UiPath IXP generative extraction technical deep dive

About this course

Technical prerequisites

INTRODUCTION TO IXP

What is UiPath IXP?

How does IXP fit in agentic automation?

INTRODUCTION TO THE PAYSTUBS USE-CASE

Introduction to Paystubs use case

BUILD THE PAYSTUBS PROJECT

Build the paystub project

Validate & iterate the paystub project

EVALUATE THE PAYSTUBS PROJECT

Measure model performance

CONSUME THE PAYSTUBS PROJECT

Publish the project

Consume the project

IXP PROMPTING BEST PRACTICES

Prompting best practices

ADDITIONAL RESOURCES

Learning resources

COMPLETE THE MODULE

Complete the module

About this course

Welcome!

Step into the world of **UiPath IXP Generative Extraction** - a new approach to handling **unstructured and complex documents** with **inference-first experience.** This course is built around a common use case: extracting key data from **Paystub documents**, which are

notoriously variable and difficult to process using traditional rule-based or machine learning methods.

You'll learn how to use **prompt engineering** to design the extraction schema and improve performance by modifying the prompt instructions. You'll explore each stage of the IXP experience —**Build**, **Measure**, and **Publish**—and discover how to deploy your first IXP project in **UiPath Studio** and **Studio Web** for end-to-end automation.

Course duration: 80 minutes.

Learning objectives

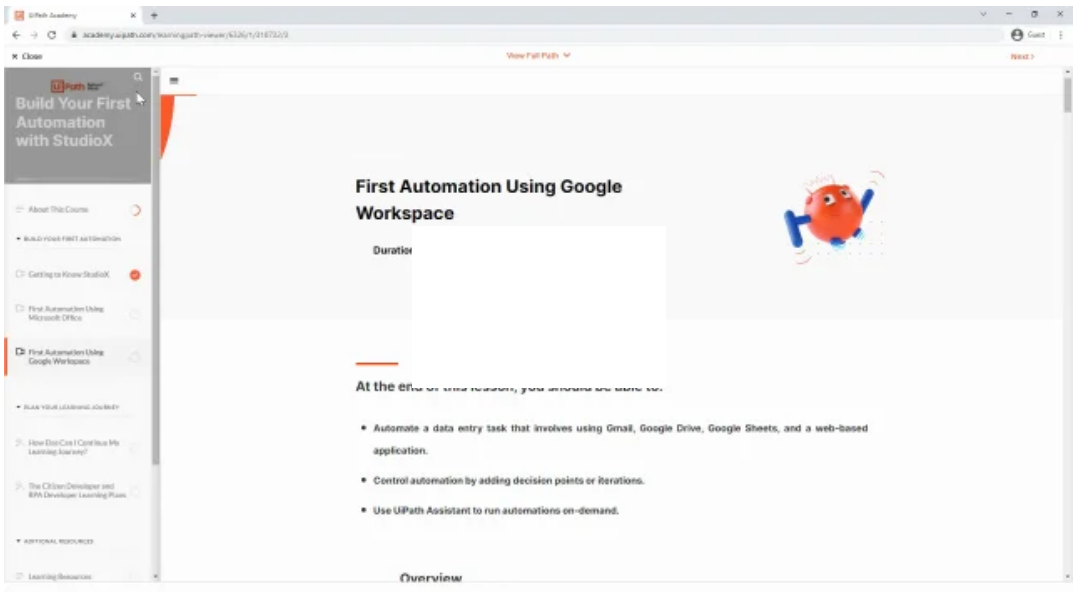
- Discover UiPath IXP capabilities.
- Understand IXP positioning in agentic automation.
- Build and iterate your IXP generative extraction project.
- Measure and optimize extraction performance.
- Publish and deploy your IXP project.
- Integrate your IXP project into automation workflows.
- Get familiar with IXP prompting best practices.

Get the most out of this course by:

- Following and building along with the videos. We provide the input documents you need before each demo.
- Attempting to solve the practice exercises without checking the solution, but rather checking the hints first. If you get stuck, try using the UiPath Documentation, and the UiPath Community Forum to find the answer to your specific problem.
- Participating in discussions on the UiPath Forum to help/receive help from fellow learners.

How to navigate through this course

Play the below video to learn how to navigate through the course and make the best use of the user interface.



Enjoy!

Technical prerequisites

Technical prerequisites

What technical setup do I need?

1 **Automation Cloud account**

An **active** Automation Cloud account to manage your automation and resources.

For more information, refer to the Automation Cloud introduction.

2

Automation Cloud tenant with IXP service and AI units

IXP service can be added by a **tenant admin.**

3

UiPath Studio Desktop

To be able to consume the UiPath IXP project within your automation workflows, you need to have UiPath Studio Desktop installed.

For more information, refer to the UiPath Studio Desktop documentation.

4

UiPath Assistant or Robot connected

To be able to consume the UiPath IXP project within your automation workflows in UiPath Studio Desktop, you need to be connected to that specific organization and tenant.

For more information, refer to the Orchestrator user documentation.

Resources

1

Sample documents

You can use the samples below to create the UiPath IXP Generative extraction project.

ZIP

samples.zip
4 MB

2

Taxonomy schema (Studio Desktop)

To simplify the taxonomy creation during the UiPath Studio Desktop implementation, you can use the predefined taxonomy file provided below.

After installing the UiPath.IntelligentOCR.Activities, replace the taxonomy.json file from DocumentProcessing folder with the one below.

FILE

taxonomy.json

29.2 KB

3

Solution implementation

Below is the solution containing all the workflows presented in the use case demo video, found in the lesson Introduction to Paystubs use-case.

FILE

Paystubs_LoanProcessing_WF.uis

146.4 KB

Lesson 3 of 13

What is UiPath IXP?

What is UiPath IXP?

- **Duration:** 5 min

What you'll learn in this lesson:

At the end of this lesson you should be able to:

1. Discover UiPath IXP capabilities and the new generative extraction capability for complex and unstructured documents.

IXP stands for **Intelligent Xtraction & Processing**, and it is a multi-modal data extraction capability. IXP converts data from unstructured documents and messages into structured data for automation and analytics.

It's built to unify and simplify how enterprises extract structured information from **unstructured inputs**—including documents, emails, tickets, and other business communications.

Category	Intelligent document processing (IDP)		
Product	**UiPath IXP** Multi-modal data classification and extraction that unlocks enterprise data at speed and scale		
Product capabilities	**Communications mining**	**Document understanding***	**Generative extraction for unstructured & complex documents**
Description	Extract key intent, sentiment and context data from messages and communications.	Extract relevant data from structured and semi-structured documents.	Extract relevant data from high complexity, varied structured documents.
	Example documents Emails, workflow tickets, and messages etc.	**Example documents** Forms, purchase orders, applications, and invoices etc.	**Example documents** Reports, contracts, and lease agreements etc.

**Available as a standalone capability in Automation Cloud, Automation Suite & Standalone deployment*

In the following lessons, we will focus on the **unstructured and complex documents** capability, which is also known as **IXP Generative Extraction**.

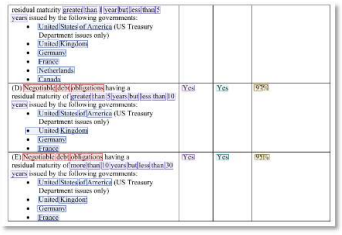
IXP Generative Extraction capability

IXP Generative Extraction is a new **inference-first experience** that provides **zero-shot and few-shot extraction** for the most complex and unstructured documents such as

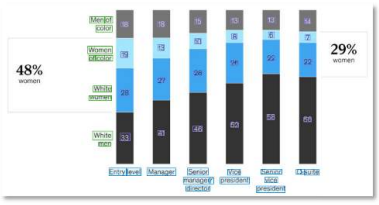
contracts, mortgage applications, or medical documents.

- Handles documents containing:
    - Free-form paragraphs, narratives, or text
    - Complex elements such as nested tables, charts, graphics, checkboxes, call-out boxes, signatures, handwriting, etc.

- Supports extraction of inferred values—data not explicitly shown, but deduced via document context
- Works well for:
    - Document layouts and formats that vary significantly
    - Documents requiring flexible extraction schema
    - Multiple document types that are stacked in a single file (i.e. packet documents)



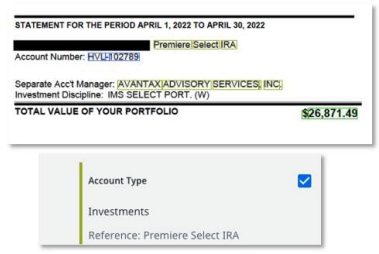
Complex tables



Charts & Graphs

Handwriting, Signatures & Stamps

Packet documents



Inferred values

Checkboxes & Call-out-boxes

When to use IXP Generative Extraction?

Choose this capability when one or more of the following apply:

- Documents include **paragraphs**, **tables**, **handwriting**, or **graphic elements**
- Layouts and fields **vary widely across documents**
- Some extracted values must be **inferred** or **aggregated**
- **Multiple document types** are combined within one file, and splitting is not possible
- Documents have **less than 150 pages** (temporary limitation)

Use **Document Understanding** for **structured/semi-structured** documents with no complex elements or where an existing pre-trained OOB model is applicable.

Use **Communications Mining** for **unstructured communications** like emails, tickets, chats, or other conversational content.

Key takeaways

- [ ] UiPath IXP is the unified, AI-powered solution for turning diverse document and communication content into structured, actionable data
- [ ] IXP Generative Extraction capability enables inference-first extraction for highly complex and unstructured documents
- [ ] Use this capability for documents with high variability, complex elements, or inferred values

Want to find out more?

Use the resource below to learn more about this topic.

UiPath IXP - Official documentation
Learn more about IXP.

GO TO DOCS

Next, let’s take a look at how IXP & Agents can work together.

How does IXP fit in agentic automation?

How does IXP fit in agentic automation?

- **Duration:** 5 min

Overview

This lesson introduces how UiPath IXP and UiPath Agents can work together to automate end-to-end business processes at scale. Learners will understand how agents can leverage IXP as a tool, data source, or document reasoning engine. Through real-world examples like remittance processing, the lesson illustrates various integration patterns between IXP, agents, robots, and humans.

Why use IXP and Agents together?

UiPath Agents are AI-powered, task-executing entities capable of autonomous decision-making. Agents perform reasoning over data, select tools (like IXP, APIs, or other systems), and complete tasks without being explicitly programmed step-by-step.

While IXP excels at **understanding complex documents** and **providing reliable structured data**, Agents excel at **contextual decision-making** and **multi-step automation orchestration**. Together, they enable end-to-end automation of document-centric processes.

Collaboration patterns

There are three primary ways UiPath Agents and IXP can work together:

1

Pattern A: Agent receives IXP output as input

- The IXP model processes incoming documents and extracts structured data points.
- The extracted data is passed to the Agent, which might apply additional business rules or post-processing logic to evaluate the outputs.
- Based on this evaluation, the Agent might:

- Approve or reject a document
- Generate a report
- Upload data into another system
- Trigger escalations

✅ **Use Case Fit**: repeatable document processing tasks that require agentic reasoning

2

Pattern B: Agent uses IXP as a tool during execution

- The Agent decides mid-task to invoke an IXP model.
- The IXP model acts like a callable tool: extract structured data → return it → continue the task logic.

✅ **Use Case Fit**: dynamic or context-dependent extraction tasks

3

Pattern C: Agent queries IXP as a repository or index

- The Agent can search and query previously processed documents within IXP.
- This pattern supports retrospective and analytical questions like:
  - “How many claims over $10K did we process last month?”
  - “What was the most common denial reason in Q2?”
  - “Show all contracts signed with Vendor X.”

✅ **Use Case Fit**: Auditing, reporting, data analytics, compliance validation

Agent receiving IXP model output as an input

Documents/messages are processed using IXP and these outputs are sent to an Agent as inputs to execute some agentic reasoning/logic

Agent selecting to use an IXP model for a task

Agent has access to IXP models as tools to process documents/ messages as part of the agent run.

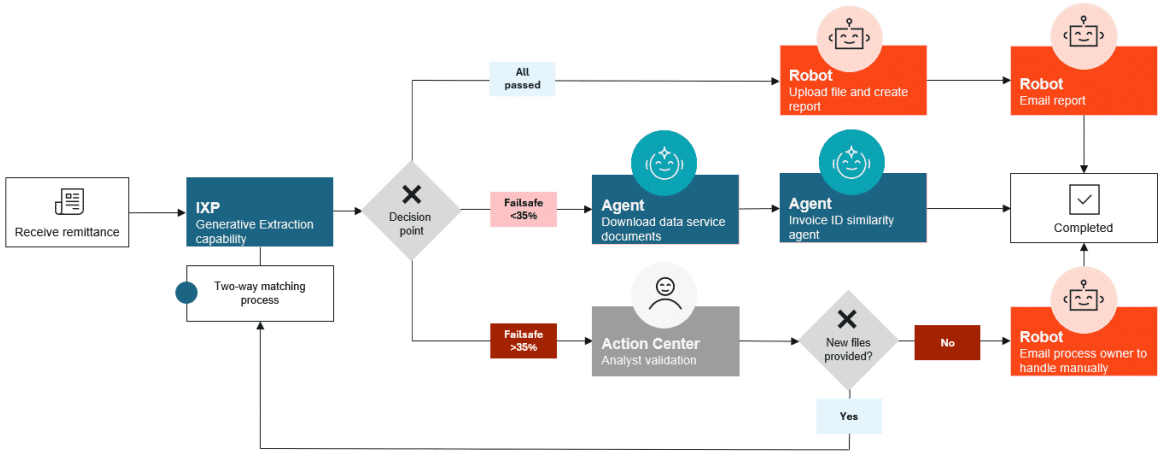
Agent querying IXP as a repository to complete a task

Agent has access to IXP project as a tool/index to do Q&A (e.g., how many contracts were processed in Q4, what was the average contract amount, etc.)

Example use-case

Here is an example of how IXP works together with agents, robots and humans to automate an end-to-end process.

Remittance processing is the workflow of receiving, validating, and reconciling payments from customers or clients. It typically involves extracting key data from remittance documents—such as payer details, invoice numbers, and payment amounts—and matching them with open invoices in financial systems. This process entails accurate allocation of payments and helps maintain up-to-date accounts receivable records.



Benefits of IXP-Agents collaboration

Integrating IXP and Agents offers strategic advantages beyond what each tool can do alone.

IXP Generative extraction has been built to **reduce document processing time-to-value** by streamlining:

- How the prompts are defined (i.e., what do you want to do)
- How the ground truth is defined (i.e., what do you expect to happen)
- How the predictions are compared to ground truth (i.e., are your expectations met)

User Experience

Document-focused user experience to see the document, predicted values, ground truth, and references all in one view

Templates

Out-of-the-box templates to get a head start on common document types across different industries

Taxonomy Manager

Built-in taxonomy manager to easily map model requirements to your business requirements

Performance Metrics

Out-of-the-box performance metrics to understand performance for each level of your taxonomy

Currently, IXP Generative extraction works well for handling:

✅ **Repeatable** extraction tasks with **well-defined** taxonomies

✅. **Single** PDF or image as input

✅ Documents with **150 pages or less**

In the future, IXP Generative extraction will support:

📌 Extraction tasks with **dynamic, context-dependent** taxonomies

📌 Tasks where **multiple** PDF/image files need to be processed in the same request

📌 Documents with **more than 150 pages**

Next, let's take a look at the use case featured in this course.

Introduction to Paystubs use case

Introduction to Paystubs use case

- **Duration:** 10 min

Overview

In this lesson, you'll explore a **real-world use case** involving the automation of **paystub extraction** in the context of **loan application processing**. Financial institutions require accurate and timely income verification from applicants—often through paystubs. However, due to

diverse formats, scanned files, and document complexity, manual review is time-consuming and error-prone.

You'll learn how UiPath IXP can solve this problem through **inference-first extraction**, making it an ideal tool for automating high-volume, high-variety documents like paystubs.

Business Context: Why paystubs matter?

In industries such as banking, insurance, and HR services, paystubs are among the **most frequently submitted documents**. They serve as proof of income, employment, and financial stability. Whether someone is applying for a personal loan, a mortgage, government benefits, or simply onboarding to a new job, they're often asked to submit multiple paystubs from recent months.

Loan officers and verification agents rely on this data to make critical business decisions. However, the volume and inconsistency of these documents create bottlenecks, especially when reviewing hundreds of them per day. Automating this workflow can significantly reduce turnaround time, minimize manual effort, and improve data accuracy —key goals for any enterprise automation strategy.

Challenges in paystub processing

Despite them being a common document type, paystubs are **not standardized**. Each employer uses their own payroll provider or in-house system, leading to hundreds or even thousands of potential layouts. Some are exported as PDFs with selectable text, while others are scanned copies or even photos from mobile phones. Key fields may be

presented in a variety of ways (e.g., “Pay Period” vs. “Earnings From/To”), and data may be split across sections or require context to interpret.

This inconsistency challenges the traditional rule-based or template-based extraction methods. Simple OCR combined with regex might work on one layout, but can fail entirely on a slightly different layout. The unpredictability of table structures, column headers, and footnotes further complicates the extraction. What’s needed is a **flexible, layout-agnostic extraction** approach that uses **natural language prompting** —something that UiPath IXP delivers through the Generative extraction capability.

Scenario overview

You're working with a **mid-sized financial institution** that handles **consumer loan applications** online. One of the most common requirements for each applicant is to provide a **proof of income** in the form of **recent paystubs** (also called payslips). Paystubs must be reviewed to verify the applicant's income, employment, and financial stability before the loan can be approved.

The current process is **manual**:

- A back-office agent opens each application
- Downloads and opens each paystub
- Searches for and copies key data into an internal loan management system
- Flags incomplete or inconsistent files for follow-up

As application volume increases, the manual approach leads to:

- Long turnaround times (2–3 days per loan file)
- Frequent data entry errors
- Resource bottlenecks during peak application periods

The bank wants to automate this process with **UiPath IXP**, leveraging the **Generative extraction capability**, which will serve as the foundation for your course project.

Automation goals for the course use-case

Your goal is to design and implement a **paystub extraction model** that can:

1. Automatically extract critical income-related data from diverse paystub formats
2. Handle unstructured documents using natural language instructions
3. Flag missing fields or inconsistencies using validation logic
4. Integrate the results with downstream UiPath workflows for decisions or manual review

This use case will be built incrementally as you progress throughout the course.

Extraction fields needed

In an automated paystub review process, it's critical to extract the right fields with high confidence. These fields form the foundation for downstream calculations, validations, and decisions.

For instance:

- **Employee Name** and **Employer Name** are used to match identity and verify employment.

- **Gross Pay** and **Net Pay** help assess income level and loan affordability.
- **Pay Period** and **Pay Date** ensure income is recent and consistent.
- **Deductions** and **Taxes** help lenders understand financial obligations that affect disposable income.
- **Hours Worked** and **Pay Rate** give insight into job stability and payment regularity.

Each of these data points must be captured **accurately**, even when they're buried in variable formats, across multiple lines, or require aggregation.

Why use IXP Generative extraction for this use case?

UiPath IXP Generative extraction capability solves this problem by offering a powerful, genAI powered extraction tool driven by **prompt engineering**. Instead of configuring position-based rules or training machine learning models, users simply describe what they want to extract in natural language. IXP uses **large language models (LLMs)** under the hood to understand the document context and identify the relevant data points, regardless of its layout.

