for Document Understanding processes, as they are not supported. Items waiting for Human Validation for more than 24h will be marked as **Abandoned**.

Starting Document Understanding Jobs

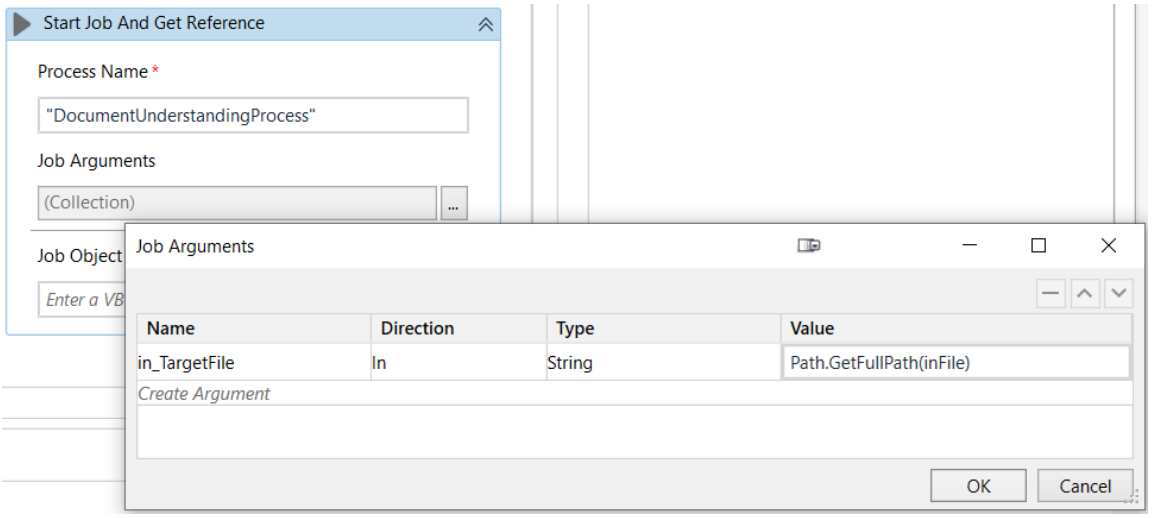
A dedicated Dispatcher mechanism is required to start Document Understanding jobs in Unattended scenarios. There are two recommended approaches:

1. A Queue Trigger, tasked to start a new job for every new Queue item, using the settings stated in the image below



Note: Due to the current Job Count Strategy for triggers, some queue items experience delays before a processing job is triggered. This behavior is on the roadmap to be addressed.

2. Having the Dispatcher process use a Start Job activity



Note: Currently, only the **Start Job And Get Reference** activity has support for passing job arguments. This activity is part of the **UiPath.Persistence.Activities** package.

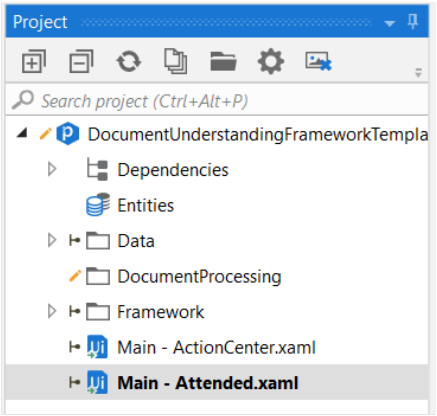
Studio Project Overview

The Document Understanding Process is available as a UiPath Studio project template and the projects created using it automatically include all files.

