

5.1 Create a multimodal analysis solution with Azure AI Content Understanding

Use **Azure AI Content Understanding** for **multimodal content analysis and information extraction**.

Learning objectives

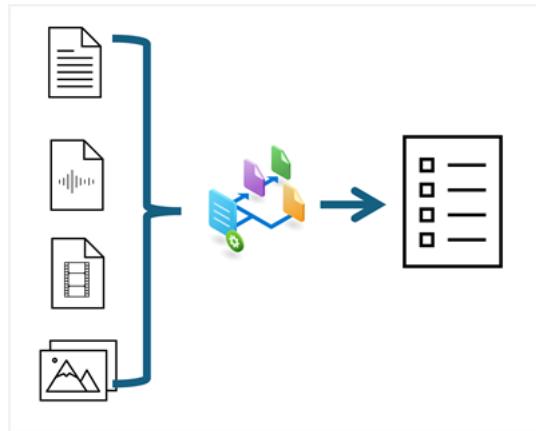
After completing this module, you will be able to:

- Describe capabilities of Azure AI Content Understanding.
- Use Azure AI Content Understanding to build a **content analyzer**.
- Consume a **Content Understanding analyzer** by using the REST API.

Introduction

Organizations today rely on information that is often locked up in content assets such as documents, images, videos, and audio recordings. Extracting information from this content can be challenging, laborious, and time-consuming, and organizations often need to build solutions based on multiple technologies for content analysis depending on the formats being used.

Azure AI Content Understanding is a **multimodal service** that simplifies the creation of AI-powered analyzers that can extract information from **content** in practically any format.



In this module, you'll explore the capabilities of Azure AI Content Understanding, and learn how to use it to build custom analyzers.

Note: Azure AI Content Understanding is currently in public preview. Details described in this module are subject to change.

What is Azure AI Content Understanding?

Azure AI Content Understanding is a *generative AI service* that you can use to extract insights and data from multiple kinds of content. With Content Understanding, you can quickly build applications that analyze complex data and generate outputs that can be used to automate and optimize processes.

Content Understanding is a component of Azure AI services. To use it, you need to provision an Azure AI services resource in your Azure subscription. You can develop and manage a Content Understanding solution:

- In the **Azure AI Foundry portal**
- By using the **Content Understanding REST API**

Multimodal content analysis

Content Understanding can extract information from common kinds of content, enabling you to use a single service with a straightforward and consistent development process to **build multimodal content analysis solutions**.

Documents and forms

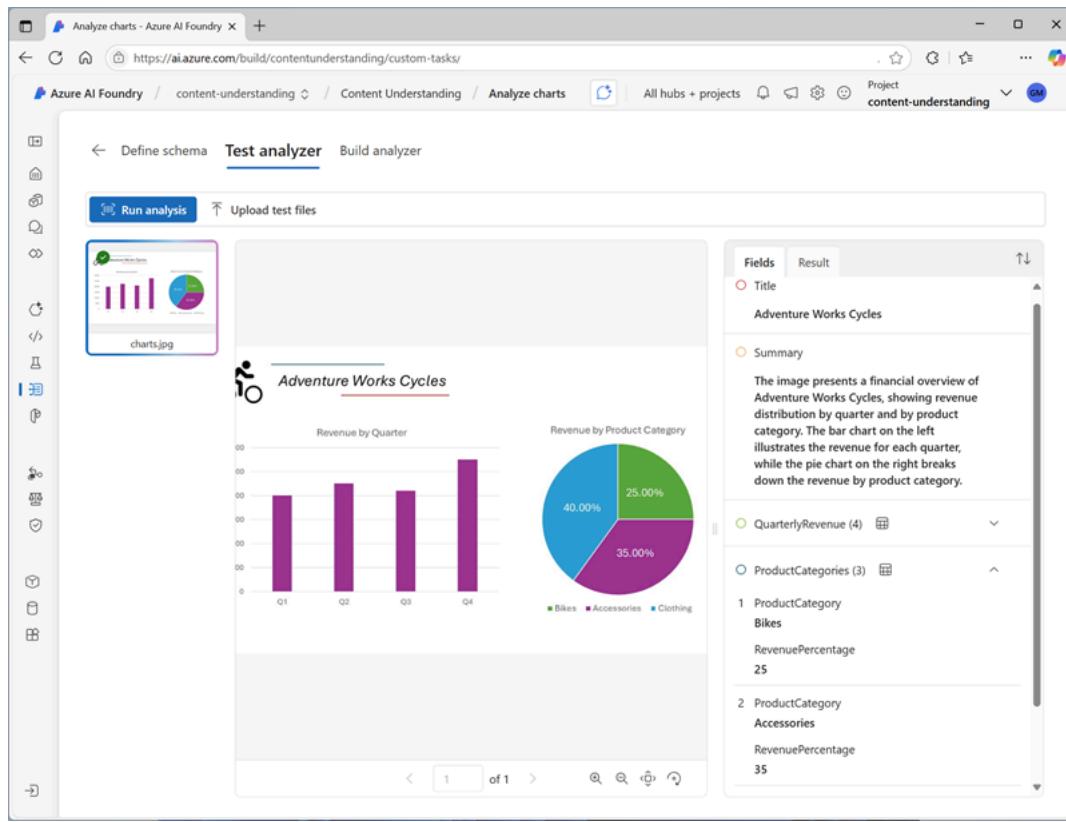
You can use Content Understanding to **analyze documents and forms and retrieve specific field values**. For example, you could extract key data values from an invoice to automate payment processing.

The screenshot shows the Azure AI Foundry interface with the project 'content-understanding'. In the center, there's a 'Test analyzer' section with two invoices uploaded: 'invoice-1234.pdf' and 'invoice-1235.pdf'. The 'invoice-1235.pdf' is highlighted with a blue box. On the right, a table displays the extracted fields and their confidence scores:

Fields	Result	↑
CustomerAddress	p.1	86.90%
	321 Pond Lane Waterville England GL1 010	
CustomerName	p.1	96.80%
	Ava Jones	
InvoiceDate	p.1	99.80%
	2025-07-03	
InvoiceId	p.1	97.20%
	1235	
InvoiceTotal	p.1	97.60%
	115.62	
SubTotal	p.1	98.90%
	91.48	
TotalTax	p.1	98.40%
	9.14	
VendorAddress	p.1	97.80%

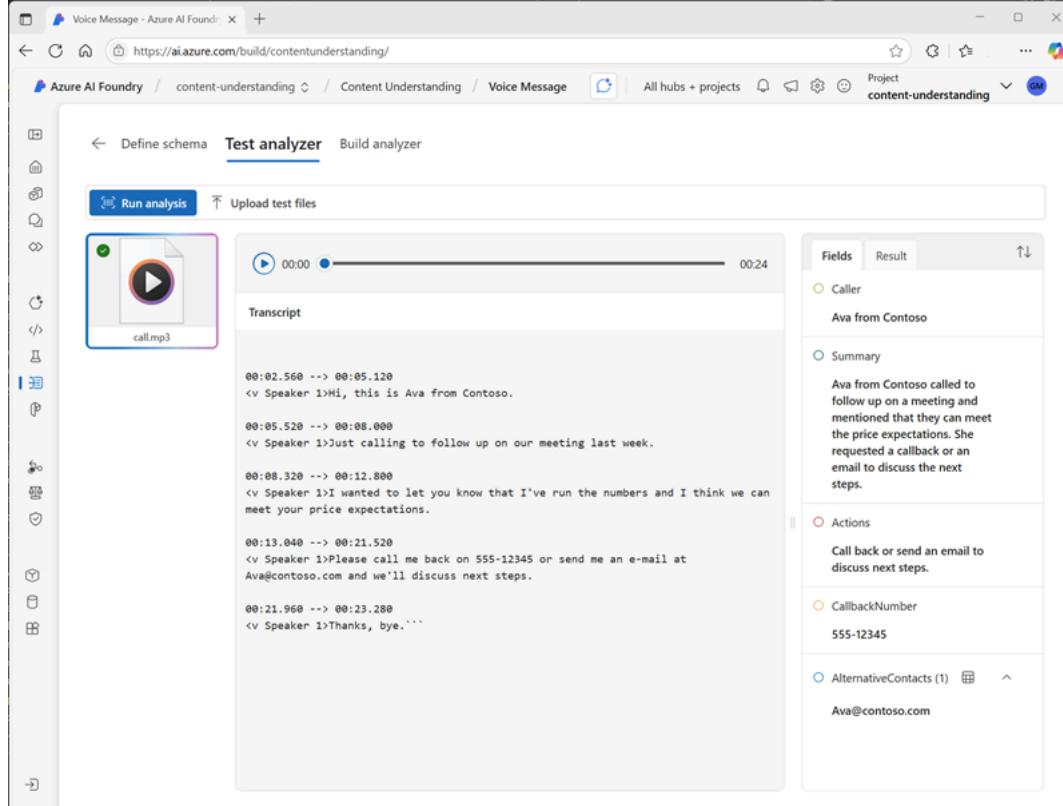
Images

You can analyze images to *infer information from visuals such as charts, identify physical defects in products or other items, detect the presence of specific objects or people, or determine other information visually*.



Audio

Analysis of audio enables you to automate tasks like *summarizing conference calls*, *determining sentiment of recorded customer conversations*, or *extracting key data from telephone messages*.