IXP Generative extraction capability is built to:

- Handle multi-layout documents
- Support extraction from complex tables
- Enable rapid iteration and testing
- Enable seamless deployment in Studio Web or Studio Desktop

Next, let's explore how to build the UiPath IXP Generative extraction project.

Build the paystub project

Build the paystub project

- **Duration:** 10 min

What you'll learn in this lesson

At the end of this lesson you should be able to:

1. Create a new IXP project using OOTB template tailored for paystubs.

2 Select a representative sample of documents to test.

3 Configure advanced model settings, including intelligent pre-processing.

Create the project

Start by creating your own **dedicated project**. This project acts as your personal workspace where you can upload sample documents, design the extraction schema, craft your prompt instructions, and test the results.

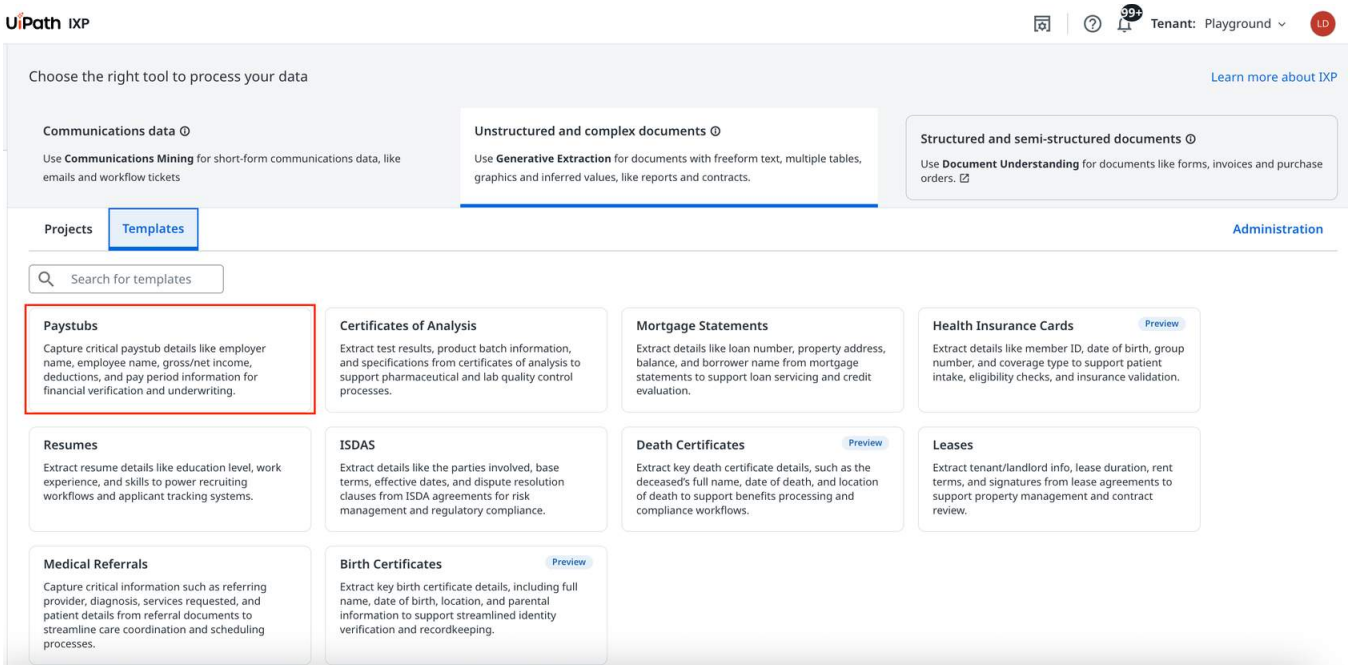
IXP generative extraction offers two ways to get started:

- **Start from scratch** — build a brand new project tailored to your specific use case.
- **Use a predefined template** — use one of the out-of-the-box templates with pre-configured fields and prompt instructions.

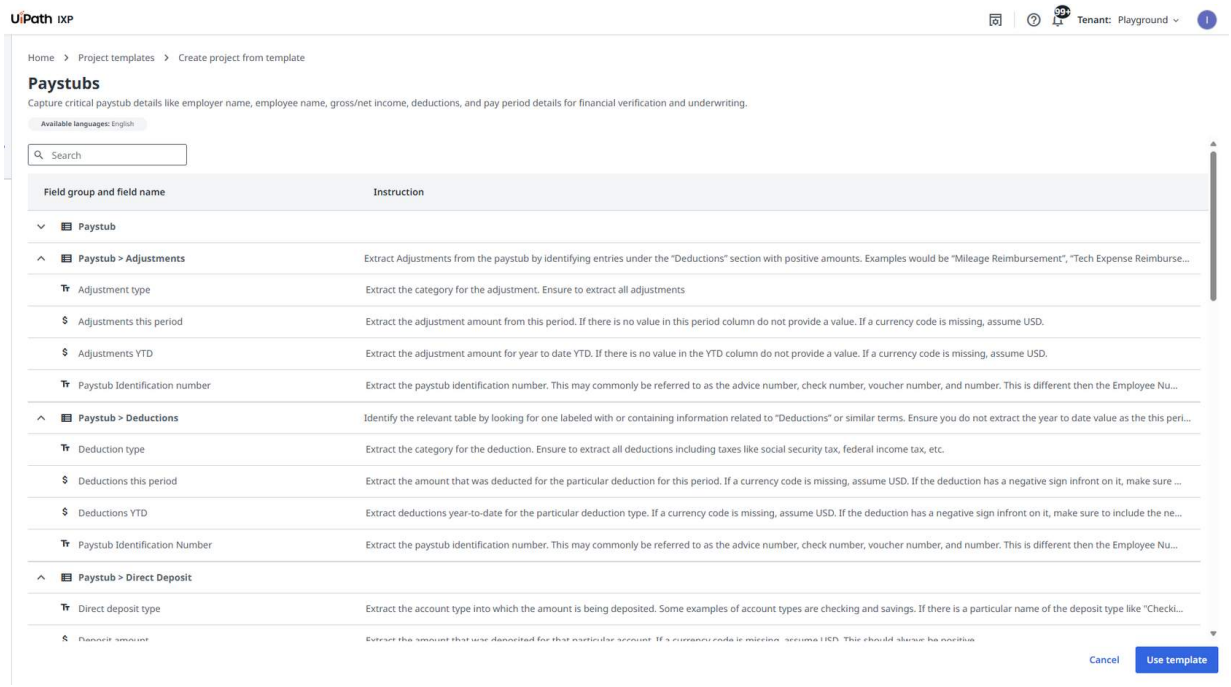
For the purpose of this course, you will be using the paystubs template.

Follow the steps below to create your project:

1. Navigate to **UiPath IXP** and select the **Unstructured and complex documents** capability. Click on **Templates** and select the **Paystubs** template.



2. You'll see a **predefined extraction schema** that's structured in logical field groups. Each field group contains a set of fields and instructions that will be used to extract the relevant information. Click **Use Template** to create the project and give it a descriptive name, such as 'Paystub_Extraction_LoanProcessing'.



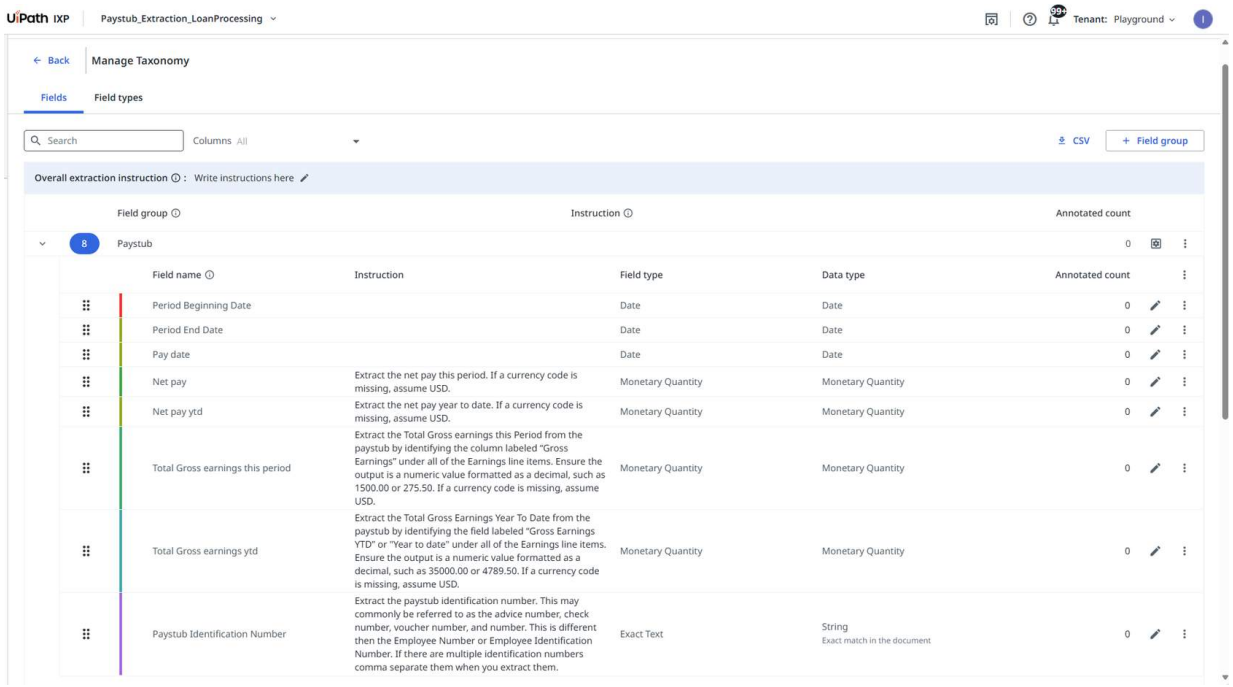
3. Once your project is created, go to **Manage Taxonomy** and review the extraction schema. The schema can be customized, if required.

    a. **Field group** are used to organize similar fields into logical sets.

    b. **Fields** are specific data points to be extracted from the document.

    c. **Field type** act as bases for fields and determine their data types (e.g., string, number, date, etc.) Field types can be shared across multiple fields.

    d. **Instructions** are natural language prompts that provide context on how to identify and extract data from the document. There are various levels of instructions, including the overall project instruction, field group instructions, field type instructions, and individual field instructions.



Upload sample documents

It is crucial to test the schema on a **representative sample of documents** that mirror the **real-world complexity and diversity**.

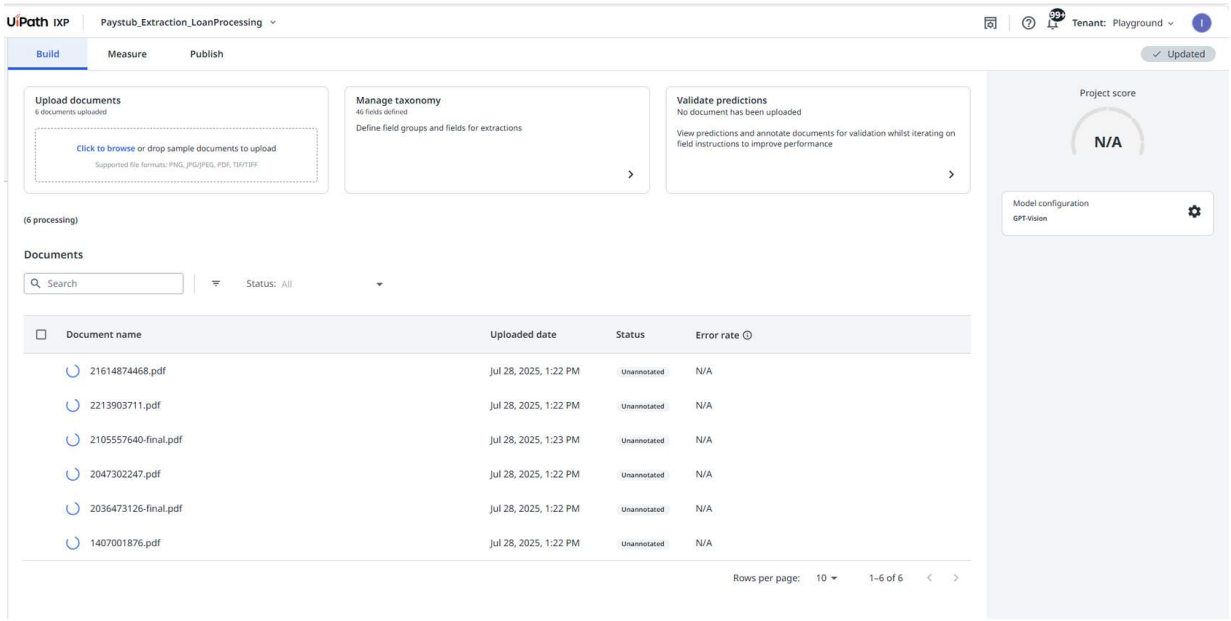
What to include in your Paystubs sample set:

- Paystubs from multiple companies or payroll systems
- A mix of different document formats (e.g., native PDFs and scanned PDFs or images)
- Paystubs with various layouts and elements:
  - Different sections (earnings, benefits, YTD totals)
  - Different fields (e.g., "Net Pay", "Take-home")

- Complex tables (hourly breakdowns, bonuses, deductions)

For the purpose of this course, you will be using a small sample of 6 paystubs (these can be found in the *Prerequisites* section).

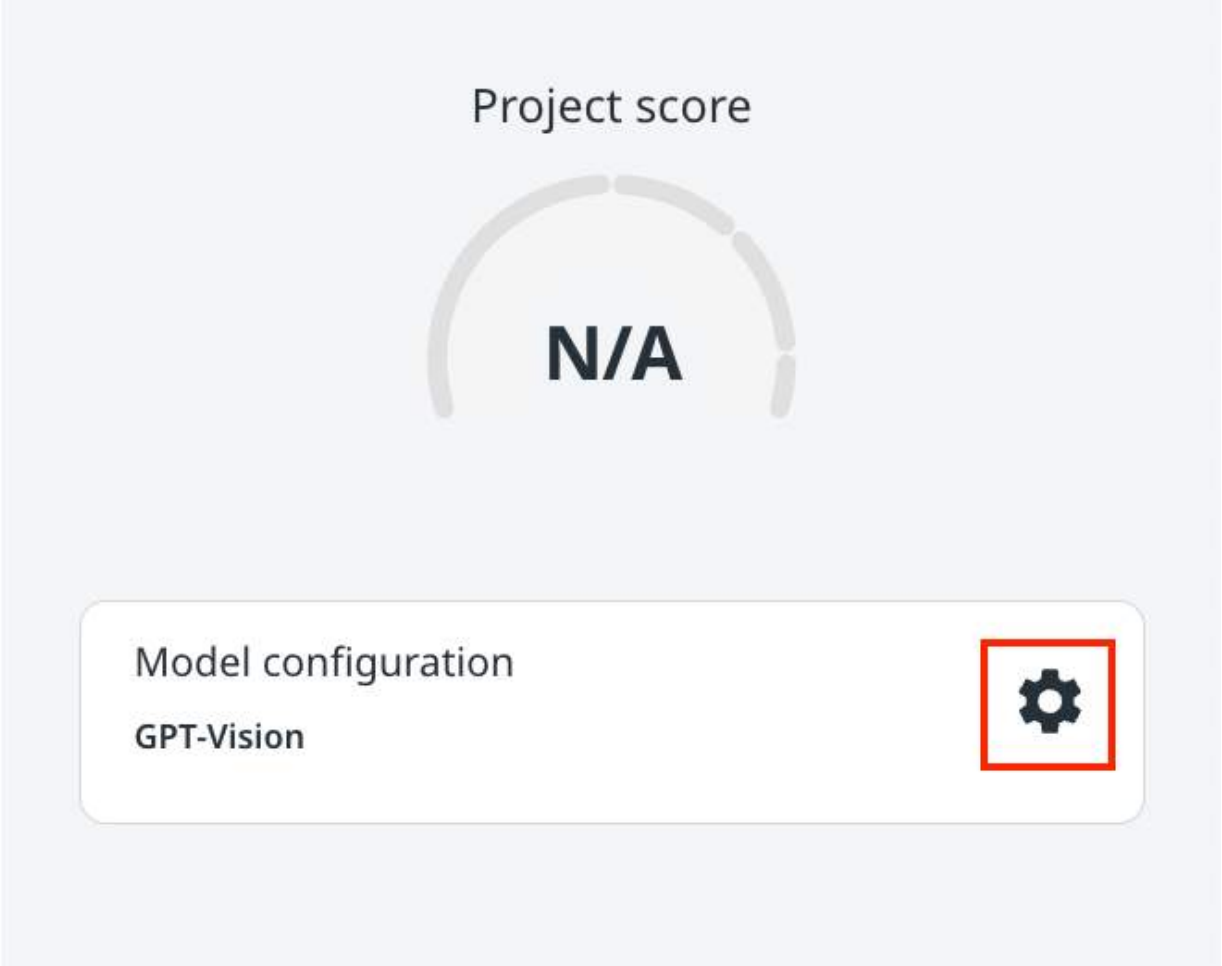
To upload your documents, navigate to the **Build tab** and use **Click to browse** to select your files or simply **drag and drop** them into the designated upload area.

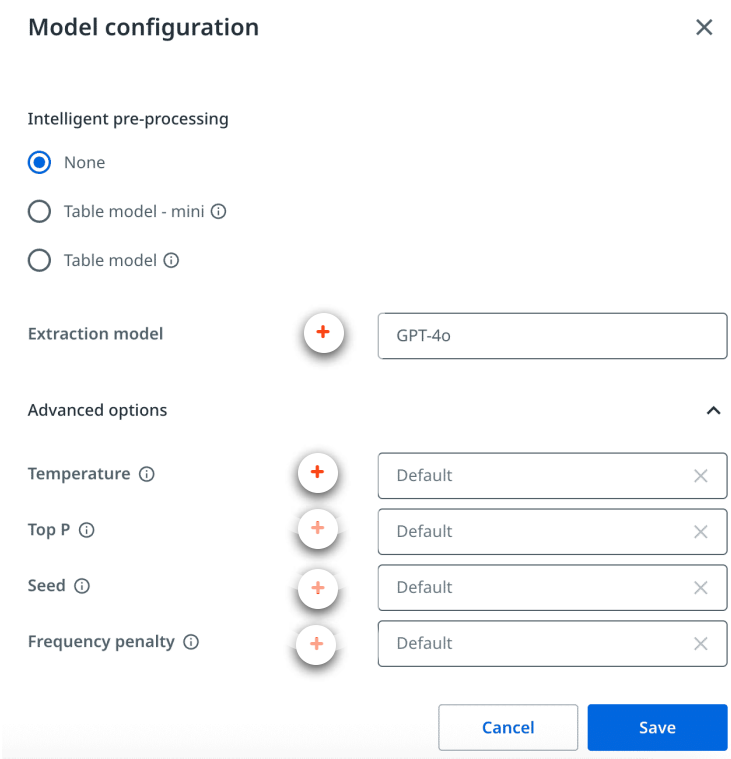


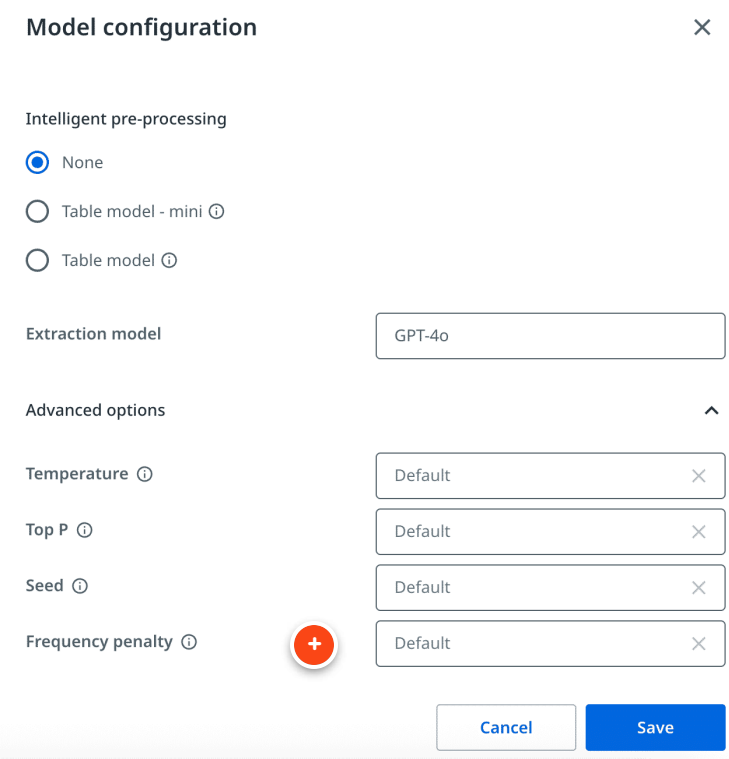
Configure the model

You can configure your **extraction model** and other **advanced settings** in Model configuration. Changing these settings can impact model's predictions and improve its performance.