Settings for Attended Processes

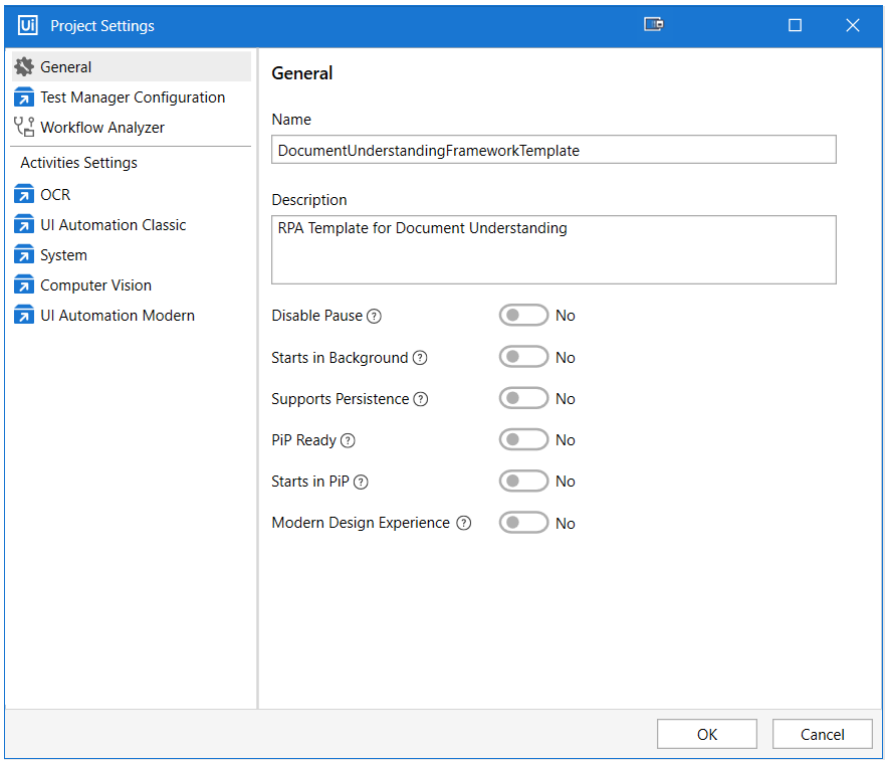
Main workflow for Attended Processes

Right-click on **Main-Attended.xaml** and set it as the **Main** workflow for the project.



Disable Background Running & Persistence Support

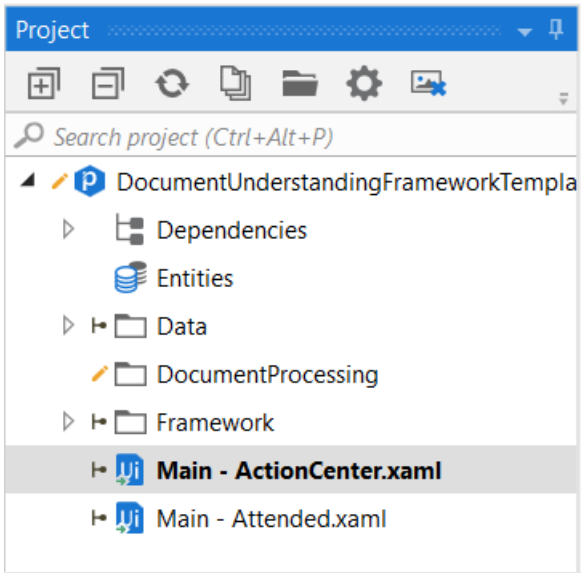
Open the Project Settings and make sure that **Starts in Background** and **Supports Persistence** are set to **No**.