Video

Video accounts for a large volume of the data captured today, and you can use Content Understanding to **analyze and extract insights from video** to support many scenarios. For example, to *extract key points from video conference recordings*, *to summarize presentations*, or *to detect the presence of*

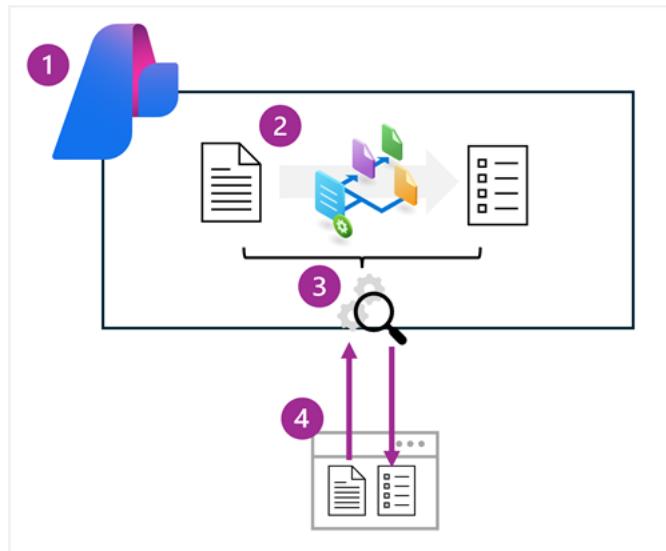
specific activity in security footage.

The screenshot shows the Azure AI Foundry interface with the project 'content-understanding'. On the left, there's a sidebar with various icons. In the center, under 'Test analyzer', there's a video thumbnail for 'meeting-2.mp4' with a play button. Below it is a bar chart titled 'Projected user adoption' showing three bars. A timeline slider is at 00:33. To the right, there's a 'Fields' section and a 'Result' section. The 'Fields' section lists 'Shot 2 00:23' and 'summary'. The 'Result' section contains a narrative about Harry sharing a slide with projected user adoption metrics for the first three months, impressing Lisa. It also lists participants (Harry, Lisa), assigned actions (task: send presentation slide to Lisa), and shared slides (Projected user adoption metrics for the first three months).

Create a Content Understanding analyzer

Content Understanding solutions are based on the creation of an **analyzer**; which is trained to extract specific information from a particular type of content based on a schema that you define.

The high-level process for creating a Content Understanding solution includes the following steps:



1. Create an Azure AI services resource.
2. Define a **Content Understanding schema** for the information to be extracted. This can be based on a **content sample** and a **analyzer template**.
3. Build an **analyzer** based on the completed schema.
4. Use the **analyzer** to extract or generate fields from new content.

Numerous **analyzer templates** are provided to help you develop an appropriate analyzer for your needs quickly. Additionally, because of the generative AI capabilities of Content Understanding, you can use minimal training data to define a schema by example. In many cases, the service accurately identifies the data values in the sample content that map to the schema elements automatically, though you can also explicitly label fields in content such as documents to improve the performance of your analyzer.

Creating an analyzer with Azure AI Foundry

While you can provision an Azure AI services resource and develop a complete Content Understanding solution through the REST API, **the preferred approach for AI development projects is to use Azure AI Foundry**. Specifically, you can **use the Azure AI Foundry portal to create a project, define a Content Understanding schema, and build and test an analyzer**.

1. Creating a Content Understanding project

In Azure AI Foundry, you can *create a project* in an existing AI hub, or you can create a new hub as you create the project. In addition to the AI hub itself, creating a hub provisions the Azure resources needed to support one or more projects; including an Azure AI services resource, storage, and a key vault resource to store sensitive details like credentials and keys.

Note: Content Understanding schemas can only be created in Azure locations where the service is supported. For more information, see [Content Understanding region and language support](#).

2. Defining a schema

After creating a project, **the first step in building an analyzer is to define a schema** for the content the analyzer will process, and the information it will extract. **Azure AI Foundry provides a schema editor interface** in which you can upload a file (document, image, audio, or video) on which the schema should be based. You can then apply an appropriate schema template and define the specific fields you want the analyzer to identify.

Note: The templates and field types available in a schema depend on the content type of the file on which the schema is based. Some content types support additional optional functionality, such as extracting barcodes and formulae from text in documents. For more information about using Content Understanding with different content types, see the following articles in the product documentation:

- [Content Understanding document solutions](#)
- [Content Understanding image solutions](#)
- [Content Understanding audio solutions](#)
- [Content Understanding video solutions](#)

3. Testing

You can **test the analyzer schema** at any time during the development process by running analysis on the sample file used to define the schema or other uploaded files. **The test results include the extracted field values and the JSON format output** returned by the analyzer to client applications.

4. Building an analyzer

When you're satisfied with the performance of your schema, you can build your analyzer. Building an analyzer makes it **accessible to client applications through Content Understanding endpoint** for the Azure AI services resource associated with your project.

After building your analyzer, you can continue to test it in the Azure AI Foundry portal, and refine the schema to create new named versions with different capabilities.

Use the Content Understanding REST API

The **Content Understanding REST API** provides a programmatic interface that you can use to **create, manage, and consume analyzers**.

To use the REST API, your *client application submits HTTP calls to the Content Understanding endpoint* for your Azure AI services resource, **passing one of the authorization keys in the header**. You can obtain the endpoint and keys in the Azure portal or in the Azure AI Foundry portal. You can also use the Azure AI Foundry API to connect to the project and retrieve the endpoint and key for your Azure AI Services resource programmatically.

The screenshot shows the Azure AI Foundry interface for the 'content-understanding' project. On the left, there's a sidebar with various options like Overview, Model catalog, Playgrounds, AI Services, Agents, Templates, Fine-tuning, Content Understanding (selected), Prompt flow, Assess and improve, Tracing, Evaluation, Safety + security, My assets (Models + endpoints, Data + indexes, Web apps), and Management center. The main content area has tabs for 'Endpoints and keys' and 'Project details'. Under 'Endpoints and keys', there's an 'API Key' field containing a redacted string, 'Included capabilities' (Azure AI inference, Azure OpenAI, Azure AI Services selected), and 'Azure AI Services endpoint' set to 'https://xxxxxxxxxxxxxxxxxxxxxx.cognitiveservices.azure.com/'. It also shows 'Speech to text endpoint' at 'https://westus.stt.speech.microsoft.com' and 'Text to speech endpoint' at 'https://westus.tts.speech.microsoft.com'. A link to 'API documentation' is present. Under 'Project details', there's a 'Project connection string' field with a redacted value, a 'Subscription' section showing 'My Subscription', and a 'Management center' button.

Using the REST API to analyze content

One of the most common uses of the REST API is to submit content to an existing analyzer that you have previously built, and retrieve the results of analysis. The analysis request returns an **operation ID** value that represents an asynchronous task. Your client application must then use another request to pass the operation ID back to the endpoint and retrieve the **operation status** - potentially polling multiple times until the operation is complete and the results are returned in JSON format.

For example, to analyze a document, a client application might submit a POST request to the analyze function containing the following JSON body:

```
POST {endpoint}/contentunderstanding/analyzers/{analyzer}:analyze?api-version={api version} { "url": "https://host.com/doc.pdf" }
```

Note: You can specify a URL for the content file location, or you can include the binary contents of the file.

Assuming the request is authenticated and initiated successfully, the response will be similar to this example:

```
Operation-Id: 1234abcd-1234-abcd-1234-abcd1234abcd
```

```
Operation-Location: {endpoint}/contentunderstanding/analyzers/{analyzer}/results/1234abcd-1234-abcd-1234-abcd1234abcd?api-version={api version} { }
```

Your client application must then use the **operation ID** that has been returned to check the status of the operation until it has succeeded (or failed) by submitting a GET request to the results function.

```
GET {endpoint}/contentunderstanding/analyzers/{analyzer}/results/1234abcd-1234-abcd-1234-abcd1234abcd?api-version={api version}
```

When the operation has completed successfully, the response contains a JSON payload representing the results of the analysis. The specific results depend on the content and schema.

Note: For more information about the Content Understanding REST API, see the [reference documentation](#).

Exercise - Extract information from multimodal content

Extract information from multimodal content

In this exercise, you use Azure Content Understanding to extract information from a variety of content types; including an invoice, an images of a slide containing charts, an audio recording of a voice messages, and a video recording of a conference call.

A screenshot of a web browser window titled "Azure AI Foundry" and "Azure-AI-Content-Understanding". The URL is <https://ai.azure.com/?tid=4e1ee3db-4df6-4142-b7b9-bec15f1...>. The page displays a search bar at the top with the placeholder "Search for a model". Below it, a section titled "Latest models" shows six model cards:

- DeepSeek-R1-0528** (Chat completion)
- grok-3** (Chat completion)
- grok-3-mini** (Chat completion)
- model-router** (Chat completion)
- o3** (Chat completion)
- o4-mini** (Chat completion)

Below this, a section titled "Explore more capabilities" features a large image of a lion's face with the text "Boundless innovation". A callout box contains the text:

Drive efficiency and engagement with transformative workflows and cutting-edge videos using Sora in Azure AI Foundry Models.

[Go to video playground](#)

A green arrow points from the "Explore Azure AI Services" section below to the "Go to video playground" button.

Build apps with code templates
Use code templates to quickly create a proof-of-concept app and deploy it to production.

Explore Azure AI Services
Create market-ready AI applications using customizable APIs and models.

AI Services - Azure AI Foundry Azure-AI-Content-Understanding

<https://ai.azure.com/explore/aiservices?tid=4e1ee3db-...>

MBI BIM AI PennDOT GIS Azure OneDrive Udemy Azure DevOps WorkshopPLUS

 **Speech**
Enhance customer experiences through speech to text, text to speech, and speech translation features.
[View all Speech capabilities](#)

 **Language + Translator**
Analyze, summarize and translate using LLM-powered natural language processing capabilities.
[View all Language + Translator capabilities](#)

 **Vision + Document**
Discover information and insights from documents, images and video with OCR and multi-modal AI.
[View all Vision + Document capabilities](#)

 **Content Safety**
Detect harmful, offensive, or inappropriate user-generated or AI-generated content in your app including text, image, and multi-modal APIs.
[View all Content Safety capabilities](#)



Azure AI Foundry / AI Services / Vision + Document

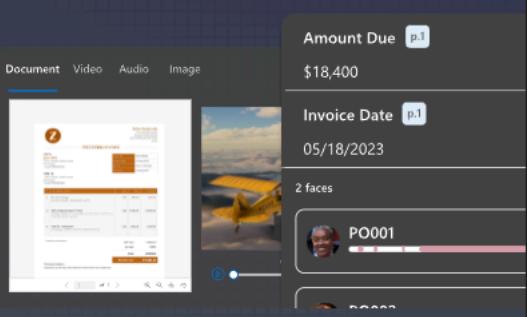
← Vision + Document

Give your apps the ability to read text, analyze images, process documents and detect faces with technology like optical character recognition (OCR) and machine learning.