To change model settings, navigate to the **Build tab** and click on the **gear icon**.

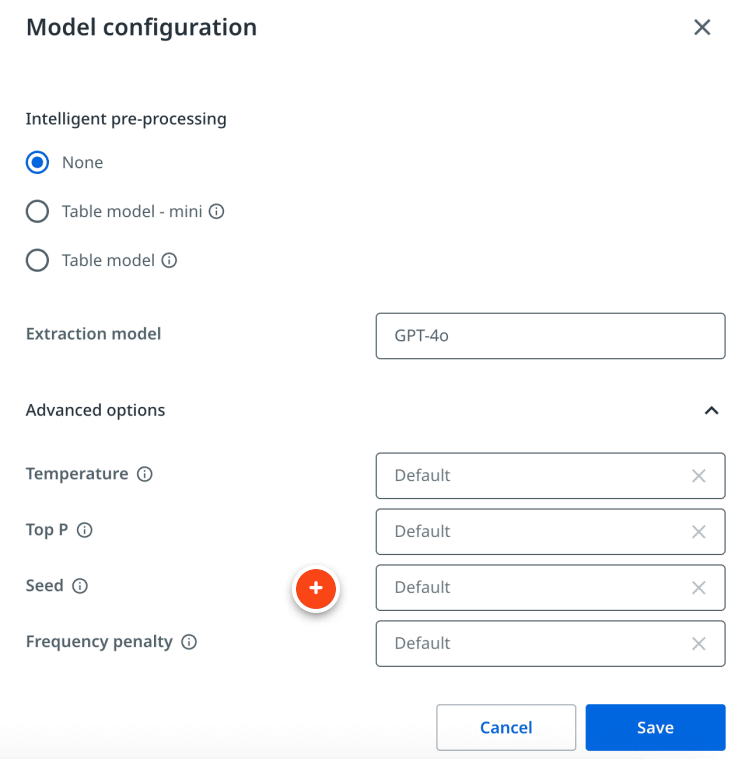



Model configuration
Intelligent pre-processing
None
Table model - mini
Table model
Extraction model
GPT-4o
Advanced options
Temperature
Default
Top P
Default
Seed
Default
Frequency penalty
Default
Cancel
Save



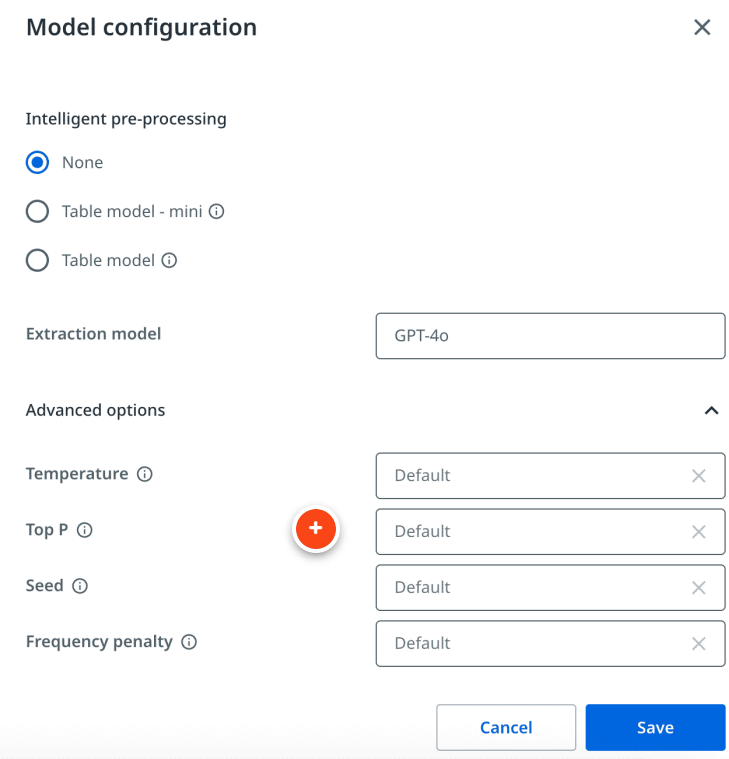
Frequency penalty

Number between -2.0 and 2.0. Positive values reduce the likelihood the model will repeat tokens present in the text so far.



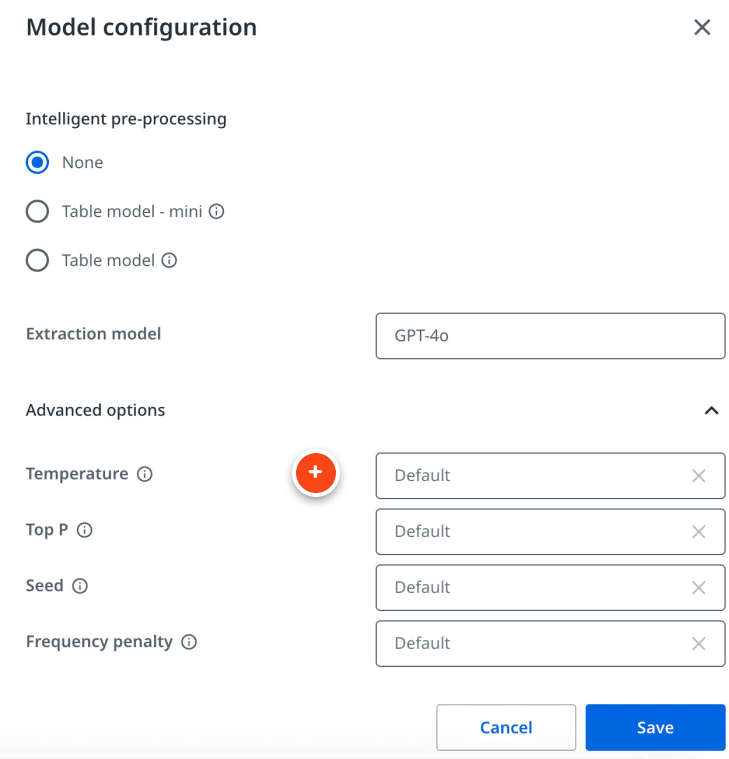
Seed

If specified, repeated requests with the same seed and parameters should return the same result.



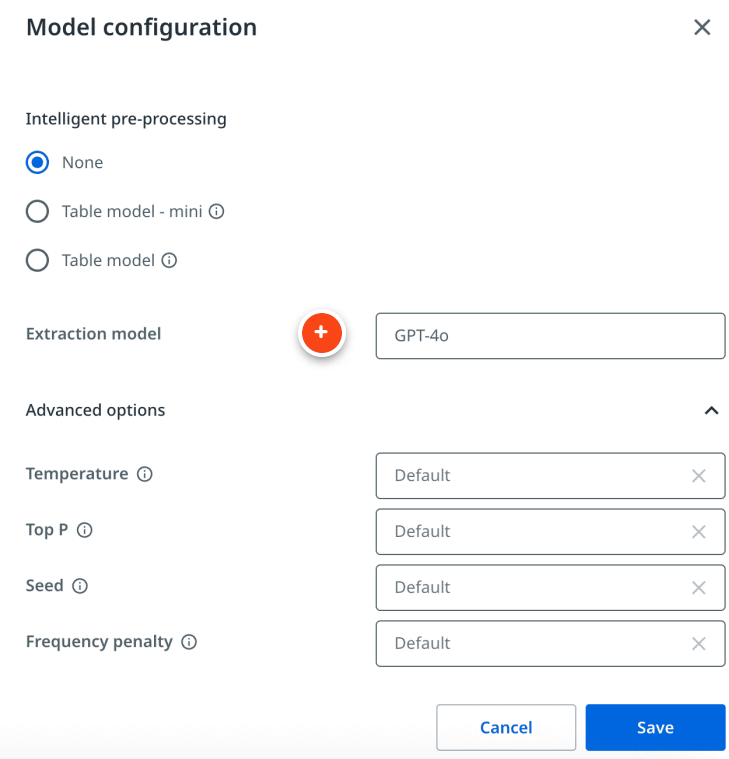
Top P

Number between 0.0 and 1.0. Only sample from tokens with the top_p probability mass.



Temperature

Number between 0.0 and 2.0. The sampling temperature to use - higher values will make the output more random.



Extraction model

The underlying LLM. Please note that currently, only GPT-Vision is supported. Additional LLMs will be added in future releases.

> Currently, the most important model setting to consider is **Intelligent pre-processing**. All other settings (Temperature, Top P, Seed, Frequency penalty) have been preconfigured to the default values based on internal testing and research.

What is intelligent pre-processing?

Intelligent pre-processing enables the **image + text model**, which improves **table detection and reconstruction.** This is particularly helpful when dealing with complex tables like paystubs.

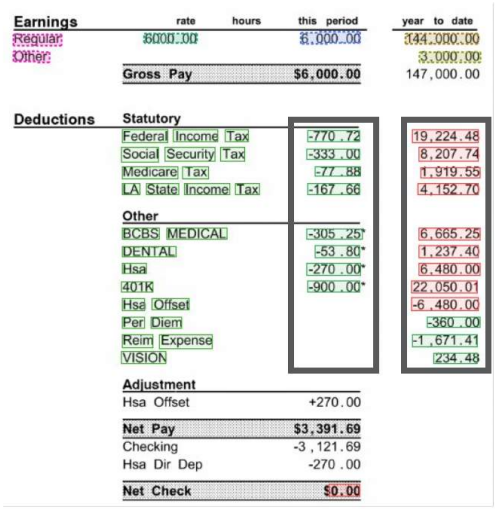
There are 3 options available:

- **None:** use this setting if your documents do not contain any tabular data (selected by default)
- **Table model - mini:** best suited for documents with simple tables or multiple tables
- **Table model:** best suited for documents with complex nested tables, tables with merged cells, bullet points, or tables spanning across multiple pages

Go ahead and enable the **Table model-mini** in your own Paystub project.

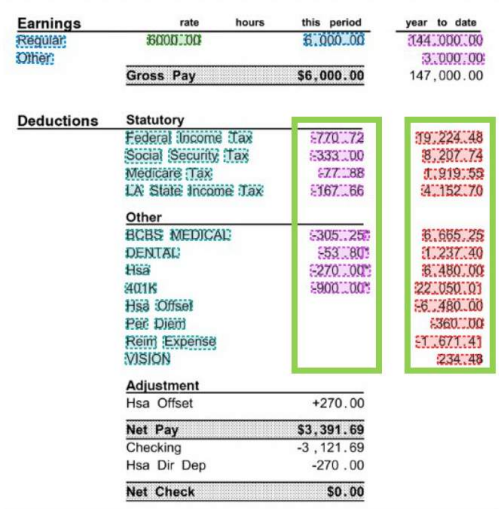
Below is an example how the Table model improved paystubs extraction.

Extraction querying LLM w/o Table Enrichment – This Period column values confused for Year to Date



Earnings	rate	hours	this period	year to date
Regular	6000.00		6,000.00	144,000.00
Other				3,000.00
Gross Pay			$6,000.00	147,000.00
Deductions	**Statutory**			
	Federal Income Tax		-770.72	19,224.48
	Social Security Tax		-333.00	8,207.74
	Medicare Tax		-77.88	1,919.55
	LA State Income Tax		-167.66	4,152.70
	Other			
	BCBS MEDICAL		-305.25*	6,665.25
	DENTAL		-53.80*	1,237.40
	Hsa		-270.00*	6,480.00
	401K		-900.00*	22,050.01
	Hsa Offset			-6,480.00
	Per Diem			-360.00
	Reim Expense			-1,671.41
	VISION			234.48
	Adjustment			
	Hsa Offset		+270.00	
Net Pay			$3,391.69	
Checking			-3,121.69	
Hsa Dir Dep			-270.00	
Net Check			$0.00	

Extraction w/ Table Enrichment: Both column values extracted correctly



Earnings	rate	hours	this period	year to date
Regular	6000.00		6,000.00	144,000.00
Other				3,000.00
Gross Pay			$6,000.00	147,000.00
Deductions	**Statutory**			
	Federal Income Tax		-770.72	19,224.48
	Social Security Tax		-333.00	8,207.74
	Medicare Tax		-77.88	1,919.55
	LA State Income Tax		-167.66	4,152.70
	Other			
	BCBS MEDICAL		-305.25	6,665.25
	DENTAL		-53.80	1,237.40
	Hsa		-270.00	6,480.00
	401K		-900.00	22,050.01
	Hsa Offset			-6,480.00
	Per Diem			-360.00
	Reim Expense			-1,671.41
	VISION			234.48
	Adjustment			
	Hsa Offset		+270.00	
Net Pay			$3,391.69	
Checking			-3,121.69	
Hsa Dir Dep			-270.00	
Net Check			$0.00	

Want to find out more?

Use the resources below to learn more about this topic.

UiPath IXP - Official documentation

Learn more about model building in IXP Generative extraction.

GO TO DOCS

Next, let’s explore how to validate model's predictions and modify prompt instructions.

Validate & iterate the paystub project

Validate & iterate the paystub project

- **Duration:** 15 min

What you'll learn in this lesson

At the end of this lesson you should be able to:

1. Review initial predictions and see how well they are performing.

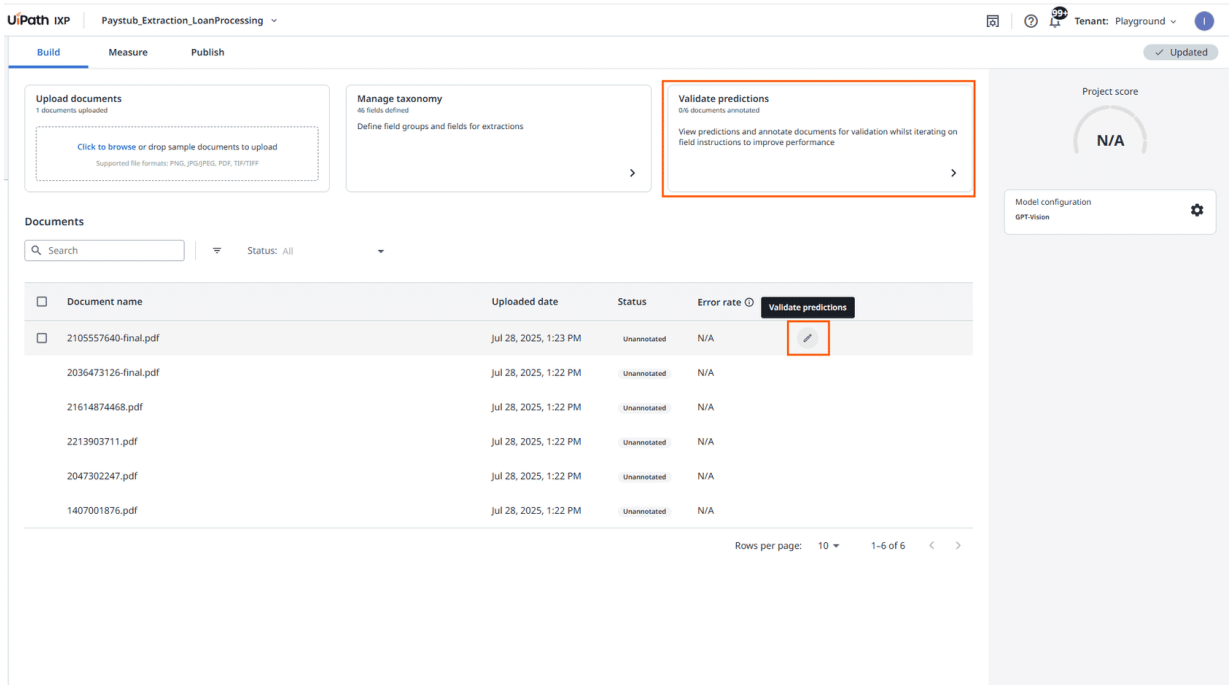
2. Modify your prompt instructions and test the impact on model's performance.

3. Validate extractions to gather ground truth for validation.

Review predictions

Once the documents are uploaded, the next step is to **review the predictions**. Predictions are generated automatically, based on the fields and instructions defined in the schema. Even though the schema has been imported as part of a template, it's essential to **review predictions** across your document set and **make necessary adjustments to your prompt instructions**, if needed.