Settings for Unattended Processes

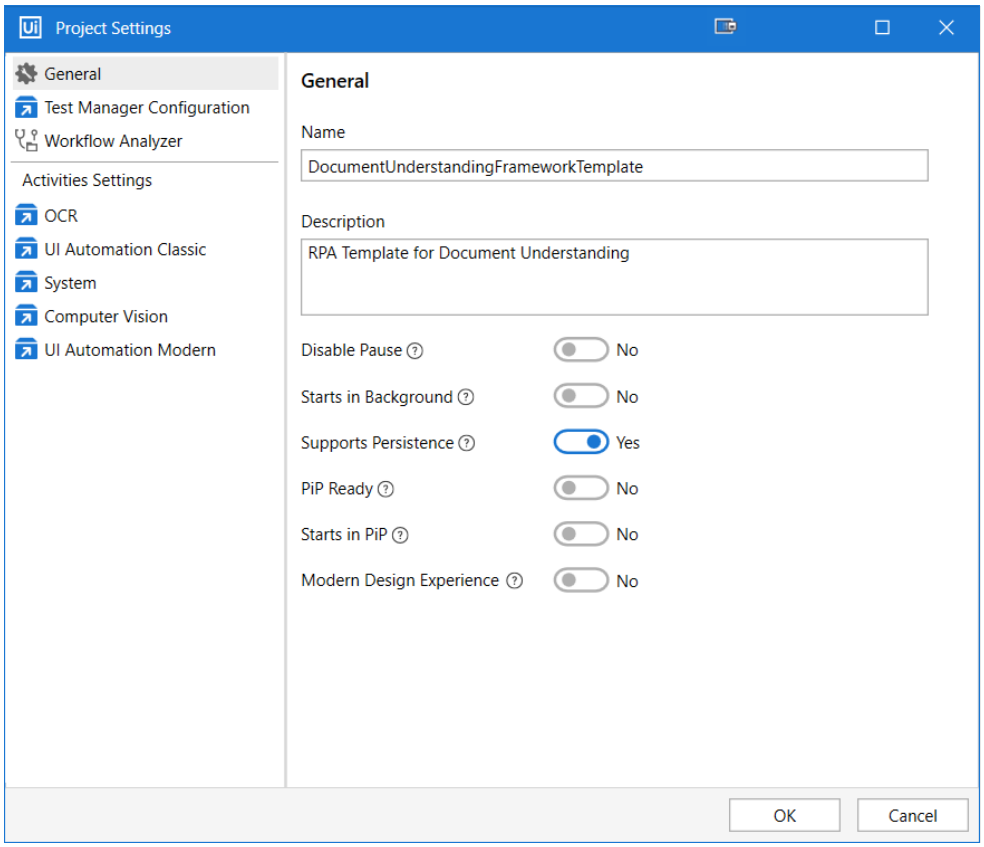
Main workflow for Unattended Processes

Right-click on **Main-ActionCenter.xaml** and set it as the **Main** workflow for the project.



Enable Persistence Support

Open the Project Settings and make sure that **Supports Persistence** is set to **Yes**.



Example Implementation

To facilitate understanding the Document Understanding Process template, it comes with a pre-implemented example. It showcases the processing of 4 types of documents.

Workflow-specific details can be found in the next chapter.

Project Files

Data\Config.xlsx

Configuration file for project settings. Minimum configuration required:

- Configure your Document Understanding Api Key under an **Orchestrator Asset** named **DocumentUnderstandingApiKey**.
- If using Action Center, configure the **StorageBucketName** under the **Settings** sheet.
- Configure the **DocumentUnderstandingQueueName** if using queues.

Main-ActionCenter.xaml

Workflow to be set and used as Main for attended Document Understanding processes that use Action Center for Human-in-the-Loop.

Arguments:

- in_TargetFile (default **Nothing**): specifies what file should be processed.
- in_UseQueue (default **False**): specifies whether Orchestrator queues are used. If set to True, the value of the in_TargetFile argument is ignored and the file to be processed is fetched from the Transaction Item.

Main-Attended.xaml

Workflow to be set and used as Main for attended Document Understanding processes.

Arguments:

- in_TargetFile (default **Nothing**): specifies what file should be processed.
- in_UseQueue (default **True**): specifies whether Orchestrator queues are used. If set to True, the value of the in_TargetFile argument is ignored and the file to be processed is fetched from the Transaction Item.

Framework\00_ReadConfigFile.xlsx

Reads the contents of the Config file into a Config dictionary at runtime.

Note: Does not load Orchestrator assets!

No custom code was added here for the purpose of creating the example implementation.

Framework\10_InitializeProcess.xaml

Workflow that loads the Taxonomy and Orchestrator assets. Any process-specific initialization code belongs here.

No custom code was added here for the purpose of creating the example implementation.

Framework\15_GetTransactionItem.xaml

Gets the next Transaction Item when using Orchestrator queues. The target file to be processed is expected to be found under the **TargetFile** key of the Transaction Item's **SpecificContent**.

Also loads all the Transaction Item's **SpecificContent** into the Config dictionary for ease of use.