Content Understanding

Turn unstructured documents, images, video, and audio into structured data with the new Generative AI-powered Content Understanding service.

[Try Content Understanding](#)



View all other vision capabilities

Module assessment

1. What kinds of AI solution is Azure AI Content Understanding designed to help you build? **Analyzers that extract information from documents, images, videos, and audio files.**
2. Which graphical tool should you use to create an Azure AI Content Understanding project? **Azure AI Foundry portal.**
3. What should you define for the information you want to extract from content? **A schema**

Summary

Azure AI Content Understanding is a multimodal AI service that enables you to extract information from many different kinds of content.

In this module, you learned how to use the **Azure AI Foundry portal** to create a Content Understanding project and build an **analyzer**.

Note: For more information about Azure AI Content Understanding, see [Azure AI Content Understanding documentation](#).

5.2 Create an Azure AI Content Understanding client application

Use the **Azure AI Content Understanding REST API** for multimodal content analysis and information extraction.

Learning objectives

After completing this module, you will be able to:

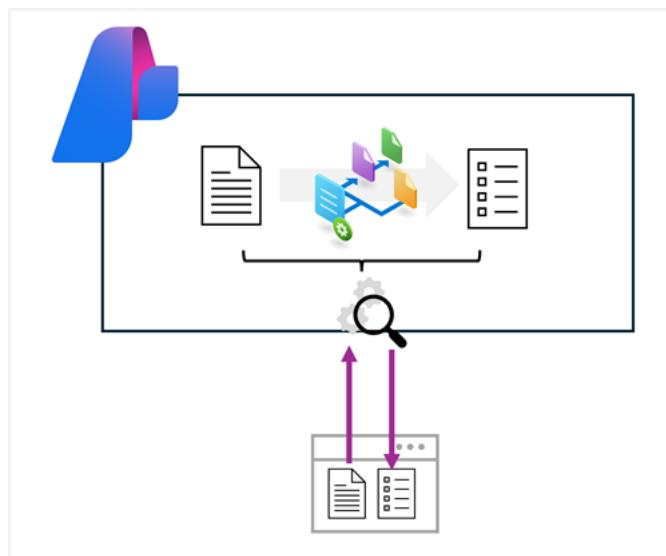
- Use the Azure AI Content Understanding REST API to **build a content analyzer**.
- Use the Azure AI Content Understanding REST API to **consume a content analyzer**.

Introduction

Azure AI Content Understanding is a multimodal service that simplifies the creation of AI-powered analyzers that can extract information from **multiple content formats**, including *documents, images, audio files, and videos*.

Tip: To learn how to build Azure AI Content Understanding analyzers, complete the [Create a multimodal analysis solution with Azure AI Content Understanding](#) module.

You can develop client applications that use **Azure AI Content Understanding analyzers** by using the **Azure AI Content Understanding REST API**; which is the focus of this module.



In this module, you'll learn how to write code that **uses the REST API to submit a content file to an analyzer and process the results**.

Note: Azure AI Content Understanding is currently in public preview. Details described in this module are subject to change.

Prepare to use the AI Content Understanding REST API

Before you can use the **Azure AI Content Understanding REST API**, you need an **Azure AI services multi-services resource** in your Azure subscription. You can provision this resource in the following ways:

- Create an **Azure AI services** resource in the Azure portal.
- Create an **Azure AI Foundry** hub, which includes an Azure AI services resource by default.

Tip: Creating an Azure AI Foundry hub enables you to work in an Azure AI Foundry project, in which you can use visual tools to create and manage Azure AI Content Understanding schemas and analyzers.

After you've provisioned an Azure AI services resource, you need the following information to connect to the **Azure AI Content Understanding REST API** from a client application:

- The Azure AI services resource **endpoint**
- One of the **API keys** associated with the endpoint.

You can obtain these values from the Azure portal, as shown in the following image:

The screenshot shows the Microsoft Azure portal interface. In the top left, it says "my-ai-svcs - Microsoft Azure". The address bar shows "https://ms.portal.azure.com/". The top navigation bar includes "Microsoft Azure", a search bar, and a "Copilot" button. Below the navigation is a breadcrumb trail: "Home > Microsoft.CognitiveServicesAISevices-20250515181355 | Overview > ai-services > my-ai-svcs". The main title is "my-ai-svcs | Keys and Endpoint". On the left, there's a sidebar with various service links like Overview, Activity log, Access control (IAM), Tags, etc., and the "Keys and Endpoint" link is highlighted. The main content area has a note about API keys, two text input fields for "KEY 1" and "KEY 2" each with a copy icon, and a dropdown for "Location/Region" set to "eastus". Below this, there are tabs for OpenAI, Speech, Content Safety, Computer Vision, and Content Understanding, with "Content Understanding" being the active tab. A note below the tabs says "Use the below endpoints to call into Content Understanding APIs." followed by a link "Learn more" and the endpoint URL "https://my-ai-svcs.services.a...". At the bottom left, there's a note "Add or remove favorites by pressing Ctrl+Shift+F".

If you're working within an Azure AI Foundry project, you can find the **endpoint** and **key** for the associated *Azure AI services resource* in the *Azure AI Foundry portal*, as shown in the following image:

The screenshot shows the Azure AI Foundry interface for a 'content-understanding' project. On the left, there's a sidebar with various options like Overview, Model catalog, Playgrounds, AI Services, Agents, Templates, Fine-tuning, Content Understanding (selected), Prompt flow, Assess and improve, Tracing, Evaluation, Safety + security, My assets (Models + endpoints, Data + indexes, Web apps), and Management center. The main content area has tabs for 'Endpoints and keys' and 'Project details'. Under 'Endpoints and keys', there's an 'API Key' field containing a redacted key, 'Included capabilities' (Azure AI inference, Azure OpenAI, Azure AI Services selected), and 'Azure AI Services endpoint' set to https://xxxxxxxxxxxxxxxxxxxxxx.cognitiveservices.azure.com/. It also shows 'Speech to text endpoint' at https://westus.stt.speech.microsoft.com/ and 'Text to speech endpoint' at https://westus.tts.speech.microsoft.com/. A link to 'API documentation' is present. Under 'Project details', there's a 'Project connection string' (redacted) and a 'Subscription' section showing 'My Subscription'. At the top right, there's a 'Help' button.

When working in an Azure AI Foundry project, you can also write code that uses the **Azure AI Foundry SDK** to connect to the project using **Microsoft Entra ID authentication**, and retrieve the connection details for the Azure AI services resource; including the endpoint and key.

Tip: To learn more about programming with the Azure AI Foundry SDK, complete the [Develop an AI app with the Azure AI Foundry SDK](#) module.

Create a Content Understanding analyzer

In most scenarios, you should consider creating and testing analyzers using the **visual interface in the Azure AI Foundry portal**. However, in some cases you might want to create an analyzer by submitting a **JSON definition of the schema** for your desired content fields to the **REST API**.

Defining a schema for an analyzer

Analyzers are based on **schemas that define the fields you want to extract or generate from a content file**. At its simplest, a schema is a set of fields, which can be specified in a JSON document, as shown in this example of an analyzer definition:

```
{ "description": "Simple business card", "baseAnalyzerId": "prebuilt-documentAnalyzer", "config": { }, "returnDetails": true }
```

This example of a custom analyzer schema is based on the **pre-built document analyzer**, and describes *two fields* that you would expect to find on a business card: *ContactName* and *EmailAddress*. Both fields are defined as string data types, and are expected to be extracted from a document (in other words, the string values are expected to exist in the document so they can be "read"; rather than being fields that can be generated by inferring information about the document).

Note: This example is deliberately simple, with the minimal information needed to create a working analyzer. In reality, the schema would likely include more fields of different types, and the analyzer definition would include more configuration settings. The JSON might even include a sample document. See the [Azure AI Content Understanding REST API documentation](#) for more details.

Using the REST API to create an analyzer

With your analyzer definition in place, you can use the REST API to submit it to Azure AI Content Understanding to be created. The JSON data is submitted as a `PUT` request to the **endpoint** with the **API key in the request header** to start the analyzer creation operation.

The response from the `PUT` request includes a `Operation-Location` in the header, which provides a **callback URL** that you can use to **check on the status of the request** by submitting a `GET` request.

You can use any HTTP-capable client tool or language to submit the request. For example, the following Python code submits a request to create an analyzer based on the contents of a file named `card.json` (which is assumed to contain the JSON definition described previously)

```
import json
import requests

# Get the business card schema
with open("card.json", "r") as file:
    schema_json = json.load(file)

# Use a PUT request to submit the schema for a new analyzer
analyzer_name = "business_card_analyser"

headers = {
    "Ocp-Apim-Subscription-Key": "<YOUR_API_KEY>",
    "Content-Type": "application/json"}

url = f"<YOUR_ENDPOINT>/contentunderstanding/analyzers/{analyzer_name}?api-version=2025-05-01-preview"

response = requests.put(url, headers=headers, data=json.dumps(schema_json))

# Get the response and extract the ID assigned to the operation
callback_url = response.headers["Operation-Location"]

# Use a GET request to check the status of the operation
result_response = requests.get(callback_url, headers=headers)

# Keep polling until the operation is complete
status = result_response.json().get("status")
while status == "Running":
    result_response = requests.get(callback_url, headers=headers)
    status = result_response.json().get("status")

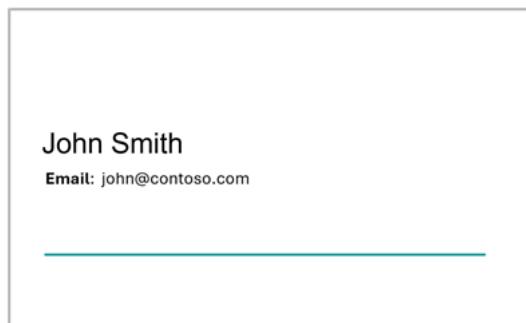
print("Done!")
```

Analyze content

To analyze the contents of a file, you can use the **Azure AI Content Understanding REST API** to submit it to the **endpoint** using a `POST` request. You can specify the **content as a URL** (for a file hosted in an Internet-accessible location) or as the **binary contents of the file** (for example, a .pdf document, a .png image, an .mp3 audio file, or an .mp4 video file). The request header must include the **API key**, and the **endpoint** address for the analyze request includes the analyzer to be used.