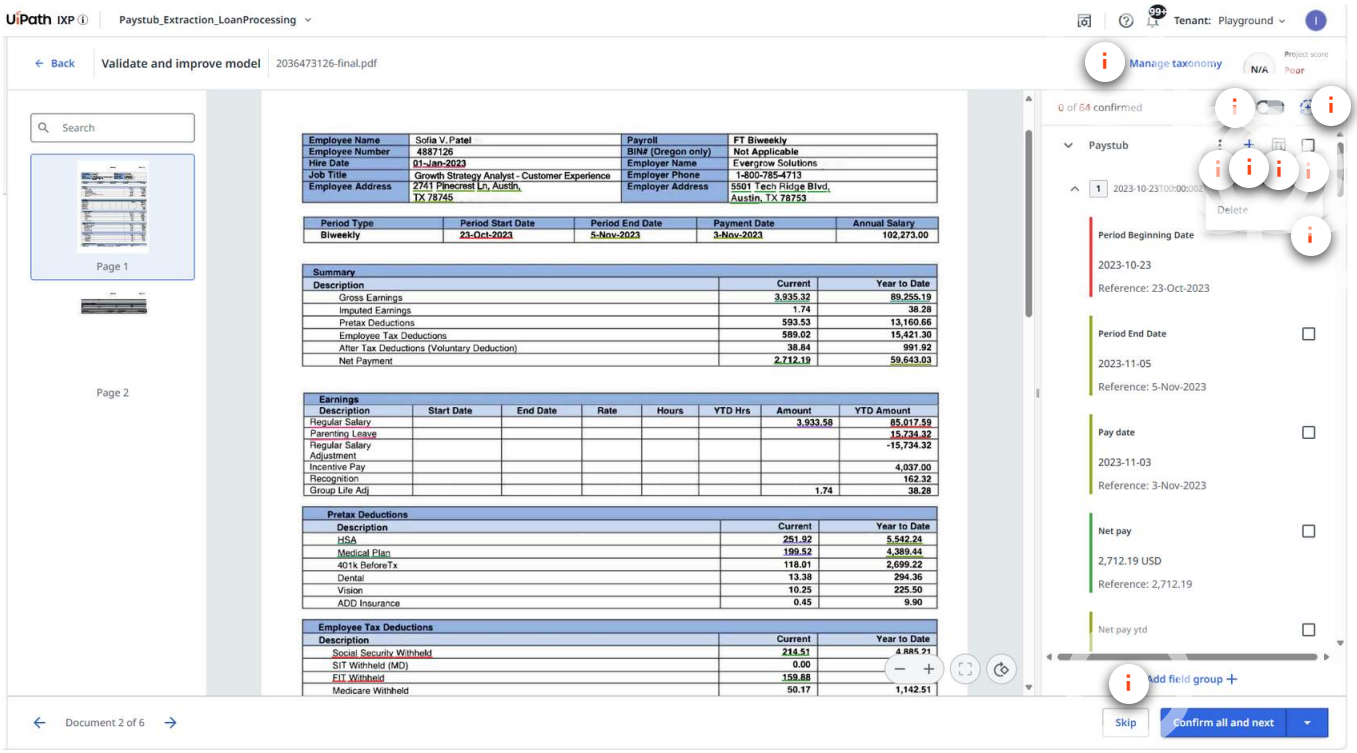
You can do this by navigating to the **Validate predictions** tab or by selecting a specific document.

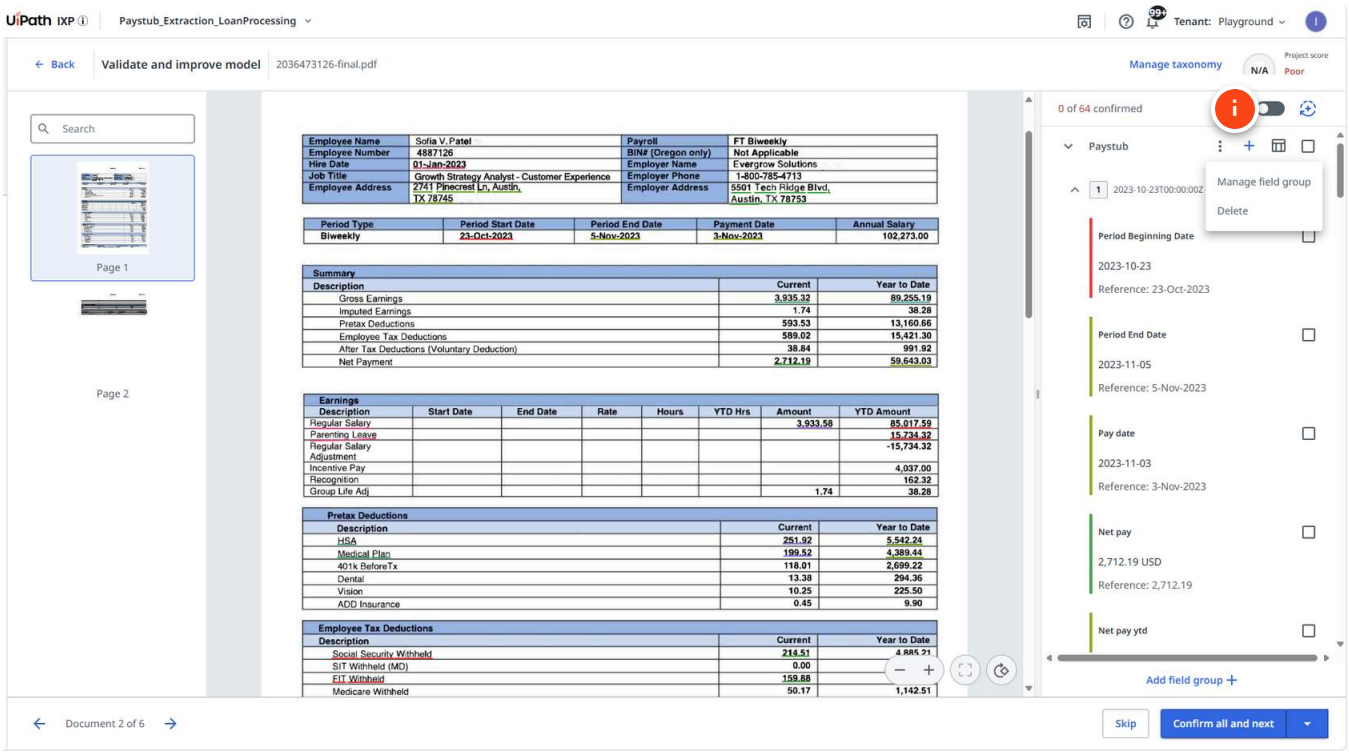


Follow these steps for your Paystubs validation:

1. Review the initial predictions on **2-3 documents**.
2. Decide whether you need to **modify your prompt instructions**.
3. Review the remaining documents in your sample. It's recommended to increase the test sample gradually (i.e., 10 → 20 → 30 → X documents). This is an **iterative process**.
4. Once the prompt instructions work relatively well across the entire sample, you can start **annotating documents**.

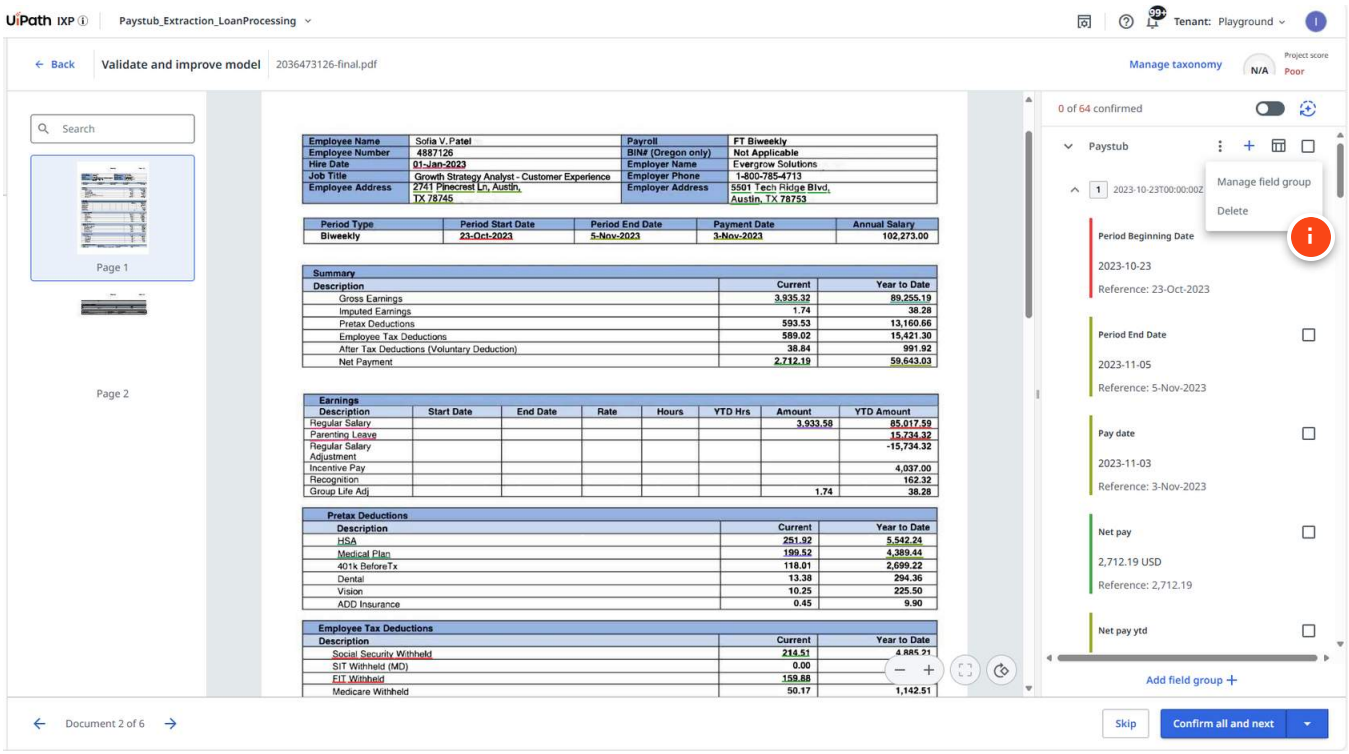
You can refer to the image below for additional tips on how to navigate to different UI features.





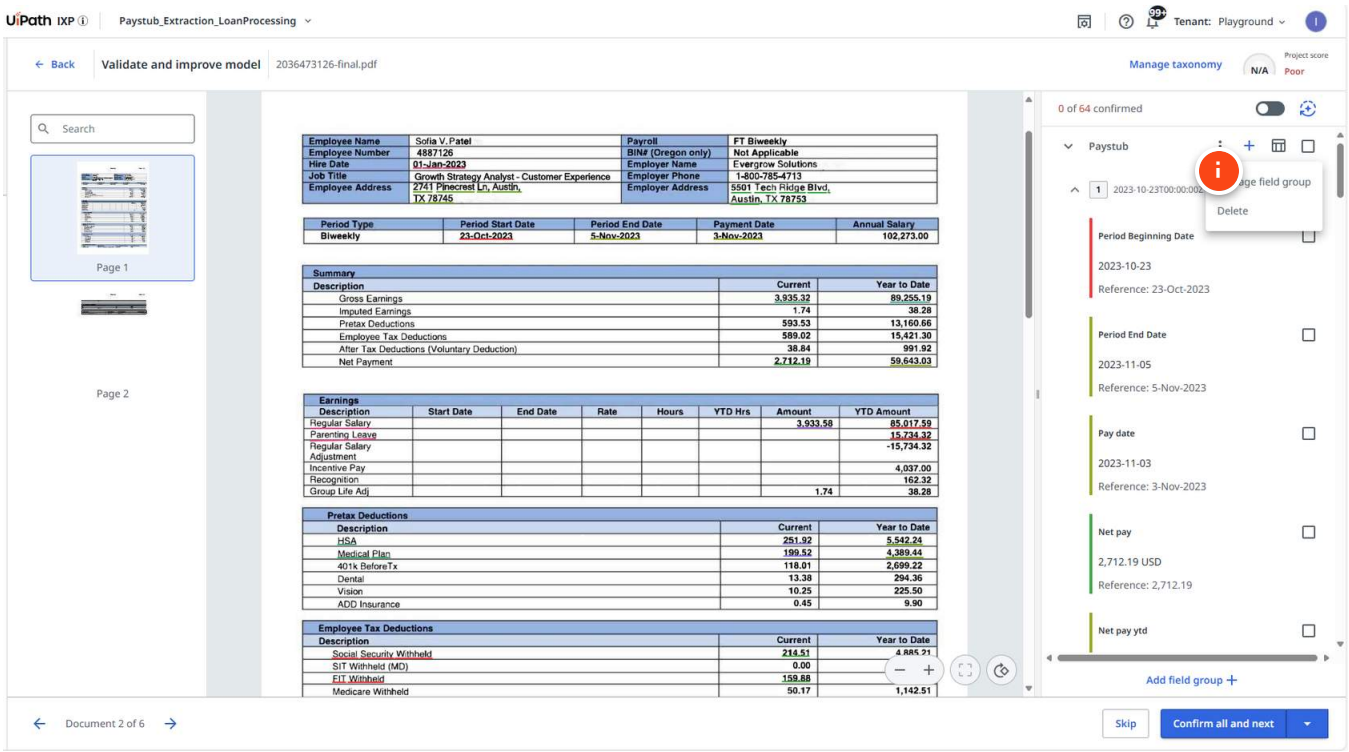
Filter to current page

Shows only the current page of the document - particularly useful when working with multi-page documents.



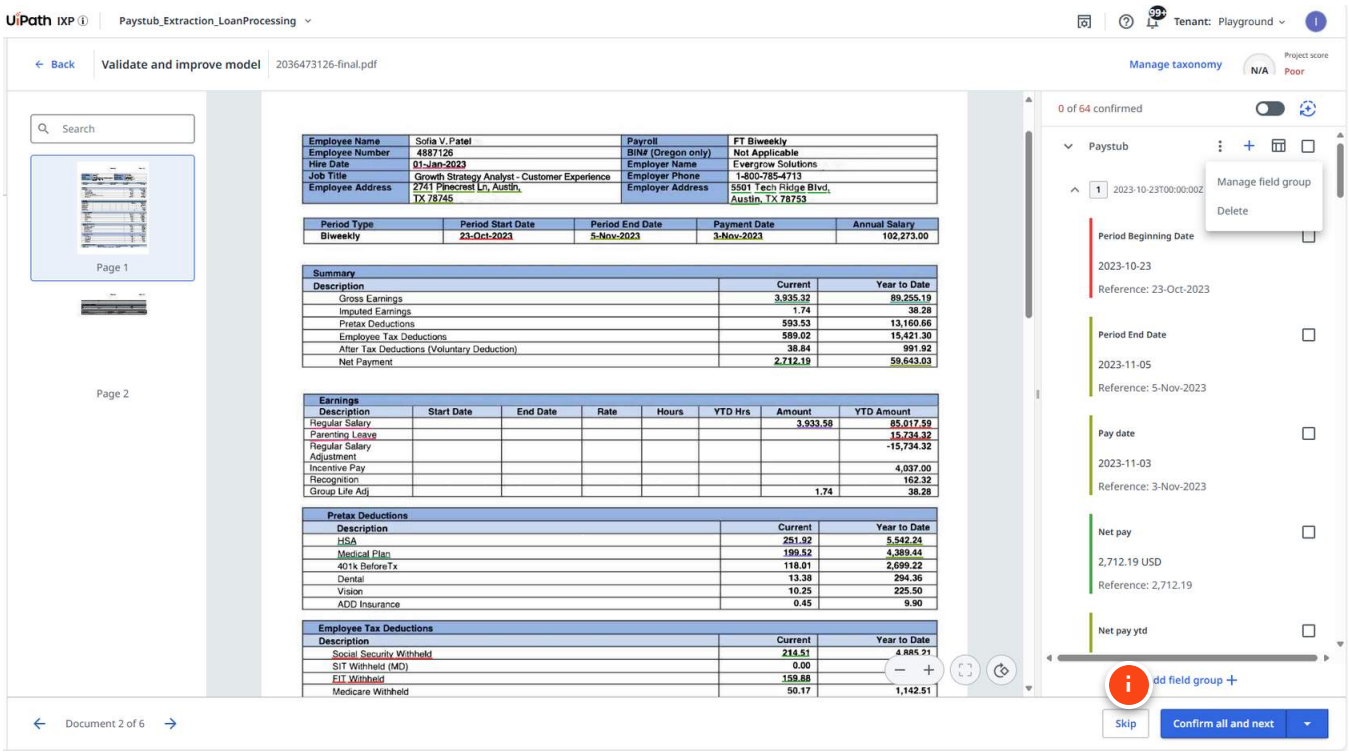
Confirm individual extraction

Confirm extraction for a specific field



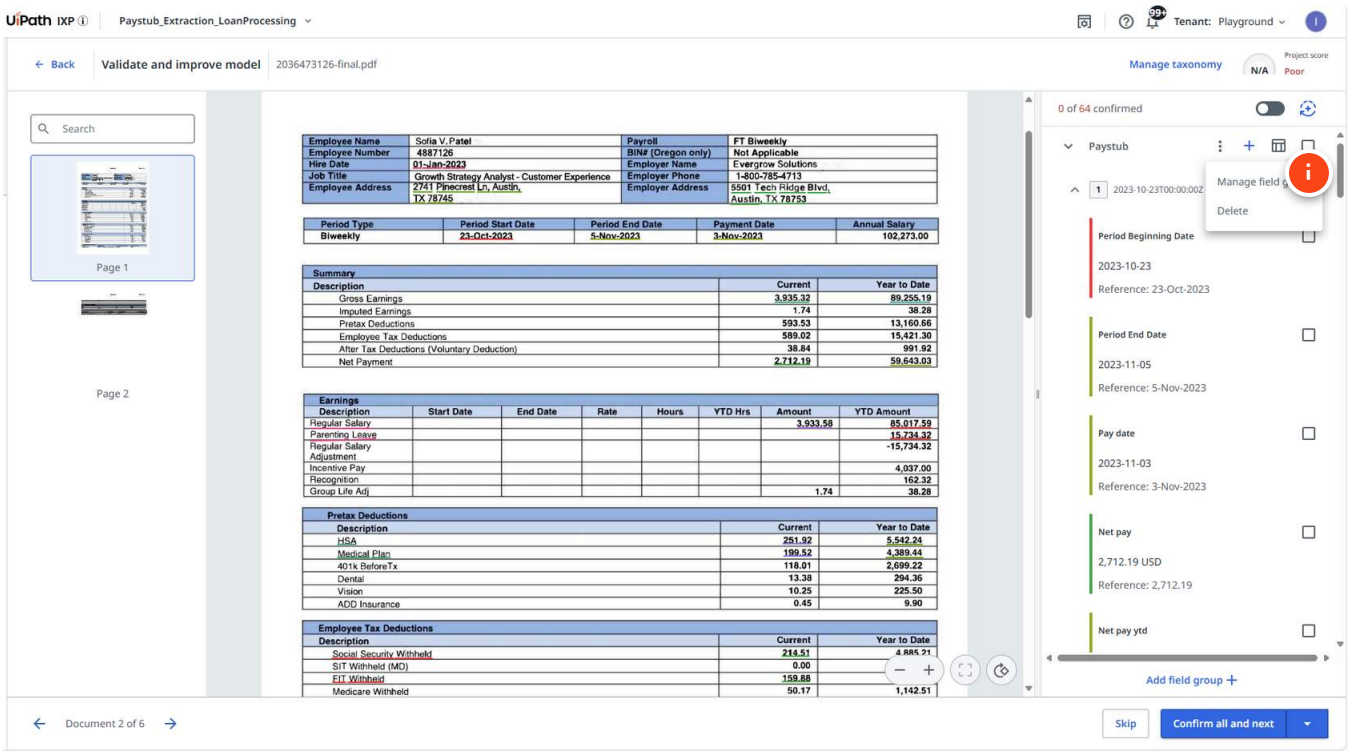
Manage field group

Manage the selected field group — edit the field group name and instruction, or delete it from the schema.