No custom code was added here for the purpose of creating the example implementation.

Framework\20_Digitize.xaml

Workflow for Pre-Digitization and Digitization logic.

The example implementation uses the **UiPath Document OCR** engine.

Framework\30_Classify.xaml

Workflow for Classification.

The example implementation uses the **Intelligent Keyword Classifier** for all document types. Please know that using the **LearningData** string is also possible.

Framework\35_ClassificationBussinessRuleValidation.xaml

2021.10: Workflow for checking Classification Success. Custom logic is required to determine classification success based on process-specific business rules. A flag can be set to force all documents through manual validation.

By default, all results are sent to human validation.

Framework\40_TrainClassifiers.xaml

Workflow for Classifier training.

The example implementation uses the **Intelligent Keyword Classifier Trainer** for all document types. It also uses the **LockFile**/**UnlockFile** reusable workflows in order to

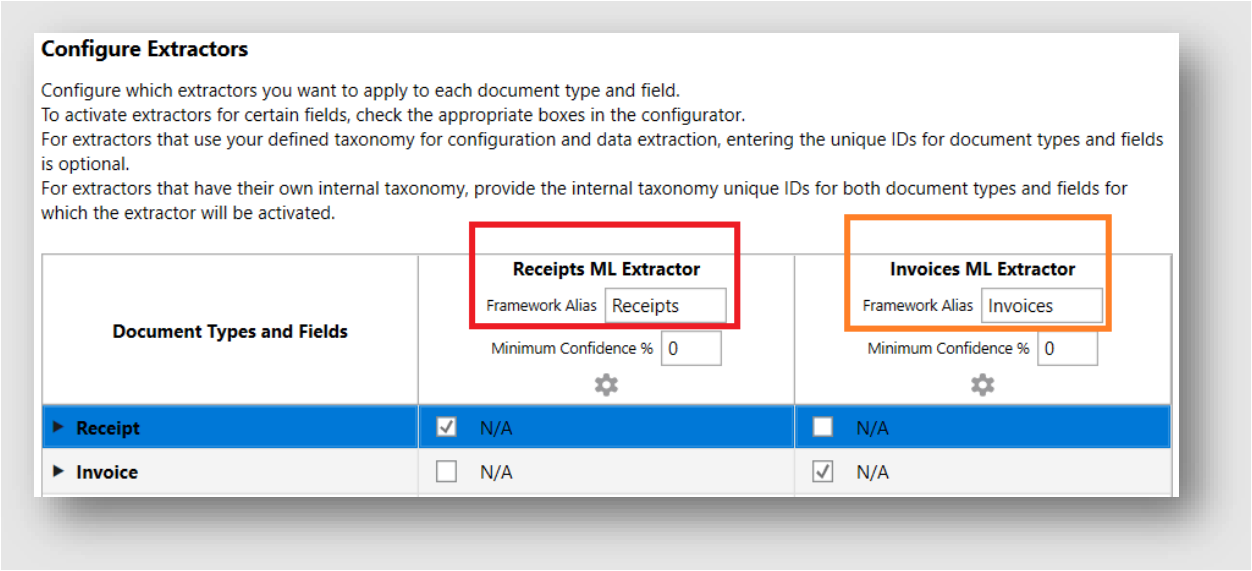
regulate access when updating the learning file. Please know that using the **LearningData** string is also possible.

Framework\50_Extract.xaml

Workflow for Pre-Extraction Processing, Data Extraction.

The example implementation uses 4 extractors:

- **Regex Based Extractor** for fixed-form Certificates
- **Intelligent Form Extractor** for fixed-form W9 Forms
- **Machine Learning Extractor** for Receipts – using “Receipts” Framework Alias for ML Extractor training
- **Machine Learning Extractor** for Invoices – using “Invoices” Framework Alias for ML Extractor training



Framework\55_ExtractionBussinessRuleValidation.xaml

2021.10: Workflow for checking Extraction Success. Custom logic is required to determine extraction success based on process-specific business rules. A flag can be set to force all documents through manual validation.