As with the request to create an analyzer, the analyze request starts an **asynchronous** operation. The `POST` request returns a unique operation ID, which you can then use in a `GET` request to **check the status of the analysis operation**.

For example, suppose you want to use the *business card analyzer* discussed previously to extract the name and email address from the following scanned business card image:



The following Python code submits a request for analysis, and then polls the service until the operation is complete and the results are returned.

```

1 import json
2 import requests
3
4 # Read the image data
5 with open("business-card.png", "rb") as file:
6     image_data = file.read()
7
8 ## Use a POST request to submit the image data to the analyzer
9 analyzer_name = "business_card_analyser"
10
11 headers = {
12     "Ocp-Apim-Subscription-Key": "<YOUR_API_KEY>",
13     "Content-Type": "application/octet-stream"}
14
15 url = f"{<YOUR_ENDPOINT>}/contentunderstanding/analyzers/{analyzer_name}:analyze?
api-version=2025-05-01-preview"
16
17 response = requests.post(url, headers=headers, data=image_data)
18
19 # Get the response and extract the ID assigned to the analysis operation
20 response_json = response.json()
21 id_value = response_json.get("id")
22
23 # Use a GET request to check the status of the analysis operation
24 result_url = f"{<YOUR_ENDPOINT>}/contentunderstanding/analyzerResults/{id_value}?
api-version=2025-05-01-preview"
25
26 result_response = requests.get(result_url, headers=headers)
27
28 # Keep polling until the analysis is complete
29 status = result_response.json().get("status")
30 while status == "Running":
31     result_response = requests.get(result_url, headers=headers)
32     status = result_response.json().get("status")
33
34 # Get the analysis results
35 if status == "Succeeded":
36     result_json = result_response.json()

```

Processing analysis results

The results in the response JSON depend on:

- The **kind of content** the *analyzer* is designed to analyze (for example, document, video, image, or audio).
- The **schema** for the analyzer.
- The **contents of the file** that was analyzed.

For example, the response from the document-based business card analyzer when analyzing the business card described previously contain:

- The extracted fields
- The *optical character recognition (OCR)* layout of the document, including locations of lines of text, individual words, and paragraphs on each page.

Here's the complete JSON response for the business card analysis:

- Confidence score
- Span
- Content
- Source

Your application must typically parse the JSON to retrieve field values. For example, the following python code extracts all of the string values:

```

# (continued from previous code example)

# Iterate through the fields and extract the names and type-specific values
contents = result_json["result"]["contents"]
for content in contents:
    if "fields" in content:
        fields = content["fields"]
        for field_name, field_data in fields.items():
            if field_data['type'] == "string":
                print(f"{field_name}: {field_data['valueString']}")

```

The output from this code is shown here:

```
ContactName: John Smith  
EmailAddress: john@contoso.com
```

Exercise - Develop a Content Understanding client application

Now it's your turn to build your own Content Understanding client application!

In this exercise, you use the Azure AI Content Understanding REST API to extract information from content by submitting a file to an analyzer.

Develop a Content Understanding client application

In this exercise, you use Azure AI Content Understanding to create an analyzer that extracts information from business cards. You'll then develop a client application that uses the analyzer to extract contact details from scanned business cards.

Module assessment

1. What configuration values are needed to use the Azure AI Content Understanding REST API? **The endpoint and key for the Azure AI service.**
2. What must be specified when calling the analyze method to extract fields from content? **The name of the analyzer.**
3. How are the extracted fields returned? **As type-specific values.**

Summary

Azure AI Content Understanding is a **multimodal AI service** that enables you to **extract information from many different kinds of content**. The REST API for the service enables you to create client applications that analyze content to extract and generate field values.

Note: For more information about Azure AI Content Understanding, see [Azure AI Content Understanding documentation](#).

5.3 Use prebuilt Document intelligence models

Learn what data you can analyze by choosing **prebuilt Forms Analyzer models** and how to deploy these models in a **Document intelligence solution**.

Learning objectives

In this module, you'll learn to:

- Identify business problems that you can solve by using prebuilt models in **Forms Analyzer**.
- Analyze forms by using the **General Document, Read, and Layout models**.
- Analyze forms by using **financial, ID, and tax prebuilt models**.

Introduction

Many forms and documents that businesses handle are common across disparate companies in different sectors. For example, most companies use invoices and receipts. Microsoft **Azure AI Document Intelligence** includes **prebuilt models** so you can handle common document types easily.

Imagine you conduct polls for private companies and political parties. Participants submit their responses as paper forms or as online PDFs. You've decided to deploy **Azure AI Document Intelligence** to streamline data entry. You want to know if you can use the prebuilt models to generate meaningful data from your forms.

In this module, you'll learn about the capabilities of the **prebuilt models in Azure AI Document Intelligence** and how to use them.

Understand prebuilt models

Prebuilt models in Azure AI Document Intelligence enable you to **extract data from common forms and documents without training your own models**.

In your polling company, polling forms are unique to each survey project, but you also use invoices and receipts to record financial transactions and you have many unstructured documents. You want to know how much work is required to extract names, addresses, amounts, and other information from these documents.

Here, you learn how prebuilt models can help you analyze common document types.

What are prebuilt models?

The general approach used in AI solutions is to provide a large quantity of sample data and then train an optimized model by trying different data features, parameters, and statistical treatments. The combination that best predicts the values that interest you constitute the trained model, and you can use this model to predict values from new data.

Many of the forms that businesses use from day to day are of a few common types. For example, most businesses issue or receive invoices and receipts. Any business that has employees in the United States must use the W-2 tax declaration form. Also you often have more general documents that you might want to extract data from. For these cases, Microsoft helps you by providing prebuilt models.

Prebuilt models are already trained on large numbers of their target form type.

If you want to use Document Intelligence to extract data from one of these common forms or documents, you can choose to use a prebuilt model and you don't have to train your own. Because Microsoft trains these models on a large corpus of examples, you can expect them to provide accurate and reliable results when dealing with their intended forms.

Several of the prebuilt models are trained on specific form types:

- **Invoice** model. Extracts common fields and their values from invoices.
- **Receipt** model. Extracts common fields and their values from receipts.
- **US Tax** model. Unified US tax model that can extract from forms such as W-2, 1098, 1099, and 1040.
- **ID document** model. Extracts common fields and their values from US drivers' licenses, European Union IDs and drivers license, and international passports.
- **Business card** model. Extracts common fields and their values from business cards.
- **Health insurance card** model. Extracts common fields and their values from health insurance cards.
- **Marriage certificate**. Extracts information from marriage certificates.
- **Credit/Debit card** model. Extracts common information from bank cards.
- **Mortgage documents**. Extracts information from mortgage closing disclosure, Uniform Residential Loan Application (Form 1003), Appraisal (Form 1004), Validation of Employment (Form 1005), and Uniform Underwriting and Transmittal Summary (Form 1008).
- **Bank statement** model. Extracts account information including beginning and ending balances, transaction details from bank statements.
- **Pay Stub** model. Extracts wages, hours, deductions, net pay, and other common pay stub fields.
- **Check** model. Extracts payee, amount, date, and other relevant information from checks.

The other models are designed to extract values from documents with less specific structures:

- **Read model**. Extracts text and languages from documents.
- **General document model**. Extract text, keys, values, entities, and selection marks from documents.
- **Layout model**. Extracts text and structure information from documents.

Features of prebuilt models

The prebuilt models are designed to extract different types of data from the documents and forms users submit. To select the right model for your requirements, you must understand these features:

- **Text extraction**. All the prebuilt models extract lines of text and words from hand-written and printed text.
- **Key-value pairs**. Many models extract spans of text within a document that identify a label or key and its response or value as key-values pairs. For example, *a typical key might be Weight and its value might be 31 kg*.
- **Entities**. Text that includes common, more complex data structures can be extracted as entities. **Entity types include people, locations, and dates**.
- **Selection marks**. Some models extract spans of text that indicate a choice as selection marks. These **marks include radio buttons and check boxes**.
- **Tables**. Many models can extract tables in scanned forms included the data contained in cells, the numbers of columns and rows, and column and row headings. Tables with merged cells are supported.
- **Fields**. Models trained for a specific form type identify the values of a fixed set of fields. For example, the Invoice model includes **CustomerName** and **InvoiceTotal** fields.

Also consider that prebuilt models are designed for and trained on generic document and form types. If you have an industry-specific or unique form type that you use often, you might be able to obtain more reliable and predictable results by using a custom model. However, **custom models take time to develop because you must invest the time and resources to train them on example forms before you can use it**. *The larger the number of example forms you provide for training, the better the model is at predicting form content accurately*.

Input requirements

The prebuilt models are flexible but you can help them to return accurate and helpful results by **submitting one clear photo or high-quality scan for each document**.

You must also comply with these requirements when you submit a form for analysis:

- The file must be in **JPEG, PNG, BMP, TIFF, or PDF** format. Additionally, the Read model can accept Microsoft Office files.
- The file must be smaller than **500 MB for the standard tier**, and **4 MB for the free tier**.
- Images must have dimensions **between 50 x 50 pixels and 10,000 x 10,000 pixels**.
- PDF documents must have dimensions **less than 17 x 17 inches or A3 paper size**.
- PDF documents must **not be protected with a password**.