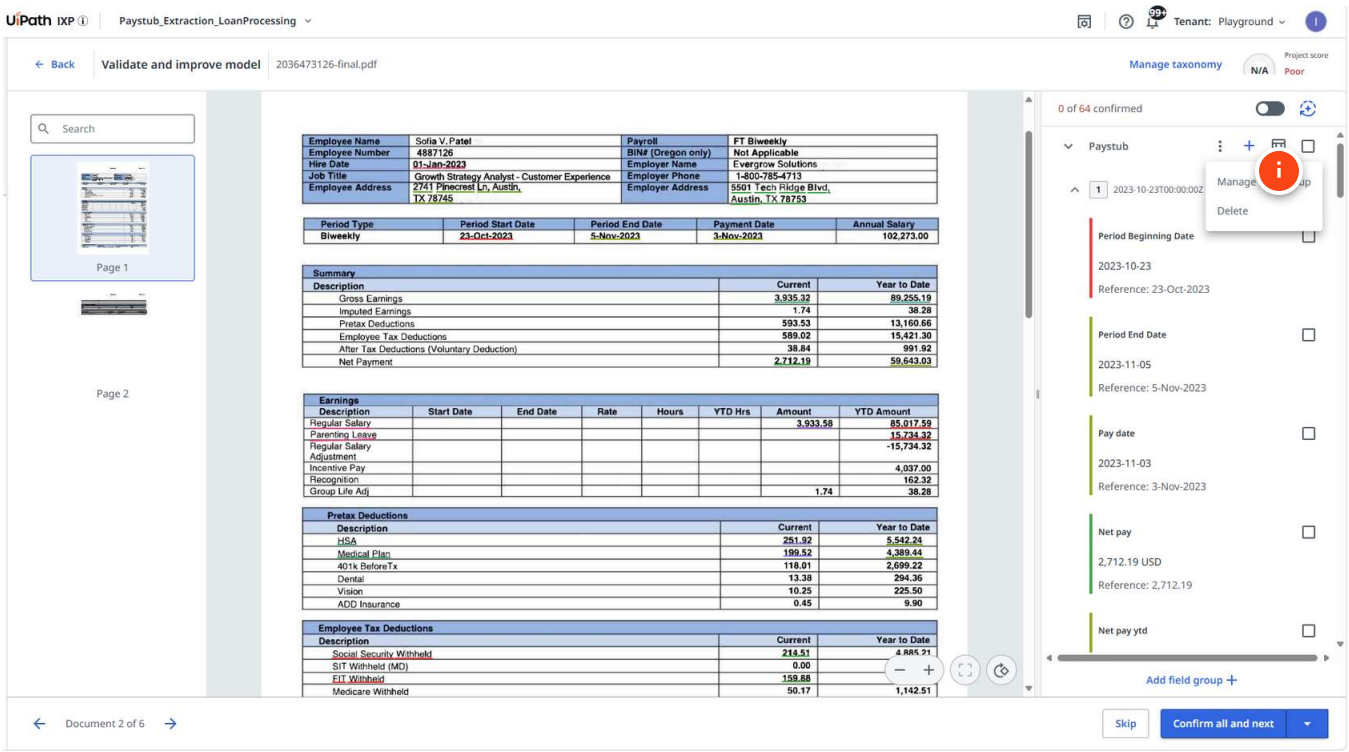
Add field group

Add a new group of related fields to your schema.



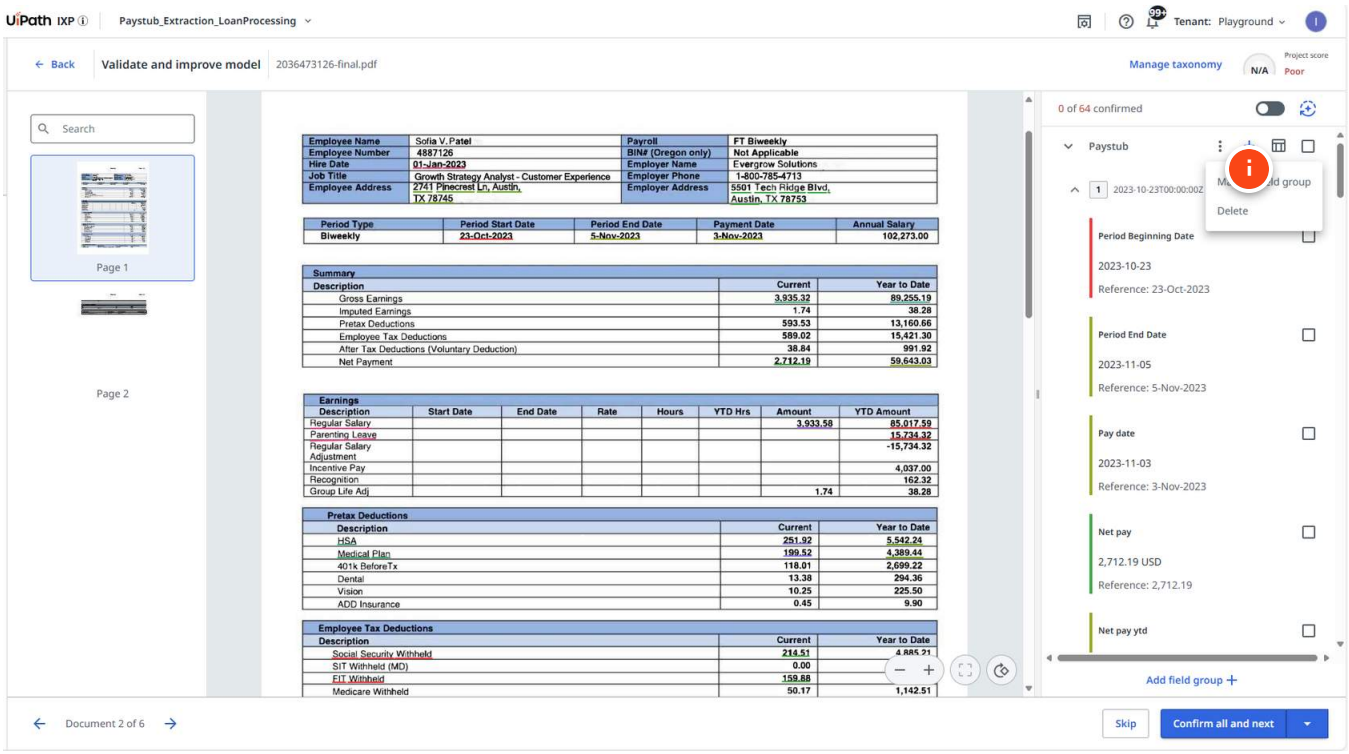
Confirm all extractions

Confirm all extractions for the entire document - make sure to review individual extractions before confirming them in bulk.



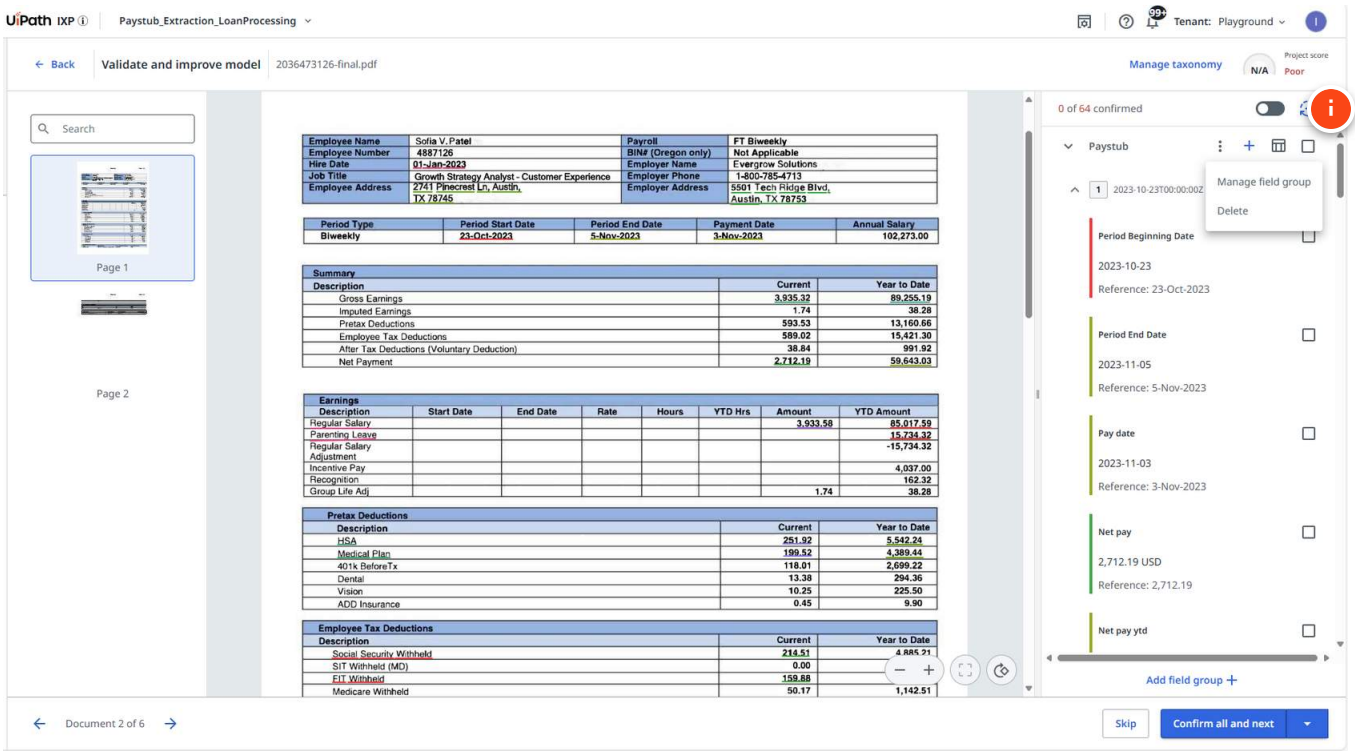
Display table view

Opens the table view, making it easier to review and validate all extractions in a specific field group — especially helpful for tabular data.



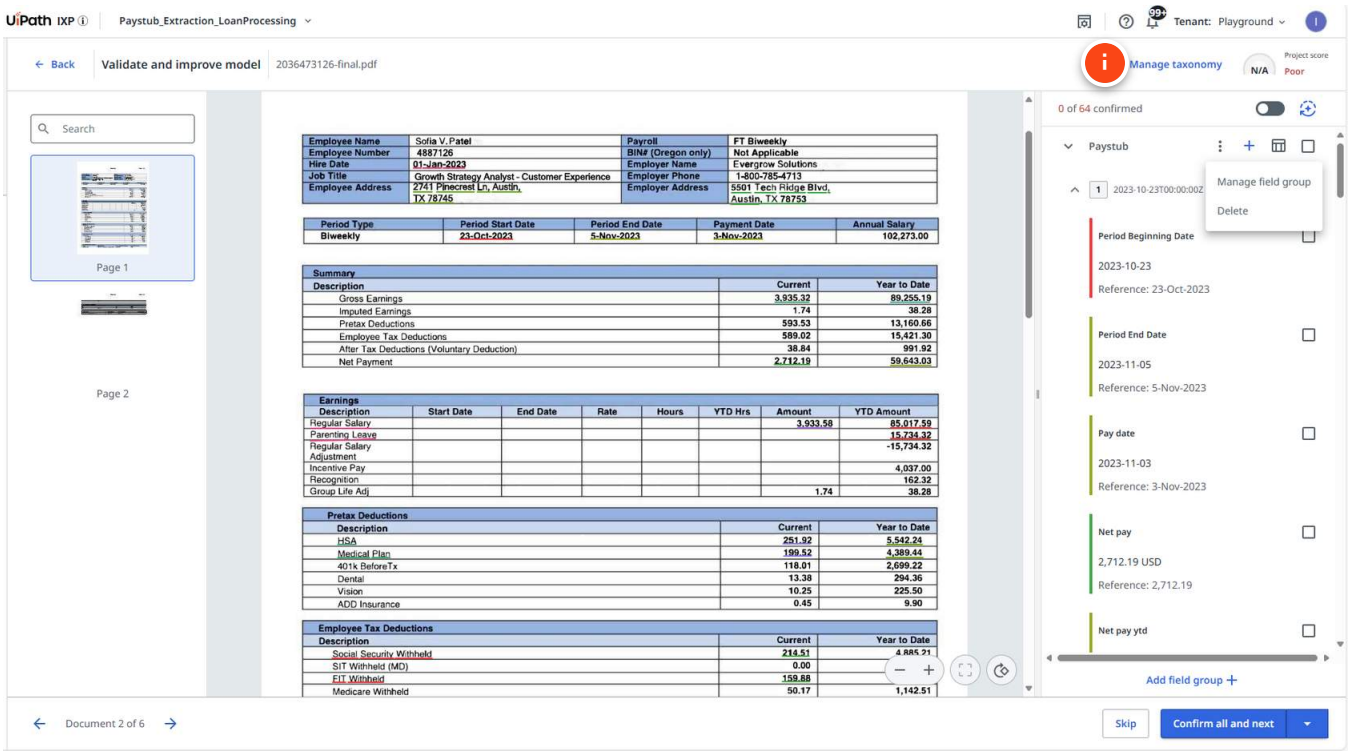
Add new extraction

Manually add a new extraction if missed by the model.



Predict extractions

Triggers the model to generate or refresh all predictions, when not generated automatically - this might be required after a schema change.



Manage taxonomy

Opens the taxonomy view, where you can modify the entire schema.

Why does fine-tuning instructions matter?

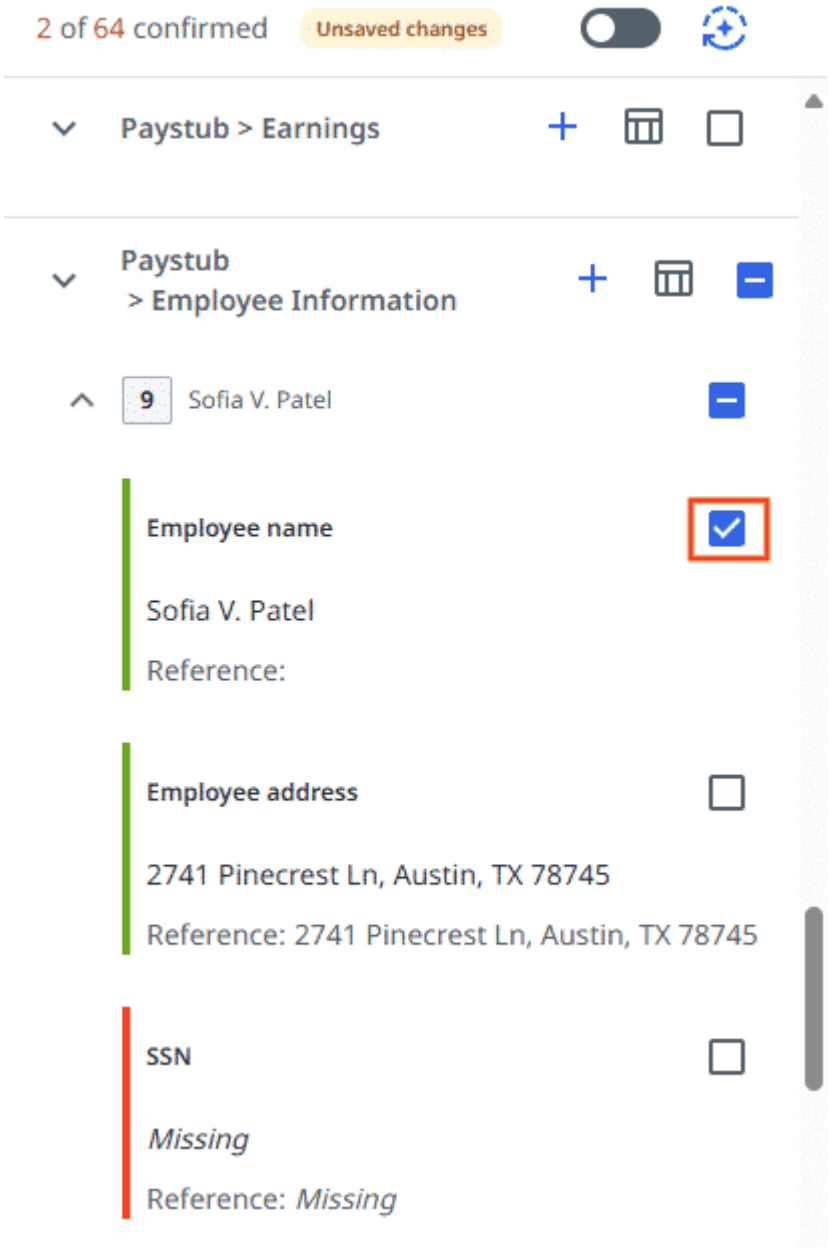
The only way to currently improve the model's performance is to **modify your prompt instructions or model configuration** (e.g., enabling intelligent pre-processing/table model).

The model **does not learn from user annotations**. However, it is still important to annotate documents to gather ground truth for validation.

How do you validate predictions?

Confirm extractions

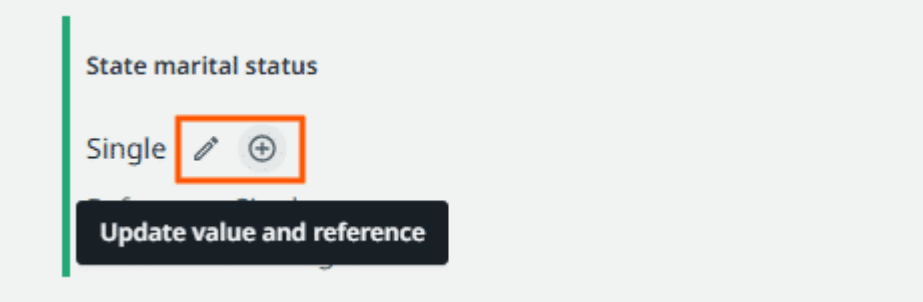
Confirm extractions if they are correct.



Correct extractions

Correct extractions if they are incorrect by either:

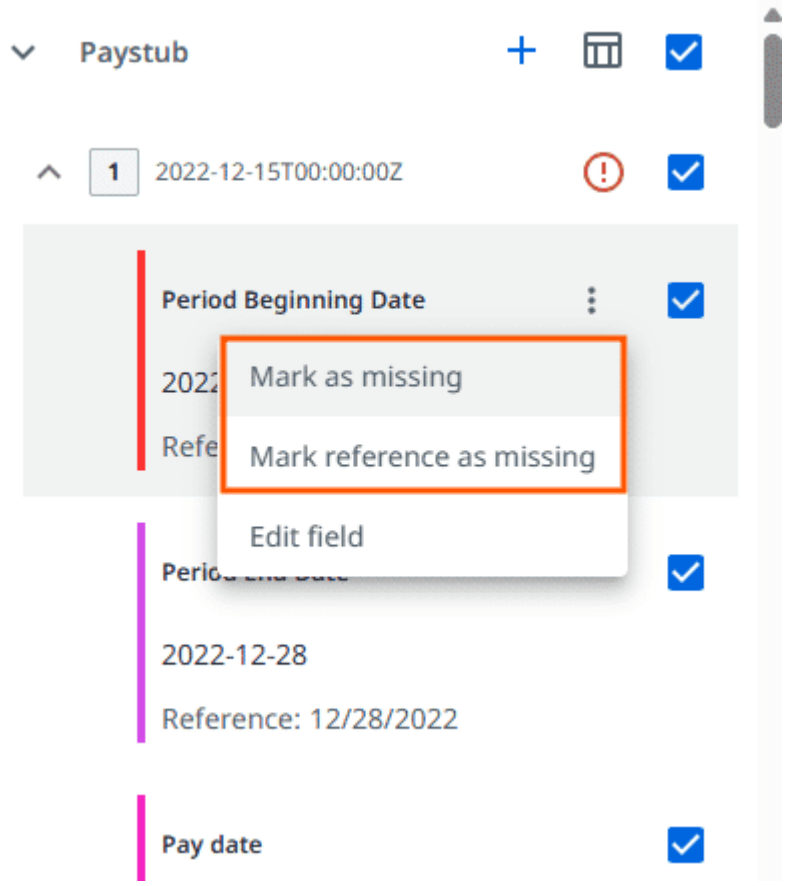
- Clicking on the pencil icon to **type in** the correct value.
- Clicking on the + icon to **highlight** the correct value and reference in the document.