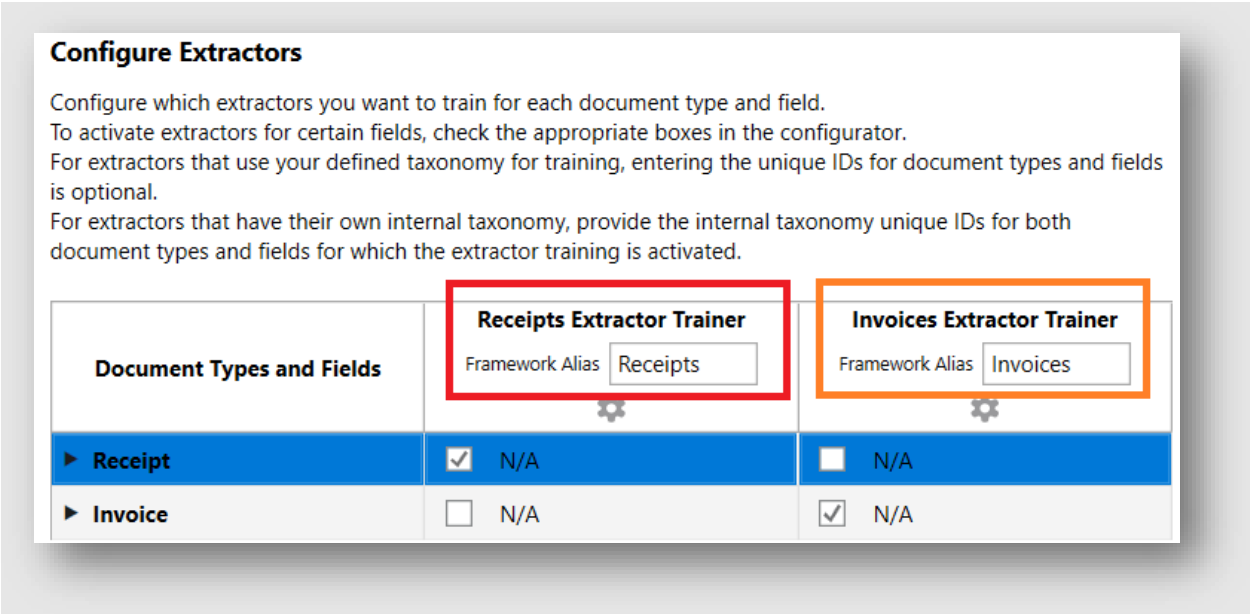
By default, all results are sent to human validation, except for invoices that go through the Invoice Post Processing workflow.

Framework\60_TrainExtractors.xaml

Workflow for Extractor training.

The example implementation uses 2 extractor trainers:

- **Machine Learning Extractor Trainer** for Receipts – using “Receipts” Framework Alias for ML Extractor training
- **Machine Learning Extractor Trainer** for Invoices – using “Invoices” Framework Alias for ML Extractor training



Framework\70_Export.xaml

Workflow for Post-Extraction Processing and Data Export logic.

The example implementation uses a simple mechanic of writing the extracted data to XLSX files. This is only one of many export possibilities besides using Data Service, a Database, using CSV, JSON format, etc.

Framework\80_EndProcess.xaml

Workflow for Post-Export Processing and Process Cleanup logic.

No custom code was added here for the purpose of creating the example implementation.

Framework\ReusableWorkflows\InvoicePostProcessing.xaml

2021.10: This workflow implements invoice post-processing using the default OOTB model. The ruleset to check against is defined in a dedicated sheet of the Config file.

The ruleset and the logic should be updated as required by the business process.

Default post-processing rules & steps:

- Verify that all Mandatory Fields and Columns are extracted
- Verify that all table rows match the rule: *Quantity* * *Unit Price* = *Line Amount*

- Verify that the sum of all *Line Amounts* = *Net Amount*
- Sum up Net Amount with all the values defined as Config "SubTotalAdditions". Verify that the sum of *Net Amount* + "SubTotalAdditions" = *Total*
- Verify the extraction confidence of all defined "ConfidenceFields" against their individual confidence thresholds
- Verify the extraction confidence of all the other fields against the "other-Confidence" threshold

This should NOT be used as-is, except for demo purposes. For a real implementation, the post-processing & validation should be tailored to the specifics of the business process.