Note: If you can, submit *text-embedded PDF files* because they eliminate errors in character recognition.

PDF and TIFF files can have any number of pages but, in the **standard tier**, only the **first 2,000 pages** are analyzed. In the **free tier**, only the **first two pages** are analyzed.

Try out prebuilt models with Azure AI Document Intelligence Studio

Azure AI Document Intelligence is designed as a web service you can call using code in your custom applications. However, it's often helpful to explore the models and how they behave with your forms visually. You can perform such experiments by using [Azure AI Document Intelligence Studio](#) and use the experience to help design and write your code.

You can choose any of the prebuilt models in Azure AI Document Intelligence Studio. Microsoft provides some sample documents for use with each model or you can add your own documents and analyze them.

The screenshot shows the Azure AI Form Recognizer Studio interface. At the top, there's a blue header bar with the title "Applied AI | Form Recognizer Studio - Preview". Below the header, there's a navigation bar with links for "Help us improve Form Recognizer", "Take our survey!", "If you need help, please contact support.", and "New support request". On the left side, there's a sidebar with a "+ Add" button, a dropdown menu, and a preview area showing a sample business card image named "bizcard.jpg". The main workspace shows a larger image of a business card with the name "Chris Smith" and various contact details. To the right, there's a results panel titled "Fields" which lists the extracted fields and their confidence scores. The results are as follows:

Field	Value	Score
DocType	businessCard	-
Addresses	4001 1st Ave NE Redmond, WA 98052	95.80%
CompanyNames	CONTOSO	94.90%
ContactNames	Chris Smith	97.80%
Content	Chris Smith	98.20%
FirstName	Chris	98.50%
LastName	Smith	98.20%
Departments	Cloud & AI Department	84.20%
Emails	chris.smith@contoso.com	84.20%

Calling prebuilt models by using APIs

Because **Azure AI Document Intelligence implements RESTful web services**, you can use web service calls from any language that supports them. However, when you use Microsoft's Azure AI Document Intelligence APIs, security and session management is simplified and you have to write less code.

Azure AI Document Intelligence is available for:

- C# and other .NET languages.

- Java.
- Python.
- JavaScript.

Whenever you want to call Azure AI Document Intelligence, you must start by **connecting and authenticating with the service in your Azure subscription**. To make that connection, you need:

- The service **endpoint**. This value is the URL where the service is published.
- The **API key**. This value is a unique key that grants access.

You obtain both of these values from the Azure portal.

Because the service can take a few seconds to respond, it's best to use **asynchronous** calls to submit a form and then obtain results from the analysis:

```
poller = document_analysis_client.begin_analyze_document("prebuilt-layout", AnalyzeDocumentRequest(url_source=docUrl))  
result: AnalyzeResult = poller.result()
```

The details you can extract from these results depend on the model you used.

Learn more

- [What is Azure AI Document Intelligence?](#)
- [Azure AI Document Intelligence models](#)

Use the General Document, Read, and Layout models

If you want to extract text, languages, and other information from documents with unpredictable structures, you can use the **read, general document, or layout models**.

In your polling company, customers and partners often send specifications, tenders, statements of work, and other documents with unpredictable structures. You want to know if Azure AI Document Intelligence can analyze and extract values from these documents.

Here, you'll learn about the **prebuilt models** that Microsoft provides **for general documents**.

Using the read model

The **Azure AI Document Intelligence read model** extracts printed and handwritten text from documents and images. It's used to provide *text extraction* in all the other prebuilt models.

The **read model** can also **detect the language** that a line of text is written in and **classify whether it's handwritten or printed text**.

Note: The read model supports more languages for printed text than handwritten text. Check the [documentation](#) to see the current list of supported languages.

For multi-page PDF or TIFF files, you can use the `pages` parameter in your request to fix a page range for the analysis.

The read model is ideal if you want to extract words and lines from documents with no fixed or predictable structure.

Using the general document model

The general document model extends the functionality of the read model by adding the **detection of key-value pairs, entities, selection marks, and tables**. The model can extract these values from **structured, semi-structured, and unstructured documents**.

The general document model is the only prebuilt model to support entity extraction. It can recognize entities such as **people, organizations, and dates** and it runs against the whole document, not just key-value pairs. This approach ensures that, **when structural complexity has prevented the model extracting a key-value pair, an entity can be extracted instead**. Remember, however, that sometimes a single piece of text *might return both a key-value pair and an entity*.

The types of entities you can detect include:

- Person . The name of a person.
- PersonType . A job title or role.
- Location . Buildings, geographical features, geopolitical entities.
- Organization . Companies, government bodies, sports clubs, musical bands, and other groups.
- Event . Social gatherings, historical events, anniversaries.
- Product . Objects bought and sold.
- Skill . A capability belonging to a person.
- Address . Mailing address for a physical location.
- Phone number . Dialing codes and numbers for mobile phones and landlines.
- Email . Email addresses.
- URL . Webpage addresses.
- IP Address . Network addresses for computer hardware.
- DateTime . Calendar dates and times of day.
- Quantity . Numerical measurements with their units.

Using the layout model

As well as extracting text, the layout model returns **selection marks** and **tables** from the input image or PDF file. It's a good model to use when you need **rich information about the structure of a document**.

When you digitize a document, it can be at an odd angle. Tables can have complicated structures with or without headers, cells that span columns or rows, and incomplete columns or rows. The layout model can handle all of these difficulties to extract the complete document structure.

For example, each **table cell** is extracted with:

- Its content text.
- The size and position of its bounding box.
- If it's part of a header column.
- Indexes to indicate its row and column position in the table.

Selection marks are extracted with their **bounding box, a confidence indicator, and whether they're selected or not**.

Learn more

- [Language support for Azure AI Document Intelligence](#)
- [Azure AI Document Intelligence read model](#)
- [Azure AI Document Intelligence general document model](#)
- [Azure AI Document Intelligence layout model](#)

Use financial, ID, and tax models

Azure AI Document Intelligence includes some prebuilt models that are trained on common form types. You can use these models to obtain the values of common fields from **invoices, receipts, business cards**, and more.

In your polling company, invoices and receipts are often submitted as photos or scans of the paper documents. Sometimes the scan is poor and the paper is creased or damaged. You want to know if Azure AI Document Intelligence can get this information into your databases more efficiently than manual data entry.

Here, you'll learn about the prebuilt models that handle financial, identity, and tax documents.

Using the invoice model

Your business both issues invoices and receives them from partner organization. There might be many different formats on paper or in digitized forms and some will have been scanned poorly at odd angles or from creased paper.

The invoice model in Azure AI Document Intelligence can handle these challenges and uses the features of the read model to extract text from invoice scans. In addition, it extracts specific fields that are commonly used on invoices including:

- Customer name and reference ID
- Purchase order number
- Invoice and due dates
- Details about the vendor, such as name, tax ID, physical address.
- Similar details about the customer.
- Billing and shipping addresses.
- Amounts such as total tax, invoice total, and amount due.

Invoices also feature lines, usually in a table, each of which is one purchased item. **For each line**, the invoice model identifies details including:

- The description and product code of the product or service invoiced.
- Amounts such as the unit price, the quantity of items, the tax incurred, and the line total.

Using the receipt model

Receipts have similar fields and structures to invoices, but they record amounts paid instead of amounts charged. Azure AI Document Intelligence faces the same challenges of poor scanning or digitization but can reliably identify fields including:

- **Merchant details** such a name, phone number, and address.
- **Amounts** such as receipt total, tax, and tip.
- The **date and time of the transaction**.

As for invoices, receipts often include a table of items, each of which is a product or service purchased. **For each of these lines**, the model recognizes:

- The name of the item.
- The quantity of the item purchased.
- The unit price of the item.
- The total price for that quantity.

Note: In Azure AI Document Intelligence v3.0 and later, the receipt model supports single-page hotel receipt processing. If a receipt is classified as a hotel receipt, the model extracts extra relevant fields such as **arrival and departure dates**.

Using the ID document model

The ID document model is trained to analyze two types of identity document:

- **United States drivers licenses**.
- **International passports**.

Note: Only the biographical pages of passports can be analyzed. Visas and other travel documents are not supported.

The ID document model can extract fields including:

- First and last names.
- Personal information such as sex, date of birth, and nationality.
- The country and region where the document was issued.
- Unique numbers such as the document number and machine readable zone.
- Endorsements, restrictions, and vehicle classifications.

Important: Since much of the data extracted by the ID document model is personal, it is of a sensitive nature and covered by data protection laws in most jurisdictions. Be sure that you have the permission of the individual to store their data and comply with all legal requirements in the way you handle this information.

Using the business card model

Business cards are a popular way to exchange contact information quickly and often include branding, unusual fonts, and graphic design elements. Fields that the business card model can extract include:

- First and last names.
- Postal addresses.
- Email and website addresses.
- Various telephone numbers.

Using other prebuilt models

Azure AI Document Intelligence offers several prebuilt models, with new models being released regularly. Before training a custom model, it's worth verifying if your use case can be analyzed accurately with one of these prebuilt models. Using a prebuilt model will benefit from rigorous testing, updated model versions, and reduced cost compared to a custom model.

Learn more

- [Azure AI Document Intelligence model overview](#)

Exercise - Analyze a document using Azure AI Document Intelligence

Use prebuilt Document Intelligence models

In this exercise, you'll set up an Azure AI Document Intelligence resource in your Azure subscription. You'll use both the Azure AI Document Intelligence Studio and C# or Python to submit forms to that resource for analysis.