Mark the field or reference as missing

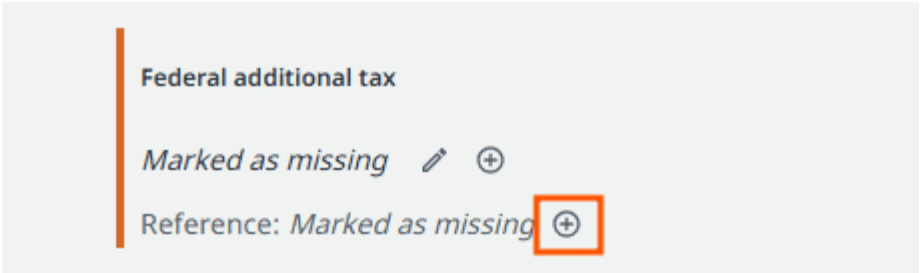
Mark the field as missing if the field is not present in the document.

Mark the reference as missing if the field does not have a matching reference (this is common for inferred fields).



Update references

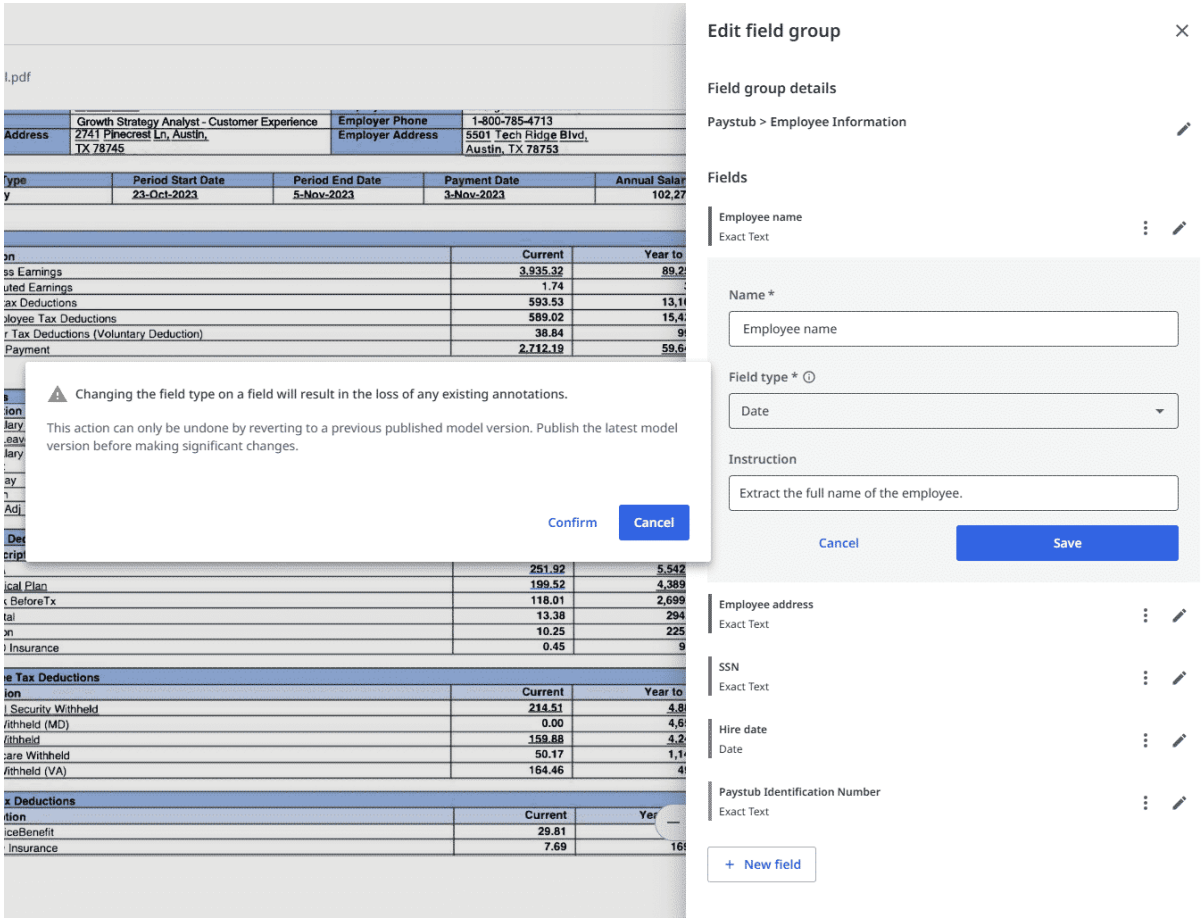
Update the reference if it's missing or is inaccurate.



Modify fields

Modify the field's name, type, or instruction to refresh the prediction. This is the easiest and fastest way to change your prompt instructions.

Remember that changing the field's type will remove any existing annotations.



Predictions and references are independent. Only the predictions (not the references) are compared to ground truth.

References are the matching references to the predicted values found in the document. Incorrect or missing references do not impact the model score or any performance metrics. The purpose of adding/correcting references is to improve the human validation

experience. There are known limitations with references, which is one of the key priorities to improve.

When can you stop validating?

It is recommended to annotate at least **10 examples for each field** so that the model can provide you with meaningful validation statistics.

This is not applicable for this exercise as you only have 6 samples - annotate all 6 documents you uploaded.

Once you start annotating your documents, you will see one of the below scenarios:

Prediction is correct

When your annotation matches the model's prediction.

Prediciton is incorrect

When your annotation is different from the model's prediction - this is automatically flagged as a mismatch.

Prediction is missing —

When you added the annotation but the model did not predict the value - this is automatically flagged as a mismatch.

Want to find out more?

Use the resources below to learn more about this topic.

UiPath IXP - Official documentation

Learn more about validation in IXP Generative extraction.

GO TO DOCS

Next, we'll look at how to assess model performance and identify areas for improvement.

Measure model performance

Measure model performance

- **Duration:** 10 min

What you'll learn in this lesson

At the end of this lesson you should be able to:

1. Evaluate model performance and identify areas for improvement.

2 Determine if the model is production-ready.

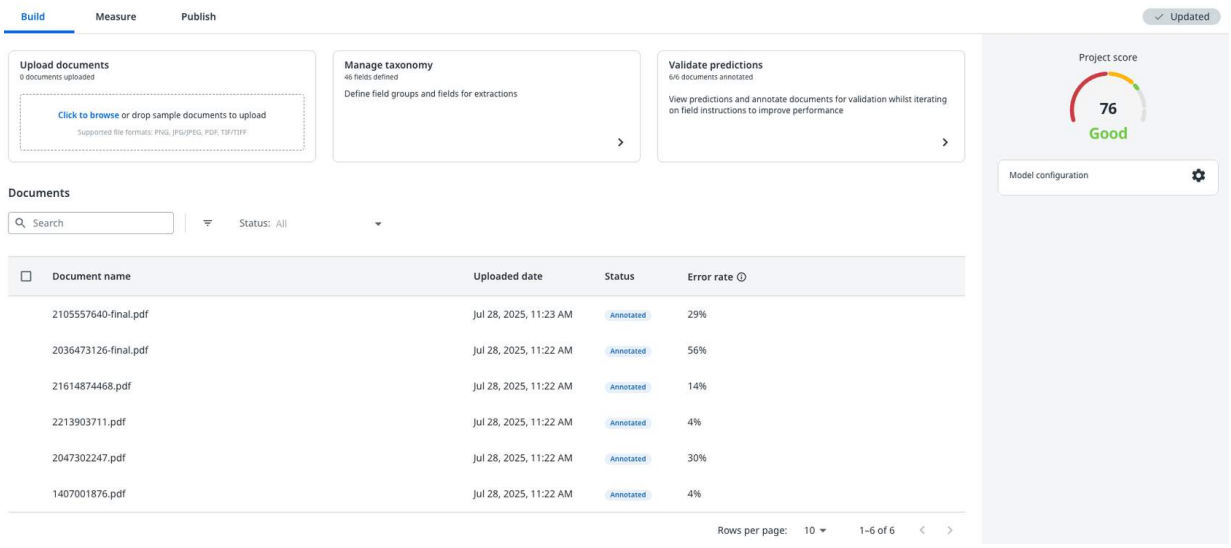
Once you fine-tuned your instructions and annotated sufficient number of documents, you can start evaluating the overall performance. You can assess the model's performance in the **Build tab** and **Measure tab**.

Understanding these metrics helps you determine if your project is **reliable, scalable, and ready for production deployment.**

Build tab performance

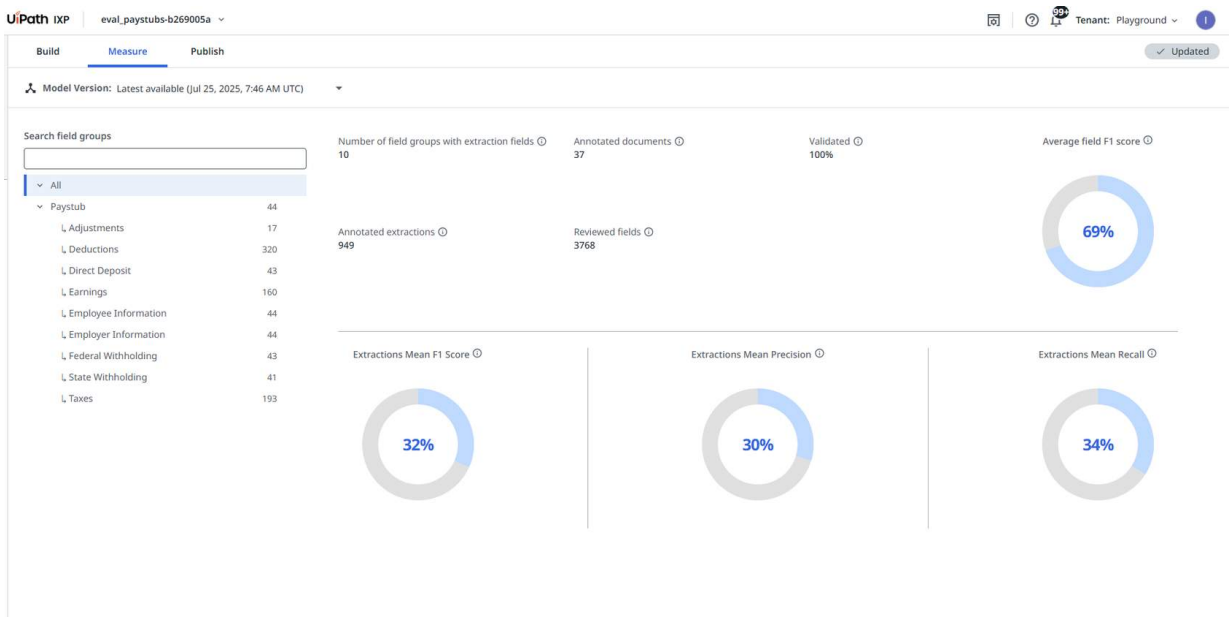
Build tab shows the overall project score, as well as the error rate for each document:

- **Project score** is calculated based on the weighted average of all field F1 scores. You want this score to be 'Good' or 'Excellent'.
- **Document error rate** indicates the number of mistakes the model made on each document (i.e., the difference between the model's predictions and your annotations). You want this number to be as low as possible.



Measure tab performance

Measure tab provides detailed performance metrics for all **fields and field groups**, including precision, recall, and F1 scores. Please note that the Measure tab will be blank if you haven't completed any annotations.



Precision: focuses on correctness of predictions and represents how many of the predicted values were actually correct.

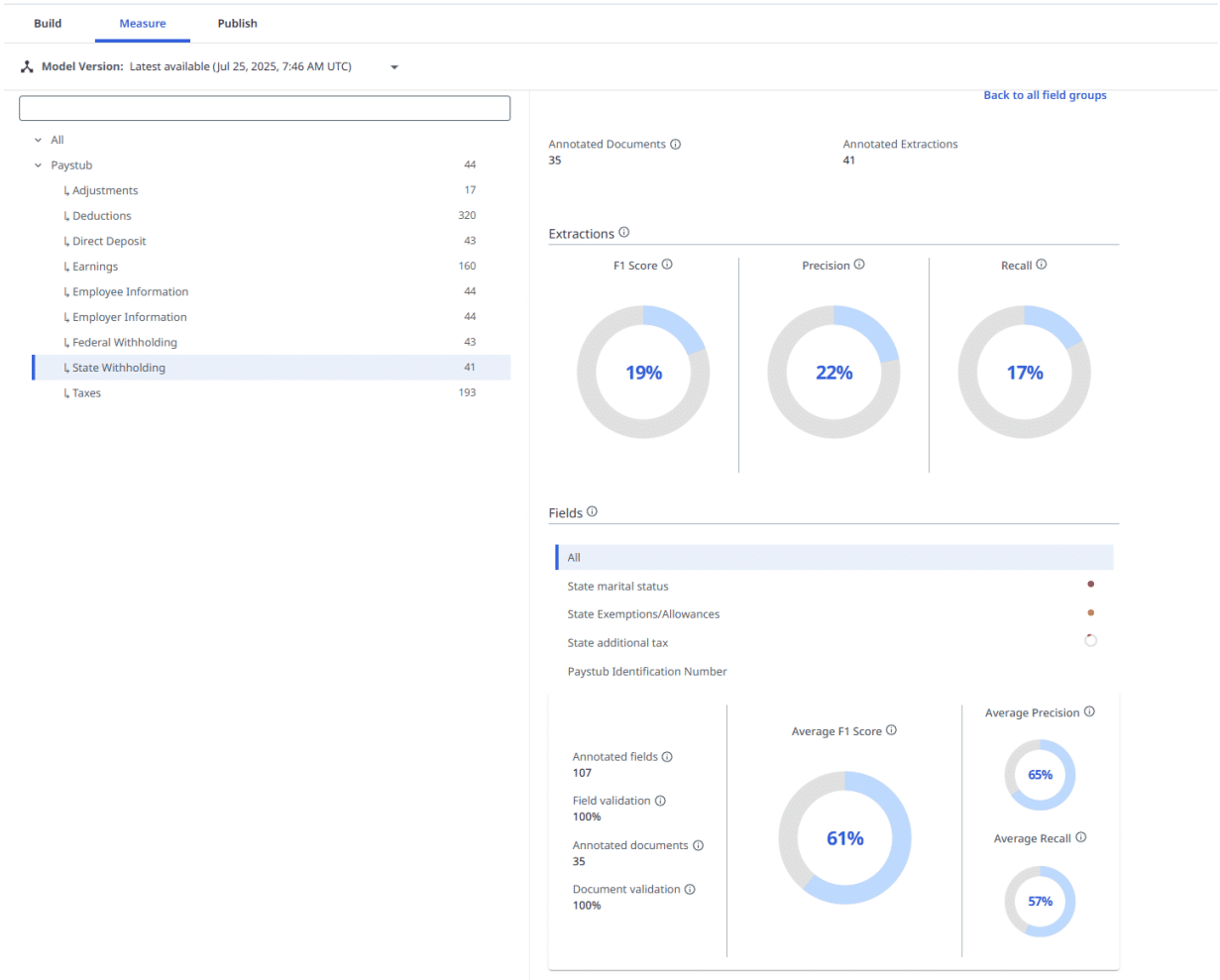
- Example: 95% precision would mean that out of 100 fields, 95 were correctly predicted and 5 were wrong.

Recall: focuses on completeness of predictions and represents how many of the correct values were actually predicted by the model.

- Example: 77% recall would mean that out of 100 fields, there were 23 fields which should have been predicted, but the model missed them.

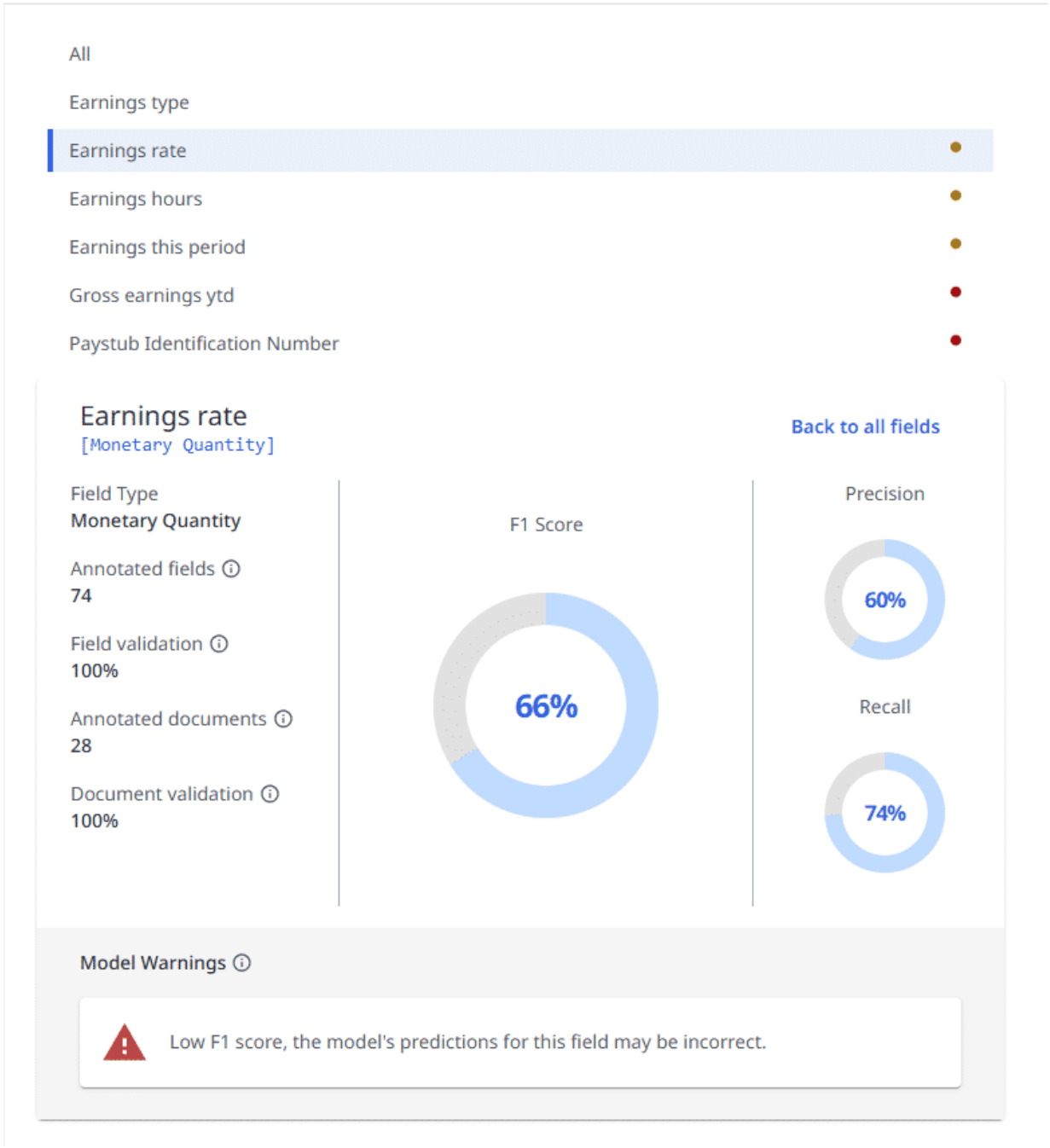
F1 score: represents the harmonic mean between precision and recall.