The default implementation uses EN-US culture information. This means '.' is the decimal separator and "," is the thousand separator. (e.g.: 10,000.00)

Framework\ERR_HandleDocumentError.xaml

Workflow that is executed when an exception occurs when processing a classified document.

No custom code was added here for the purpose of creating the example implementation.

Framework\ERR_AbortProcess.xaml

Workflow that is executed if the process is aborted due to a terminating exception. Code for error cleanup or for sending error notifications belongs here.

No custom code was added here for the purpose of creating the example implementation.

Framework\ReusableWorkflows\GetWritePermission.xaml

Helper workflow using a Queue with a single queue item as a semaphore mechanism to regulate write access to classifier training files.

Framework\ReusableWorkflows\GiveUpWritePermission.xaml

Helper workflow using a Queue with a single queue item as a semaphore mechanism to regulate write access to classifier training files.

Framework\ReusableWorkflows\LockFile.xaml

Helper workflow using simple lock/unlock mechanism to regulate write access to classifier training files.

Framework\ReusableWorkflows\UnlockFile.xaml

Helper workflow using simple lock/unlock mechanism to regulate write access to classifier training files.

Framework\ReusableWorkflows\SetTransactionProgress.xaml

Updates the **TransactionProgress** for the Transaction Item that is being processed (if using queues).

Framework\ReusableWorkflows\SetTransactionStatus.xaml

Sets and log the transaction's status. The approach is like the one used by the RE-Framework.

Quick Start Guide

Please see *Document Understanding and Queues* for the required Orchestrator version of using queues.

🔧 Orchestrator Configuration

- Configure your Document Understanding Api Key under an **Orchestrator Asset** named **DocumentUnderstandingApiKey**.
- If using Action Center, create a **Storage Bucket** for your process.
- If needed, create a **Queue** for your process.

🔧 Attended Automation

- Configure your Document Understanding Api Key under an **Orchestrator Asset** named **DocumentUnderstandingApiKey**.
- If using queues, configure the **DocumentUnderstandingQueueName**.
- Configure the **Settings for Attended Processes**.

Note: We recommend becoming familiar with the solution and the project before removing the example implementation and the taxonomy.

🔧 Unattended Automation

- Configure your Document Understanding Api Key under an **Orchestrator Asset** named **DocumentUnderstandingApiKey**.
- If using Action Center, configure the **StorageBucketName** under the **Settings** sheet,
- If using queues, configure the **DocumentUnderstandingQueueName**.
- Configure the **Settings for Unattended Processes**.

Note: When developing, it is quite simple to test your implementation in Attended mode. Simply run **Main – Attended.xaml** in order to use the local validation activities instead of going to Action Center (it is **not** needed to change process settings while testing this way).

Note: We recommend becoming familiar with the solution and the project before removing the example implementation and the taxonomy.

🔧 Dispatcher Mechanisms for Unattended Implementations

Please see **Starting Document Understanding Jobs**.

Known Issues and Limitations

- Due to the current Job Count Strategy for triggers, some queue items experience delays before a processing job is triggered. This behavior is on the roadmap to be addressed.
- When using the default **LockFile**/**UnlockFile** implementation, there is a small chance that some training data will occasionally be lost – when multiple bots update the training file at the exact same time, the last one will overwrite others' training data.
- The Document Understanding Process is currently **not** performing any cleanup of its Temp folder (where it stores split or downloaded files).
- There are some errors and warnings issued by the Project Analyzer tool. We are working on having this resolved.
- By design, the job ends successfully even if the processing of all classified documents ends with exceptions.
- On C#, *Log Entry* and *Log Exit* properties in the InvokeWorkflowFile activity are not editable; issue gives compilation errors; bug will be fixed with 22.6 release.
- On C#, *Timeout* property in the InvokeWorkflowFile activity does not support variables or any kind of expression, only hard coded values are accepted; issue gives compilation errors; bug will be fixed with 22.6 release.