Module assessment

1. You have a large set of documents with varying structures that contain customer name and address information. You want to extract entities for each customer. Which prebuilt model should you use? **General document model**.
2. You are using the prebuilt layout model to analyze a document with many checkboxes. You want to find out whether each box is checked or empty. What object should you use in the returned JSON code? **Selection marks**.
3. You submit a Word document to the Azure AI Document Intelligence general document model for analysis but you receive an error. The file is A4 size, contains 1 MB of data, and is not password-protected. How should you resolve the error? **Convert the document to PDF format**.

Summary

There are many document types that are common to most business and Azure AI Document Intelligence includes prebuilt models to handle them. If you have a collection of such forms that you want to analyze, you can extract data by using these prebuilt models and you don't have to train your own models. You can get up and running very quickly by submitting photos and scans to the most appropriate prebuilt model.

Now that you've completed this module, you can:

- Identify business problems that you can solve by using prebuilt models in Azure AI Document Intelligence.
- Analyze forms by using the General Document, Read, and Layout models.
- Analyze forms by using financial, ID, and tax prebuilt models.

Learn more

- [What is Azure AI Document Intelligence?](#)
- [Azure AI Document Intelligence models](#)
- [Language support for Azure AI Document Intelligence](#)
- [Azure AI Document Intelligence read model](#)
- [Azure AI Document Intelligence general document model](#)
- [Azure AI Document Intelligence layout model](#)
- [Azure AI Document Intelligence invoice model](#)
- [Azure AI Document Intelligence receipt model](#)
- [Azure AI Document Intelligence ID document model](#)
- [Azure AI Document Intelligence business card model](#)
- [Azure AI Document Intelligence W-2 model](#)

5.4 Extract data from forms with Azure Document intelligence

Document intelligence **uses machine learning technology to identify and extract key-value pairs and table data from form documents** with accuracy, at scale. This module teaches you how to use the **Azure Document intelligence cognitive service**.

Learning objectives

In this module, you learn how to:

- Identify how **Document intelligence's layout service, prebuilt models, and custom models** can automate processes.
- Use Document intelligence's capabilities with **SDKs, REST API, and Document Intelligence Studio**.
- Develop and test custom models.

Introduction

Forms are used to communicate information in every industry, every day. Many people still manually **extract data from forms** to exchange information.

Consider some of the instances when a person needs to process form data:

- When filing claims
- When enrolling new patients in an online management system
- When entering data from receipts to an expense report
- When reviewing an operations report for anomalies
- When selecting data from a report to give to a stakeholder

Without AI services, people need to manually sort through form documents to identify important information and then manually reenter data to record it. Some may also need to complete these tasks in real-time with a customer.

Azure Document Intelligence services provide the building blocks for automation by using intelligent services to extract data at scale and with accuracy. **Azure Document Intelligence is a Vision API that extracts key-value pairs and table data from form documents**.

Uses of the Azure Document Intelligence service include:

- Process automation
- Knowledge mining

- Industry-specific applications

In this module, you'll learn how to:

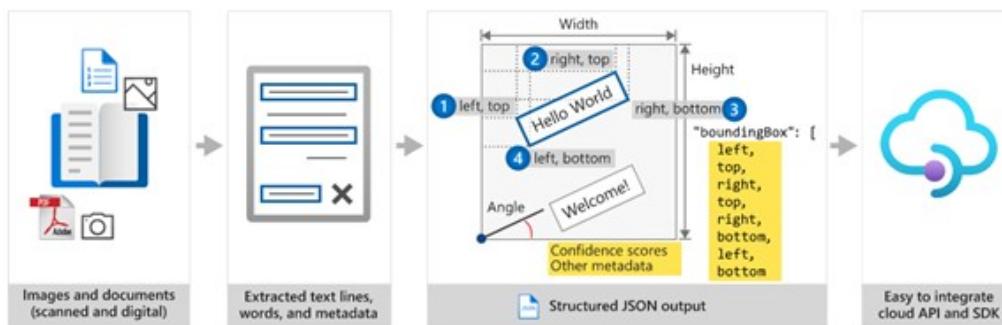
- Identify how Azure Document Intelligence's document analysis, prebuilt, and custom models can automate processes
- Use **Azure Document Intelligence's Optical Character Recognition (OCR) capabilities with SDKs and REST API.**
- Develop and test a custom Azure Document Intelligence model

To complete this module, you'll need a Microsoft Azure subscription. If you don't already have one, you can sign up for a free trial at <https://azure.microsoft.com>.

What is Azure Document Intelligence?

Azure Document Intelligence is one of many Azure AI Services, cloud-based artificial intelligence (AI) services with *REST APIs and client library SDKs* that can be used to build intelligence into your applications.

Azure Document Intelligence uses **Optical Character Recognition (OCR) capabilities and deep learning models** to **extract text, key-value pairs, selection marks, and tables from documents**.



OCR captures document structure by **creating bounding boxes around detected objects in an image**. The locations of the bounding boxes are recorded as coordinates in relation to the rest of the page. Azure Document Intelligence services return bounding box data and other information in a structured form with the relationships from the original file.

Bounding Boxes On Invoice



JSON Response

```
"boundingBox": [136, 139, 351, 138, 351, 166, 136, 166], "text": "Purchase Order", "appearance": { "style": { "name": "other", "confidence": 0.878 }}
```

To build a high-accuracy model from scratch, people need to build deep learning models, use a large amount of compute resources, and face long model training times. These factors could make a project infeasible. Azure Document Intelligence provides underlying models that have been trained on thousands of form examples. The underlying models enable you to do high-accuracy data extraction from your forms with little to no model training.

Azure Document Intelligence service components

Azure Document Intelligence is composed of the following services:

- **Document analysis models:** which *take an input of JPEG, PNG, PDF, and TIFF files and return a JSON file with the location of text in bounding boxes, text content, tables, selection marks (also known as checkboxes or radio buttons), and document structure.*
- **Prebuilt models:** which detect and extract information from document images and return the extracted data in a structured JSON output. Azure Document Intelligence currently supports prebuilt models for several forms, including:
 - W-2 forms
 - Invoices
 - Receipts
 - ID documents
 - Business cards
- **Custom models:** custom models extract data from forms specific to your business. Custom models can be trained through the Azure Document Intelligence Studio.

Note: Some Azure Document Intelligence features are in preview, as of the time this content was authored, and as a result, features and usage details might change. You should refer to the [official page](#) for up-to-date information.

Access services

You can access Azure Document Intelligence services in several ways. These options include using:

- A **REST API**
- Client library **SDKs**
- **Azure Document Intelligence Studio**
- **Azure AI Foundry**

Tip: This module's exercise focuses on the Python and .NET SDKs. The underlying REST services can be used by any language.

Check out the [documentation](#) for quick start guides on all the available SDKs and the REST API.

Get started with Azure Document Intelligence

To start a project with Azure Document Intelligence services, you need an Azure resource and selection of form files for data extraction.

Subscribe to a resource

You can access Azure Document Intelligence services via:

- An **Azure AI Service resource**: a *multi-service subscription key* (used across multiple Azure AI Services)
- OR
- An **Azure Document Intelligence resource**: a *single-service subscription key* (used only with a specific Azure AI Service)

Note: Create an **Azure AI Services resource** if you plan to access multiple Azure AI services **under a single endpoint/key**. For Azure Document Intelligence access only, create an Azure Document Intelligence resource.

Note that **you need a single-service resource if you intend to use Microsoft Entra authentication**.

You can subscribe to a service in the Azure portal or with the Azure Command Line Interface (CLI). You can learn more about the CLI commands [here](#).

Understand Azure Document Intelligence file input requirements

Azure Document Intelligence works on input documents that meet these requirements:

- Format must be **JPG, PNG, BMP, PDF (text or scanned), or TIFF**.
- The file size must be **less than 500 MB** for paid (S0) tier and **4 MB** for free (F0) tier.
- Image dimensions must be **between 50 x 50 pixels and 10,000 x 10,000 pixels**.
- The total size of the training data set must be **500 pages or less**.

More input requirements can be found in the [documentation](#) for specific models.

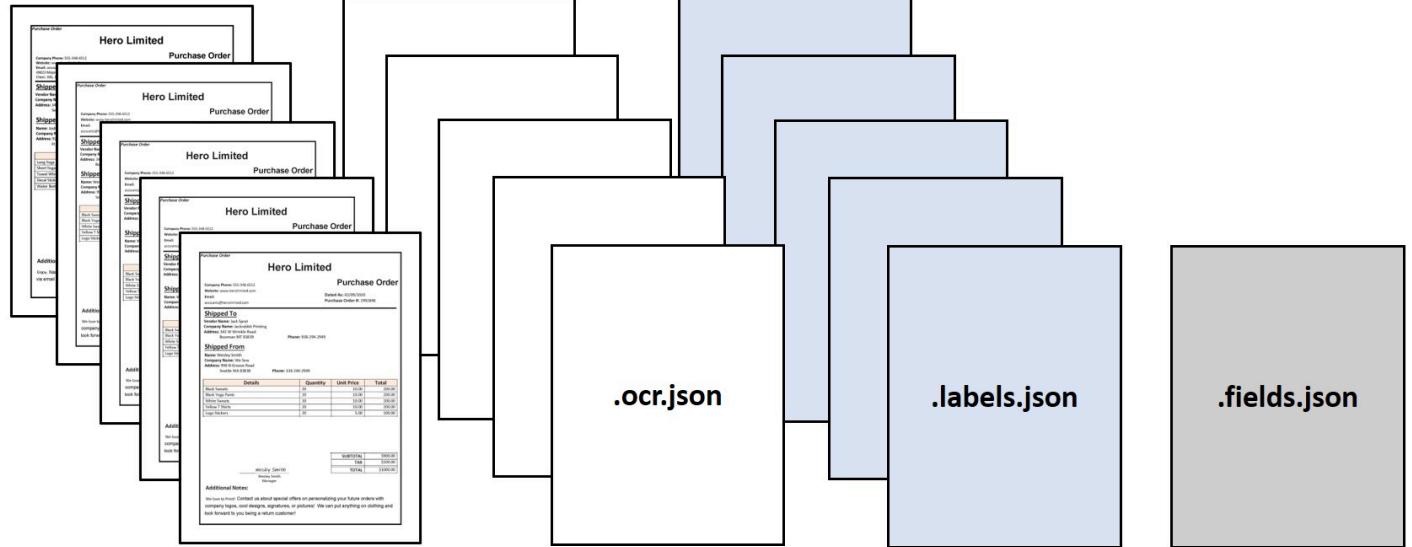
Decide what component of Azure Document Intelligence to use

After you collect your files, decide what you need to accomplish.