The field groups are displayed on the left-hand side. Click into each field group to display the field-level performance.



Some fields may show a **performance warning**. Performance warnings indicate potential performance issues and highlight areas for improvement.

- **Amber warning** — field performance is less than satisfactory and might require your attention
- **Red warning** — field performance is poor and requires your immediate attention
- **Red dial** — The F1 score is unreliable or unavailable due to the insufficient number of annotations. Each field requires at least **10 annotated examples** to provide a meaningful score



In addition to the these design-time metrics, IXP Generative extraction provides **confidence scores for each extraction in run-time**. Confidence scores are based on logprobs and are completely independent of precision, recall and F1 scores.

Improving performance

Annotated documents act as “ground truth” and are compared against model's predictions. **Only the predicted values are compared against annotations; not the references!**

You can take these targeted actions to improve the models' predictions:

- **Refine your prompt instructions:** Modify your field or field group instructions to be more explicit (see the *Prompting best practices* lesson for specific tips).
- **Adjust your overall schema:** Add or remove fields, change field types, or re-structure the field groups.
- **Update model configuration:** Tweak model parameters, such as enabling the table model for intelligent pre-processing of tabular data.
- **Re-validate and repeat:** Refresh predictions to see the impact of your schema/instructions/model configuration changes.

Example: Paystub use-case

Imagine the Measure tab shows the following:

Field	Precision	Recall	F1 Score
Employee Name	98%	95%	97%

Net Pay	85%	80%	83%
Tax Withheld	70%	60%	65%

- Low F1 score for the **Tax Withheld** field indicates that your prompt instruction may be unclear or not fit for all paystub layouts.
- You should fine-tune the instruction and add additional context or specific examples to clarify any nuances.
- You might want to consider uploading and validating additional paystub samples.
- Once done, re-run the predictions and asses the impact on the field performance.

Want to find out more?

Use the resources below to learn more about this topic.

UiPath IXP - Official documentation
Learn more about performance evaluation in IXP Generative extraction.

GO TO DOCS

Let’s see how to publish your project and configure its deployment.

Publish the project

Publish the project

- **Duration:** 5 min

What you'll learn in this lesson

At the end of this lesson you should be able to:

1. Publish a project for consumption in UiPath Studio.

2 Understand model versioning and how published projects link to automation workflows.

Publishing your project is the final step to make your model available for use in downstream automation workflows. Publishing **saves your schema and configuration into a versioned package**.

Published projects can then be consumed via UiPath Studio (Desktop or Web) using UiPath Document Understanding activities.

What does it mean to publish a project?

Publishing is the process of creating a **snapshot** of your extraction project. This snapshot includes your:

- **Schema** (field groups, fields, and all instructions)
- **Model configuration** (e.g., selected LLM, intelligent pre-processing, and other advanced settings)

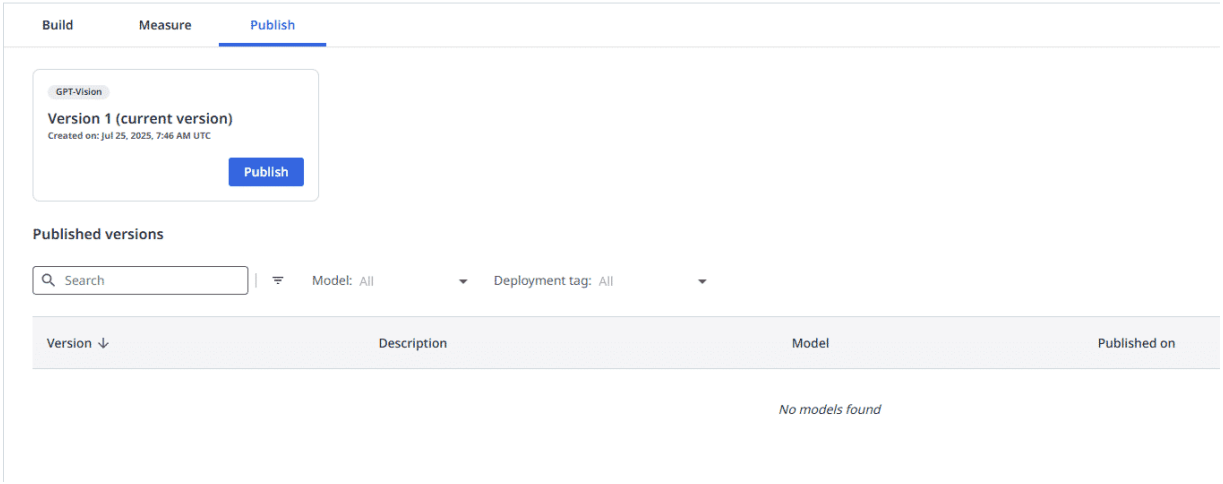
Publishing a project makes it **available for consumption** by automation workflows. It also allows you to:

- Compare different model versions
- Rollback to a previous model version
- Update your model without impacting the automation

How to publish your project?

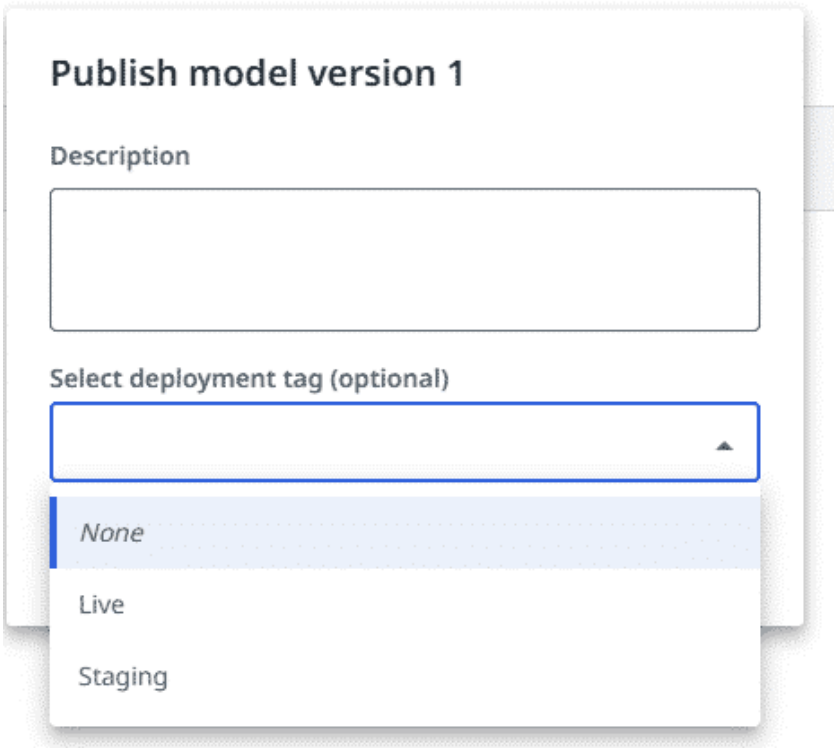
Follow the below steps:

Step 1



Navigate to the Publish tab and click on the Publish button.

Step 2



Add optional **Description:**

- Description can refer to specific changes or improvements (e.g., “Initial publish for loan processing use case, validated on 100 paystub samples”).

Add optional **Deployment tags:**

- Staging - indicates that the published version is intended for testing or pre-production deployment.
- Live - indicates that the published version is intended for production deployment.

Version management

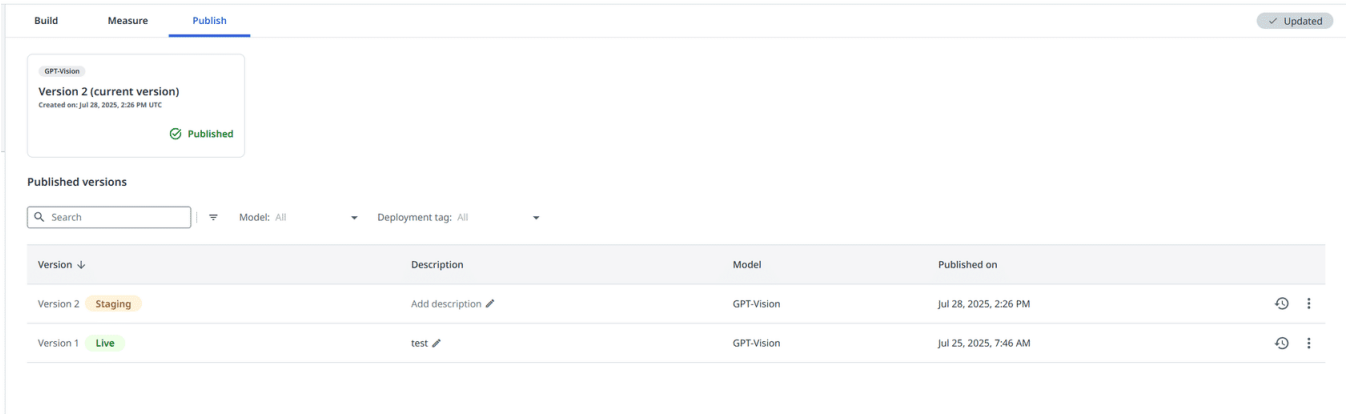
After publishing, you will see the newly created version in the version list, including:

- Version number/name with its associated deployment tag
- Publish date and time
- Description

You can:

- Revert to an earlier version
- Unpublish a certain version
- Change the deployment tag

IXP Generative extraction provides **robust governance** for enterprise customers, helping them to maintain the necessary control and traceability.



Want to find out more?

Use the resources below to learn more about this topic.

UiPath IXP - Official documentation

Learn more about model deployment in IXP Generative extraction.

GO TO DOCS

Next, you'll learn how to consume IXP Generative extraction project in UiPath Studio.

Consume the project

Consume the project

- **Duration:** 20 min

What you'll learn in this lesson

At the end of this lesson you should be able to:

1. Understand how to use IXP Generative extraction project in UiPath Studio.

2. Configure Document Understanding activities to connect to an IXP Generative extraction project.

3. Use the extracted data in an automation workflow.

Once your project is published, the next step is to use it inside UiPath Studio. This allows your automation to extract structured data from unstructured documents—such as paystubs—in runtime.

In the following lesson, we'll walk through how to consume a published IXP Generative extraction project using both **UiPath Studio Web** and **UiPath Studio Desktop**. You'll learn how to load the model, configure the necessary activities, and integrate the extracted data into your automation workflows—whether you're building in a lightweight browser-based environment or a full-featured desktop setup.

Consume IXP Generative extraction project in UiPath Studio Desktop

In the following steps, you'll learn how to consume a published project within **UiPath Studio Desktop**. We'll explore how to integrate the model using both **UiPath.DocumentUnderstanding.Activities** and the classic **UiPath.IntelligentOCR.Activities**, depending on your project setup and preferences.

Consume IXP Generative extraction project with IntelligentOCR

Before you begin your implementation, ensure that the **UiPath.IntelligentOCR.Activities** package (v6.23.2-preview or newer) is installed in your project.

When using **UiPath.IntelligentOCR.Activities** to consume an IXP Generative extraction project in Studio Desktop, you'll follow the standard Document Understanding workflow. This involves setting up the taxonomy, digitizing the documents, extracting the data using your published project, and validating and exporting the results (optional).

Below is a step-by-step breakdown:

1. ## Create taxonomy

Although the extraction logic is managed in your published IXP Generative extraction project, **UiPath.IntelligentOCR** still requires a local taxonomy file to define the document structure. This step ensures your workflow can interpret and map the extracted data correctly.

- Use the **Taxonomy Manager** in Studio to create a new taxonomy.
- Recreate the structure from your IXP Generative extraction project by defining the **Field Groups** and **Fields**.

🧩 In IXP Generative extraction, each **field group** is treated as a standalone **table**, where each **field** represents a different **column**. You should mirror this structure in your Taxonomy Manager—**create a table per each field group, and define each field as a column within that table.**

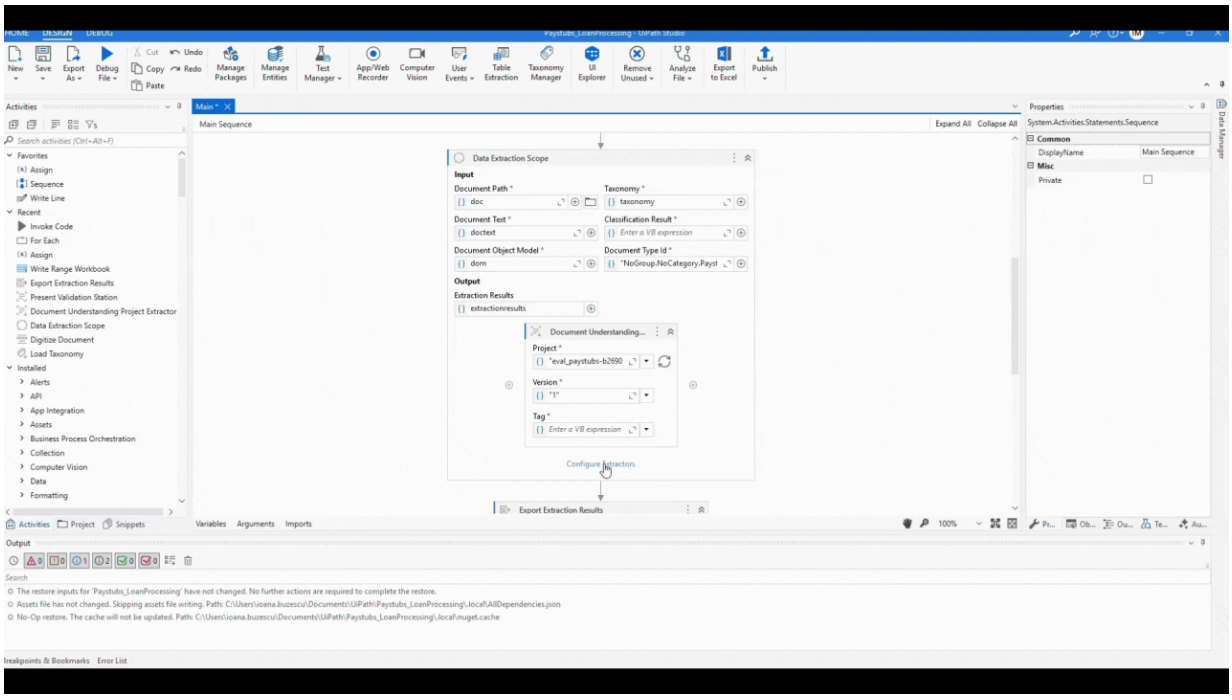
This setup allows the output from your IXP Generative extraction model to align seamlessly with the Document Understanding framework in Studio. More information on how to create the taxonomy can be found here.

If you do not want to create the taxonomy manually, you can simply replace the existing taxonomy.json file in your **DocumentProcessing** folder with the preconfigured version provided below.

2. The next steps involve loading the taxonomy, digitizing your documents, classifying them (optional), and then using the **Data Extraction Scope** to extract the relevant information.

   Configure the **Data Extraction Scope** by setting its input properties. In the **Extractors** section, add a **Document Understanding Project Extractor**. When prompted by the **Get Capabilities** window, search for your published IXP Generative extraction project and select the appropriate version.
   After loading the project, proceed to map the fields defined in your local taxonomy to ensure proper alignment between your workflow and your IXP schema.

   You can watch the gif below to help you understand the field mapping.

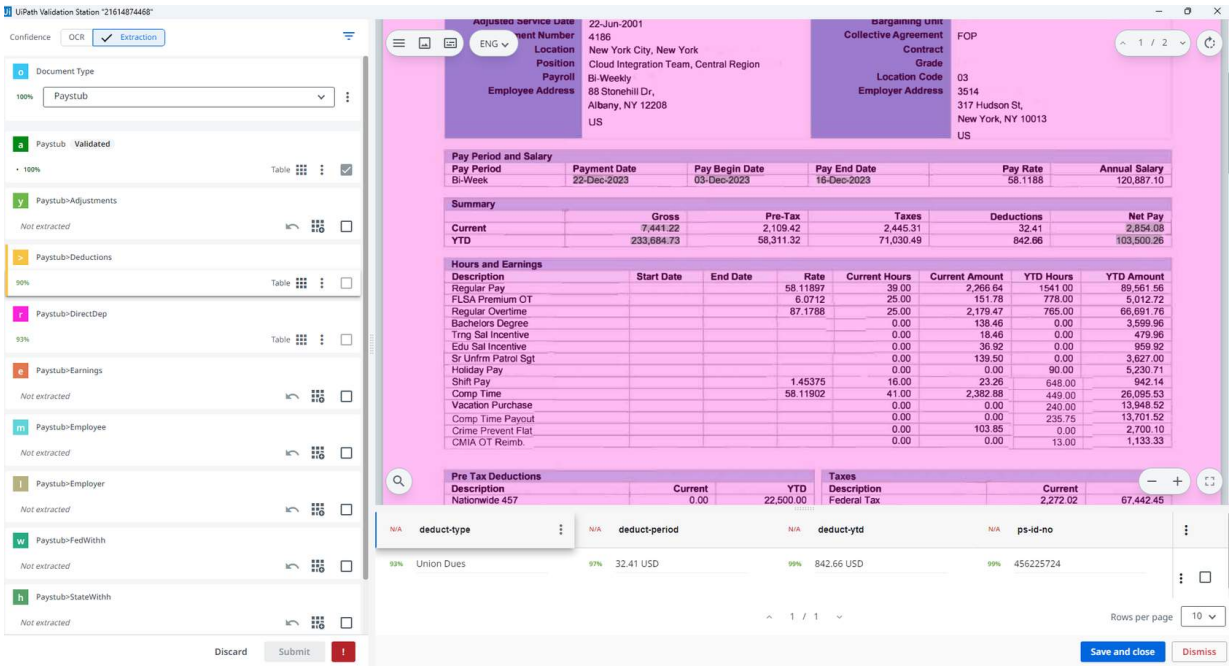


3. **Add Present Validation Station (optional but recommended)**

To verify that the data extracted correctly, insert a **Present Validation Station** activity after the **Data Extraction Scope**. This activity allows a human user to review and correct the extracted values before continuing the automation.

📌 Make sure to pass the same taxonomy and document object used earlier in the workflow. You can also configure it to only trigger validation for **low-confidence extractions**.