Use case	Recommended features to use
Use OCR capabilities to capture document analysis	Use the Layout model, Read model, or General Document model .
Create an application that extracts data from W-2s, Invoices, Receipts, ID documents, Health insurance, vaccination, and business cards	Use a prebuilt model . These models don't need to be trained. Azure Document Intelligence services analyze the documents and return a JSON output .
Create an application to extract data from your industry-specific forms	Create a custom model . This model needs to be trained on sample documents. After you train the custom model, it can analyze new documents and return a JSON output .

Train custom models

Azure's **Azure Document Intelligence service supports supervised machine learning**. You can train custom models and create composite models with form documents and JSON documents that contain labeled fields.



Examples of files needed for supervised (labeled) custom form model training

To train a custom model:

- Store sample forms in an **Azure blob container**, along with **JSON files containing layout and label field information**.
 - You can generate an **ocr.json** file for each sample form using the Azure Document Intelligence's Analyze document function. Additionally, you need a single **fields.json** file describing **the fields you want to extract**, and a **labels.json** file for **each sample form mapping the fields to their location in that form**.
- Generate a **shared access security (SAS) URL** for the container.
- Use the **Build model REST API function (or equivalent SDK method)**.
- Use the **Get model REST API function (or equivalent SDK method)** to get **the trained model ID**.

OR

- Use the Azure Document Intelligence Studio to label and train. There are two types of underlying models for custom forms: custom template models or custom neural models.
 - **Custom template models** accurately extract **labeled key-value pairs, selection marks, tables, regions, and signatures from documents**. Training only takes a few minutes, and more than 100 languages are supported.
 - **Custom neural models** are deep learned models that combine layout and language features to accurately **extract labeled fields from documents**. This model is **best for semi-structured or unstructured documents**.

Use Azure Document Intelligence models

Using the API

To extract form data using a custom model, use the analyze document function of either a **supported SDK, or the REST API**, while **supplying model ID** (generated during model training). This function starts the form analysis, which you can then request the result to get the analysis.

Example code to call your model:

```
endpoint = "YOUR_DOC_INTELLIGENCE_ENDPOINT"
key = "YOUR_DOC_INTELLIGENCE_KEY"

model_id = "YOUR_CUSTOM_BUILT_MODEL_ID"
formUrl = "YOUR_DOCUMENT"

document_analysis_client = DocumentAnalysisClient(    endpoint=endpoint, credential=AzureKeyCredential(key) )

# Make sure your document's type is included in the list of document types the custom model can analyze
task = document_analysis_client.begin_analyze_document_from_url(model_id, formUrl)
result = task.result()
```

A successful JSON response contains **analyzeResult** that contains **the content extracted** and **an array of pages containing information about the document content**.

Example analyze document JSON response:

Explore the documentation for [supported language quickstarts](#).

Understanding confidence scores

If the confidence values of the **analyzeResult** are low, try to improve the quality of your input documents.

You want to make sure that the form you're analyzing has a similar appearance to forms in the training set if the confidence values are low. If the form appearance varies, **consider training more than one model, with each model focused on one form format**.

Depending on the use case, you might find that **a confidence score of 80% or higher is acceptable for a low-risk application. For more sensitive cases, like reading medical records or billing statements, a score of 100% is recommended**.

Use the Azure Document Intelligence Studio

In addition to SDKs and the REST API, **Azure Document Intelligence** services can be accessed through a user interface called the **Azure Document Intelligence Studio**, an online tool for visually exploring, understanding, and

integrating features from the Azure Document Intelligence service. The Studio can be used to **analyze form layouts, extract data from prebuilt models, and train custom models**.

Azure AI | Document Intelligence Studio

① Azure Form Recognizer is now Azure AI Document Intelligence. [Learn more](#) about the latest updates to the service and the Studio experience.

Form Recognizer Studio

Get started with Document Intelligence Studio

Extract text, key-value pairs, tables, and structures from forms and documents using common layouts and prebuilt models, or create your own custom models. [Learn more](#)

Document analysis

Extract text, tables, structure, key-value pairs, and named entities from documents.



Read
Extract printed and handwritten text along with barcodes, formulas and font styles from images and documents.
[Try it out](#)



Layout
Extract tables, check boxes, and text from forms and documents.
[Try it out](#)



General documents
Extract key value pairs and structure like tables and selection marks from any form or document.
[Try it out](#)

Prebuilt models

Extract data from unique document types using the following prebuilt models.



Invoices
Extract invoice ID, customer details, vendor
[Try it out](#)



Receipts
Extract time and date of the transaction,
[Try it out](#)



Business cards
Extract person name, job title, address,
[Try it out](#)



Identity documents
Extract name, expiration date, machine
[Try it out](#)

The Azure Document Intelligence Studio currently supports the following projects:

- **Document analysis models**
 - **Read:** Extract **printed and handwritten text lines, words, locations, and detected languages from documents and images.**
 - **Layout:** Extract **text, tables, selection marks, and structure information from documents (PDF and TIFF) and images (JPG, PNG, and BMP).**
 - **General Documents:** Extract **key-value pairs, selection marks, and entities from documents.**
- **Prebuilt models**
- **Custom models**

Build Document analysis model projects

To **extract text, tables, structure, key-value pairs, and named entities** with **document analysis models**:

- Create an Azure Document Intelligence or Azure AI Services resource.

- Select either "**Read**", "**Layout**", or "**General Documents**" under the Document analysis models category.
- Analyze your document. You'll need your Azure Document Intelligence or Azure AI service **endpoint and key**.

Build prebuilt model projects

To extract data from common forms with prebuilt models:

- Create an **Azure Document Intelligence** or **Azure AI Services** resource
- Select one of the "**prebuilt models**" including **W-2s, Invoices, Receipts, ID documents, Health insurance, vaccination, and business cards**.
- Analyze your document. You'll need your Azure Document Intelligence or Azure AI service **endpoint and key**.

Build custom model projects

You can use **Azure Document Intelligence Studio's custom service** for the entire process of training and testing custom models.

When you use Azure Document Intelligence Studio to build custom models, the **ocr.json** files, **labels.json** files, and **fields.json** file needed for training are automatically created and stored in your storage account.

To train a custom model and use it to extract data with custom models:

- Create an Azure Document Intelligence or Azure AI Services resource
- Collect **at least 5-6 sample forms** for training and upload them to your storage account container.
- Configure **cross-domain resource sharing (CORS)**. CORS enables Azure Document Intelligence Studio to store labeled files in your storage container.
- **Create a custom model project in Azure Document Intelligence Studio**. You'll need to provide configurations linking your storage container and Azure Document Intelligence or Azure AI Service resource to the project.
- Use Azure Document Intelligence Studio to **apply labels to text**.
- Train your model. Once the model is trained, you'll receive a **Model ID and Average Accuracy for tags**.
- Test your model by analyzing a new form that wasn't used in training.

Exercise - Extract data from custom forms

In this exercise, you'll use the Azure Document Intelligence service to train and test a custom model using the Python or .NET SDK.

To complete the exercise for this module, launch the VM and follow the instructions.

Extract Data from Forms

Suppose a company currently requires employees to manually purchase order sheets and enter the data into a database. They would like you to utilize AI services to improve the data entry process. You decide to build a machine

learning model that will read the form and produce structured data that can be used to automatically update a database.

Azure AI Document Intelligence is an Azure AI service that enables users to build automated data processing software. This software can **extract text, key/value pairs, and tables from form documents using optical character recognition (OCR)**. Azure AI Document Intelligence has **pre-built models for recognizing invoices, receipts, and business cards**. The service also provides the capability to train **custom models**. In this exercise, we will focus on building custom models.

Module assessment

In this module, you have learned how to use the Azure Document Intelligence service to extract data from forms.

Consider the following review questions to check your understanding of the topics discussed in this module.

1. A person plans to use an Azure Document Intelligence prebuilt invoice model. To extract document data using the model and REST API language, what are two calls they need to make to the API? **Analyze Invoice and Get Analyze Invoice Result**.
2. A person needs to build an application that submits expense claims and extracts the merchant, date, and total from scanned receipts. What's the best way to build the application? **Use Azure Document Intelligence's prebuilt receipts model**.
3. A person is building a custom model with Azure Document Intelligence services. What is required to train a model? **Along with the form to analyze, JSON files need to be provided**.

Summary

This module focused on Azure Document Intelligence's **prebuilt service, custom service, and its client library SDKs and REST API**. We also introduced the **Azure Document Intelligence Studio** to label and train your model.

Azure Document Intelligence services can be integrated with other Azure AI Services. For example, you can try using [this tutorial](#) with **Azure Document Intelligence and Cognitive Search**.

Document intelligence is just one part of the overall Vision API in Azure AI Services. Azure Document Intelligence services are ever evolving. You can read about the latest updates [here](#).

5.5 Create a knowledge mining solution with Azure AI Search

Unlock the hidden insights in your data with **Azure AI Search**. In this module, you'll learn how to implement a **knowledge mining solution** that extracts and enriches data, making it searchable and ready for deeper analysis.

Learning objectives

After completing this module, you'll be able to:

- Implement **indexing** with **Azure AI Search**
- Use AI skills to **enrich data in an index**
- Search an index to find relevant information
- Persist extracted information in a knowledge store

Introduction

Azure AI Search is a powerful cloud-based service that enables you to **extract, enrich, and explore information** from a wide variety of data sources. In this module, you'll learn how to build **intelligent search and knowledge mining solutions** using Azure AI Search.