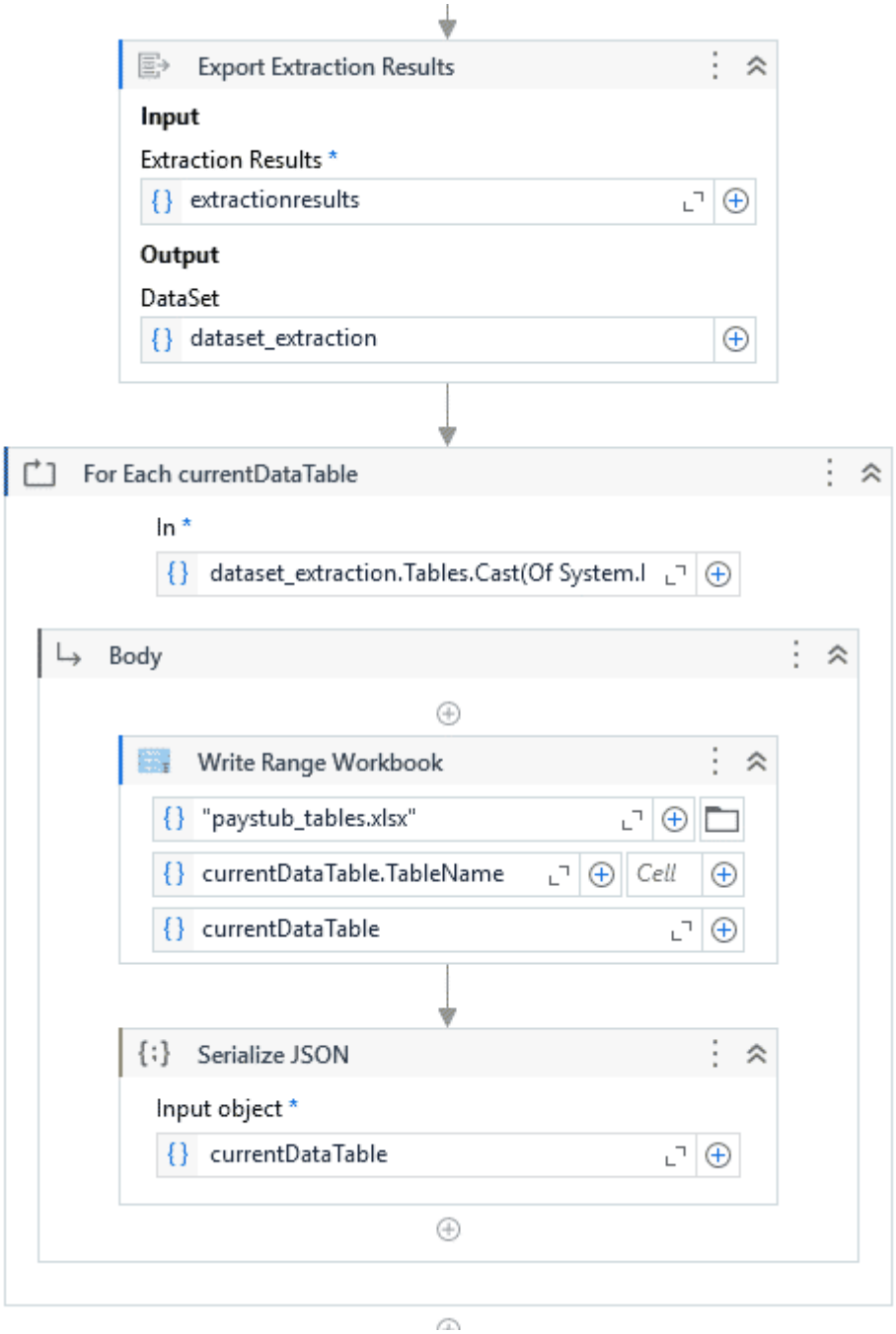
Once you run the workflow, you will notice that all the fields were extracted correctly.



4. Now that we've confirmed the extractions, it's time to **format the results**. You can choose to export the data as a CSV file or format it as a JSON string, depending on your needs.

   Drag the **Export Extraction Results** activity into the workflow canvas to iterate through the extracted tables.

- If you want to output the results to an Excel file, follow it with a **Write Range Workbook** activity and configure the destination accordingly.
- Alternatively, if you prefer the results as a JSON string, use the **Serialize JSON** activity and pass the current DataTable as input.



And that's it! Now you have the extraction results in a structured, readable format—**ready for any post-processing**, such as data validation, storage, or integration with other systems.

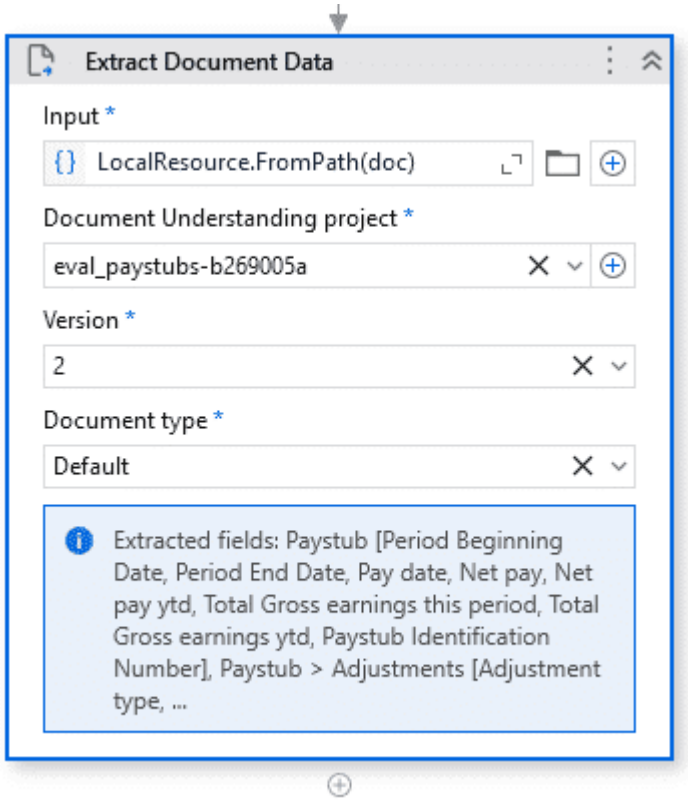
CONTINUE

Consume IXP Generative extraction project with DocumentUnderstanding

Before you begin your implementation, ensure that the **UiPath.DocumentUnderstanding.Activities** package (v2.13.2-preview or newer) is installed in your project.

When using **UiPath.DocumentUnderstanding.Activities** to consume an IXP Generative extraction project, the process is streamlined compared to traditional IntelligentOCR workflows. You no longer need to define or manage a local taxonomy—everything is already handled within your published project. Your focus shifts to orchestrating the workflow: classifying incoming documents (optional), extracting data using your IXP model, and then handling any post-processing based on the extracted results.

1. Start by dragging the **Extract Document Data** activity onto the canvas and configuring it. Simply provide the input document and select your published IXP Generative extraction project. The activity will automatically retrieve the schema from the project—no manual taxonomy setup or field mapping is needed.



2. Now let's export the extraction results as a JSON string. You can do this using a simple **Assign** activity. In the value field, use the following expression:

   > *JSONVariable = Newtonsoft.Json.JsonConvert.SerializeObject(extractionResults)*

   This will convert the extractionResults object into a JSON-formatted string, making it easier to store, transfer, or use in any downstream processing.

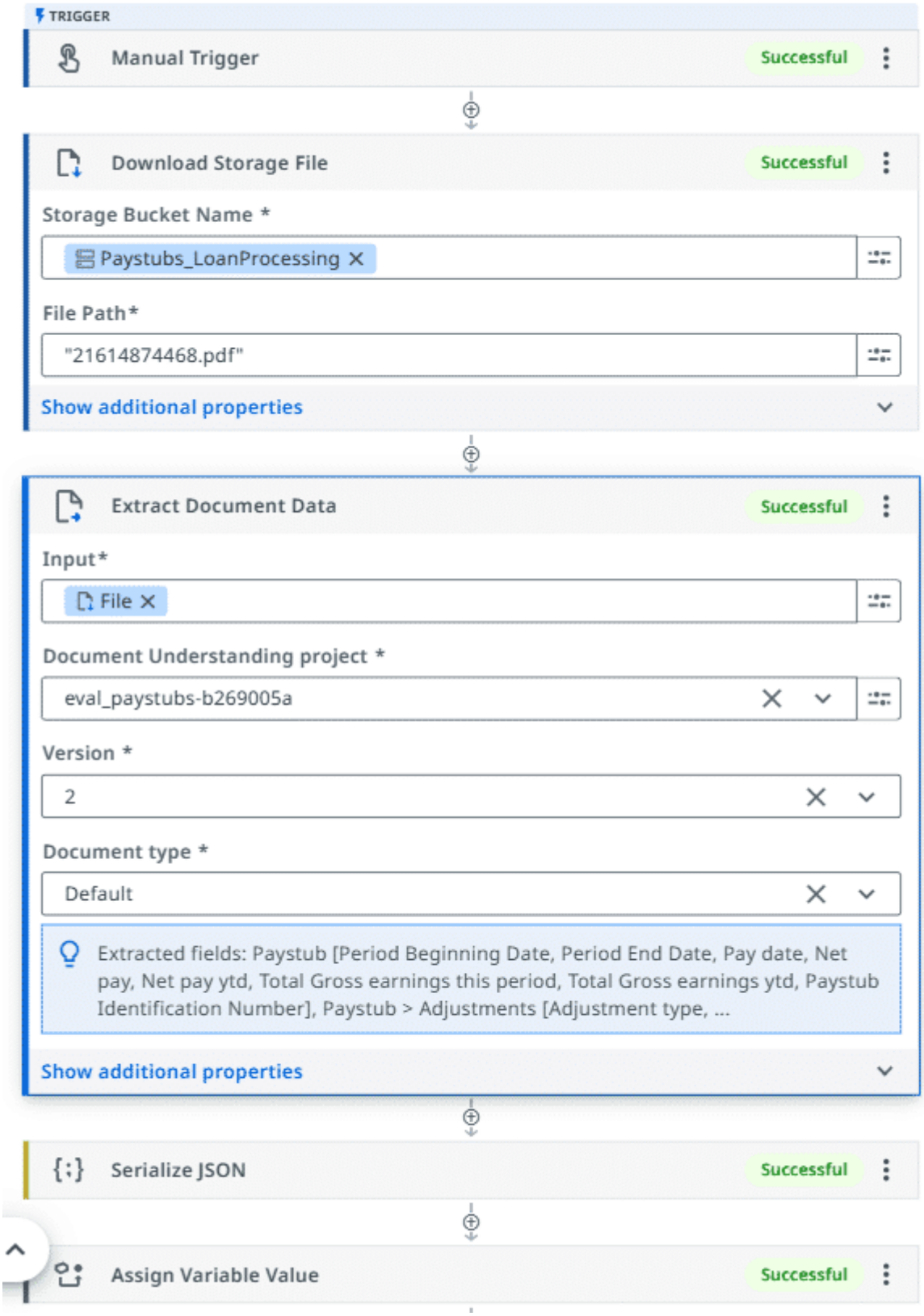
CONTINUE

Consume IXP Generative extraction project in UiPath Studio Web

You can also consume your IXP Generative extraction projects directly in UiPath Studio Web, if preferred.

Below is a step-by-step breakdown:

1. **Store documents -** Ensure your documents are stored in a suitable location. For this example, we will use an Orchestrator Storage Bucket.
2. **Create workflow -** Open UiPath Studio Web and create a new RPA workflow.
3. **Download documents**
    - Drag the **Download Storage File** activity onto the canvas.
    - Configure it to download the required documents from the Storage Bucket.
4. **Extract data**
    - Add the **Extract Document Data** activity to the workflow.
    - Select your IXP Generative extraction project to extract the data.
5. **Serialize results**
    - Convert the extraction results into a JSON string using serialization.
    - This prepares the data for any further post-processing.


TRIGGER
Manual Trigger
Successful
Download Storage File
Successful
Storage Bucket Name *
Paystubs_LoanProcessing
File Path*
"21614874468.pdf"
Show additional properties
Extract Document Data
Successful
Input*
File
Document Understanding project *
eval_paystubs-b269005a
Version *
2
Document type *
Default
Extracted fields: Paystub [Period Beginning Date, Period End Date, Pay date, Net pay, Net pay ytd, Total Gross earnings this period, Total Gross earnings ytd, Paystub Identification Number], Paystub > Adjustments [Adjustment type, ...
Show additional properties
Serialize JSON
Successful
Assign Variable Value
Successful

Resources

Below are the workflows from UiPath Studio Desktop, using both IntelligentOCR and DocumentUnderstanding activities, and Studio Web, as well as the samples.

1 **UiPath Studio Desktop workflows**

ZIP **Paystubs_LoanProcessing.zip**
215.9 KB

ZIP **Paystubs_LP_DU.zip**
189.5 KB

2 **UiPath Studio Web workflow**

FILE **Paystubs_LoanProcessing.uis**
28.6 KB

3 **Paystub samples**

ZIP **samples.zip**
4 MB

Want to find out more?

Use the resources below to learn more about this topic.

UiPath IXP - Official documentation
Learn more about model deployment in IXP Generative extraction.

GO TO DOCS

UiPath Studio Web - Official documentation
Learn more about Studio Web.

GO TO DOCS

UiPath IntelligentOCR activities - Official documentation

Learn more about IntelligentOCR activities.

GO TO DOCS

UiPath DocumentUnderstanding activities - Official documentation
Learn more about DocumentUnderstanding activities.

GO TO DOCS

Next, let's have a look at IXP prompting best practices.

Prompting best practices

Prompting best practices

- **Duration:** 10 min

What you'll learn in this lesson

At the end of this lesson you should be able to:

1. Implement general best practices when designing IXP taxonomy for unstructured and complex documents.

2 Apply specific recommendations for defining instructions on the project level, field group and field levels.

3 Understand the importance of testing and iteration to optimize model performance.

General best practices

Clarity and Simplicity

Use clear, direct, and unambiguous language. Avoid overcomplicating instructions that could confuse the model. Use plain language and keep sentences short.

Consistency

Maintain consistent terminology across fields, field groups, and instructions to avoid confusion.

Provide context

Equip the model with pertinent context to comprehend the general scope of the task. This could encompass industry information, document type, or overall data format - the model needs to understand its task. Providing more context within the prompt increases the probability of the model consistently predicting the field correctly.

Iterate

As refining prompts is an iterative process, maintaining a record of your drafts and their corresponding results can provide valuable insights for future adjustments and improvements. Write a prompt, test, edit. Repeat this process until you get your desired extraction.

Avoid negative instructions

Do not say things like: "Don’t leave out any sections of the document."
Instead, replace the previous statement with "Ensure all key sections (x,y,z) of the document are covered.

Avoid repetitive language

Repetitive language can lead to redundancy, confusion, and unclear instructions for the model.

Positive reinforcement

Whenever feasible, reinforce the prompt instruction with examples of correct responses.These instances can guide the model towards the expected outcome.

CONTINUE

Best practices by section

Project level

At the project level, instructions should establish clear context and expectations for extraction. Start by **defining the industry and document type**, which helps the model understand the scope of the task. Describe the key characteristics and typical layout of the document to guide the extraction.

If a document contains **multiple instances of the same data** (e.g., several pay periods) or **multiple document types** (e.g., paystubs + ID + employment contract) indicate this in the overall project instruction and specify how it should be extracted.

Field group level

Group similar data points into logical field groups to enhance extraction accuracy and minimize errors. Clearly explain each field group's purpose in the overall document.

Where applicable, **indicate where the data is typically located**—such as in tables, headers, or body —to help guide the model to the correct section. **Treat each field group as a table,** where each field represents a distinct column, and end all data points are extracted as a separate row/line item. This approach is key to effective data modelling in IXP.

Use field groups to **define relationships between data**, using a **">" symbol** to indicate parent-child hierarchy where needed (e.g., Paystub > Deductions; Paystub > Adjustments).

For files that contain **multiple documents** (e.g., multiple paystubs), define a **unique identifier** (e.g., Paystub ID) and include it as a field in every field group. This will ensure that all relevant data points are linked correctly and can be processed downstream.

Field level

Select **clear and intuitive field names** that use similar language as the documents to reduce ambiguity during the extraction. Field names should be specific enough to distinguish them from other similar data points.

Be explicit about what the model should extract ("extract XYZ"), followed by a specific requirement or formatting instruction. Avoid combining multiple requirements into a single instruction—**each field should focus on one main extraction goal** to maximize accuracy.

Whenever possible, **include example inputs and corresponding expected outputs** in your instruction. These examples help set a reference for the model, especially when handling complex or nuanced information.

Field type level

Choosing the correct field type is essential for accurate data formatting and seamless post-processing. Field types **determine how the extracted data is formatted and**

normalized, so they should be selected based on the downstream automation requirements.

IXP generative extraction provides several **predefined field types**. Use predefined field types for all generic fields of a certain type (e.g., monetary quantities) that **do not require unique formatting and can be applied across multiple fields**. For example, an instruction for Monetary Quantity fields might specify that all values should be extracted in USD.

Name	Instruction	Data type
Boolean	Use to represent boolean values in text.	Boolean
Date	Use to represent dates in text.	Date
Exact Text	Use to represent text that appears verbatim in text.	String Exact match in the document
Inferred Text	Use for text that might not appear verbatim in the text.	String Can be inferred based on document context
Monetary Quantity	Use to represent monetary values in text.	Monetary Quantity
Number	Use to represent amounts or quantities in text.	Number

If you require a unique field or formatting, **create a custom field type**. In such cases, clearly specify the desired data structure and format in your field type instruction. Custom field types will be required for certain inferred fields (e.g., choice).

You can find additional recommendations including **'good' and 'bad' instruction examples** in the attached **IXP prompting best practices** guide.

CONTINUE

Testing and iterating process

Start by creating all necessary fields **without adding any instructions.** Use 2–3 sample documents that represent the expected variation in layout and content. Run the initial predictions and evaluate whether they are aligned with the expected outputs.

For fields with inaccurate or incomplete results, draft your instructions and re-run the predictions to assess any improvements. **Refine the prompts iteratively** until the results are consistently correct.

Once you're happy with the performance on a smaller batch, **incrementally expand your sample set** (e.g., test on 10, then 20, then 30+ documents) to validate predictions across a broader range of documents. Ensure to review the predictions on all documents every time you change your prompt instructions.

If the performance is generally poor, consider modifying the field group level instructions. For any targeted improvements, focus on the field level refinement.

After sufficient validation and iteration, **annotate at least 10 documents** to get reliable performance metrics in the Measure tab. The Measure tab provides field group and field level metrics like precision, recall, and F1-scores. These metrics are essential for **gathering ground truth** and identifying performance improvements.

Downloadable resource

Download the IXP prompting best practices guide to create your extraction schema and prompt instructions.

PDF

IXP Prompt Best Practices April 2025.pdf

413 KB

Next, let's explore the available resources.

Learning resources

Learning resources

Official documentation

UiPath IXP - Official documentation
Learn more about IXP Generative extraction.

GO TO DOCS

UiPath DocumentUnderstanding activities - Official documentation

Learn more about DocumentUnderstanding activities.

GO TO DOCS

UiPath IntelligentOCR activities - Official documentation

Learn more about IntelligentOCR activities.

GO TO DOCS

UiPath Studio Web - Official documentation

Learn more about Studio Web.

GO TO DOCS

Complete the module

Complete the module

After exiting this module, please fill in the **Feedback Survey**.

This will ensure you receive completion on the course and will help us design better learning experiences for you.

Congratulations!

You have reached the end of the
UiPath IXP generative extraction technical deep dive
module!

EXIT MODULE

Until next time, happy automation!