We'll start by introducing the core concepts of Azure AI Search, including how to **connect to data sources and create indexes**. You'll discover how the indexing process works, and how AI skills can be used to enrich your data with insights such as **language detection, key phrase extraction, and image analysis**.

After learning how to implement an index, you'll explore how to **query and filter results** using full-text search.

Finally, you'll see how to **use the knowledge store to persist enriched data** for further analysis and integration with other systems.

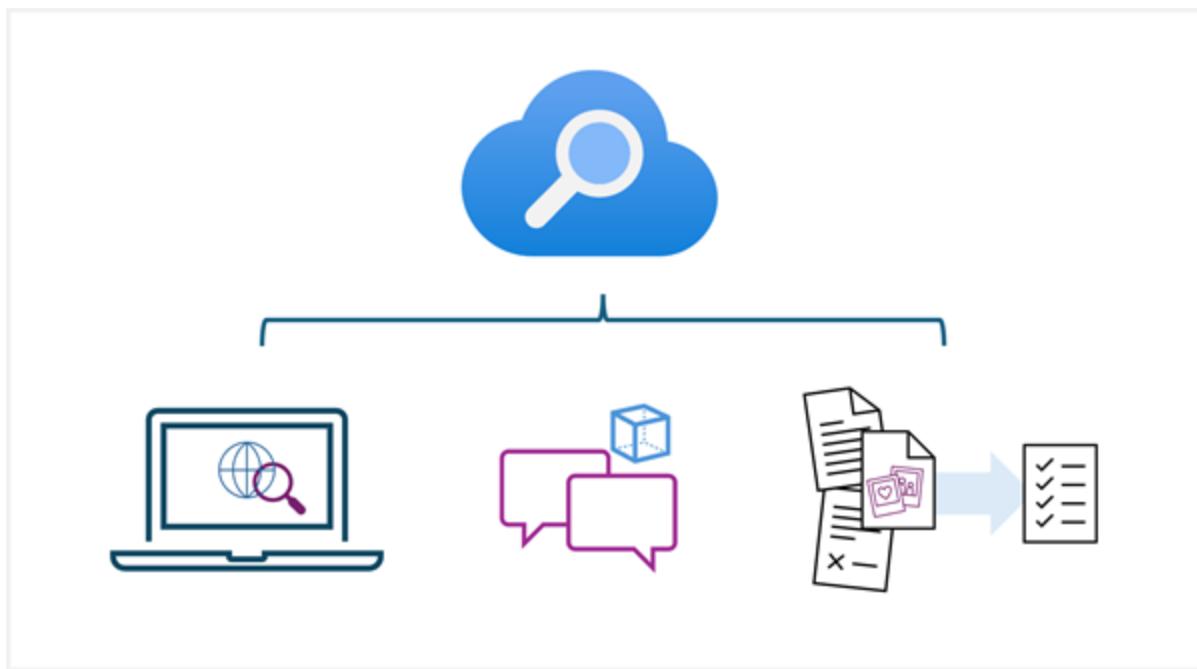
What is Azure AI Search?

Azure AI Search provides a cloud-based solution for **indexing and querying a wide range of data sources**, and **creating comprehensive and high-scale search solutions**. It provides the infrastructure and tools to create search solutions that **extract data from structured, semi-structured, and non-structured documents and other data sources**.

With Azure AI Search, you can:

- **Index documents and data** from a range of sources.
- Use AI skills to **enrich index data**.
- **Store extracted insights in a knowledge store** for analysis and integration.

Azure AI Search indexes contain insights extracted from your data; which can include **text inferred or read using OCR from images, entities and key phrases detection through text analytics**, and other derived information based on AI skills that are integrated into the indexing process.



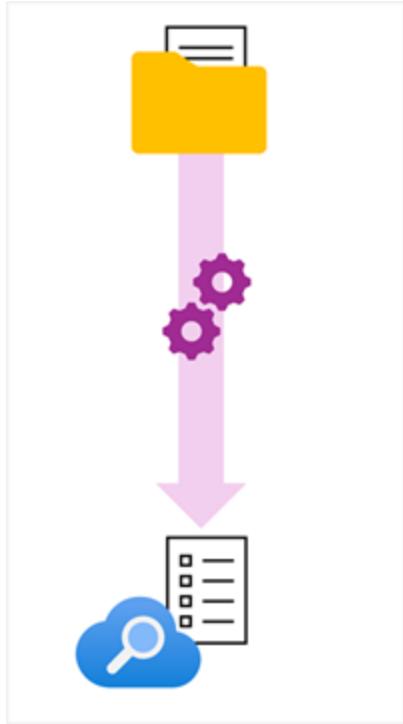
Azure AI search has many applications, including:

- Implementing an enterprise search solution to help employees or customers find information in websites or applications.
- **Supporting retrieval augmented generation (RAG) in generative AI applications** by using vector-based indexes for prompt grounding data.
- Creating **knowledge mining solutions** in which the indexing process is used to infer insights and extract granular data assets from documents to support data analytics.

In this module, we'll focus on Azure AI Search in knowledge mining scenarios.

Extract data with an indexer

At the heart of Azure AI Search solutions is the creation of an index. **An index contains your searchable content and is created and updated, unsurprisingly, by an indexer.**



The indexing process starts with a data source: the storage location of your original data artifacts; for example, an Azure blob store container full of documents, a database, or some other store.

The Indexer automates the extraction and indexing of data fields through an enrichment pipeline, in which it applies *document cracking* to extract the contents of the source documents and applies *incremental steps* to create a hierarchical (JSON-based) document with the required fields for the index definition.

The result is a populated index, which can be queried to return specified fields from documents that match the query criteria.

How documents are constructed during indexing

The indexing process works by creating a document for each indexed entity. *During indexing, an enrichment pipeline iteratively builds the documents that combine metadata from the data source with enriched fields extracted or generated by skills.* You can think of **each indexed document as a JSON**

structure, which initially consists of a document with the index fields you have mapped to fields extracted directly from the source data, like this:

- document
 - metadata_storage_name
 - metadata_author
 - content

When the documents in the data source contain images, you can configure the indexer to extract the image data and place each image in a **normalized_images** collection, like this:

- document
 - metadata_storage_name
 - metadata_author
 - content
 - normalized_images
 - image0
 - image1

Normalizing the image data in this way enables you to use the collection of images as an input for skills that extract information from image data.

Each skill adds fields to the document, so for example a skill that detects the language in which a document is written might store its output in a **language** field, like this:

- document
 - metadata_storage_name
 - metadata_author
 - content
 - normalized_images
 - image0
 - image1
 - language

The document is structured hierarchically, and the skills are applied to a specific context within the hierarchy, enabling you to run the skill for each item at a particular level of the document. For example, you could run an **optical character recognition (OCR)** skill for each image in the normalized images collection to extract any **text** they contain:

- document
 - metadata_storage_name

- metadata_author
- content
- normalized_images
 - image0
 - Text
 - image1
 - Text
- language

The output fields from each skill can be used as inputs for other skills later in the pipeline, which in turn store their outputs in the document structure. For example, we could use a merge skill to combine the original text content with the text extracted from each image to create a new **merged_content** field that contains all of the text in the document, including image text.

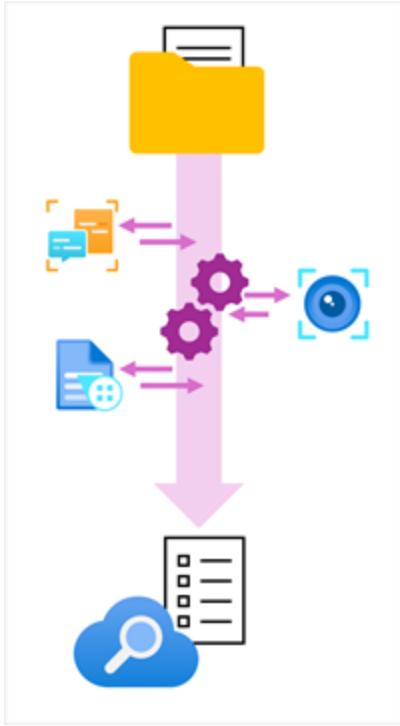
- document
 - metadata_storage_name
 - metadata_author
 - content
 - normalized_images
 - image0
 - Text
 - image1
 - Text
 - language
 - merged_content

The fields in the final document structure at the end of the pipeline are mapped to index fields by the indexer in one of two ways:

- **Fields extracted directly from the source data are all mapped to index fields.** These mappings can be **implicit** (*fields are automatically mapped to index fields with the same name in the index*) or **explicit** (*a mapping is defined to match a source field to an index field, often to rename the field to something more useful or to apply a function to the data value as it is mapped*).
- Output fields from the skills in the skillset are explicitly mapped from their hierarchical location in the output to the target field in the index.

Enrich extracted data with AI skills

The enrichment pipeline that is orchestrated by an indexer uses a skillset of AI skills to create AI-enriched fields. **The indexer applies each skill in order, refining the index document at each step.**



Built-in skills

Azure AI Search provides a collection of built-in skills that you can include in a skillset for your indexer.

Built-in skills include functionality from Azure AI services such as Azure AI Vision and Azure AI Language, enabling you to apply enrichments such as:

- Detecting the **language** that text is written in.
- Detecting and extracting **places, locations, and other entities in the text**.
- Determining and extracting **key phrases** within a body of text.
- **Translating** text.
- Identifying and extracting (or removing) **personally identifiable information (PII)** within the text.
- Extracting **text from images**.
- Generating **captions and tags** to describe images.

To use the built-in skills, your indexer must have access to an Azure AI services resource. You can use a restricted Azure AI search resource that is included in Azure AI Search (and which is limited to indexing 20 or fewer documents) or you can attach an Azure AI services resource in your Azure subscription (which must be in the same region as your Azure AI Search resource).

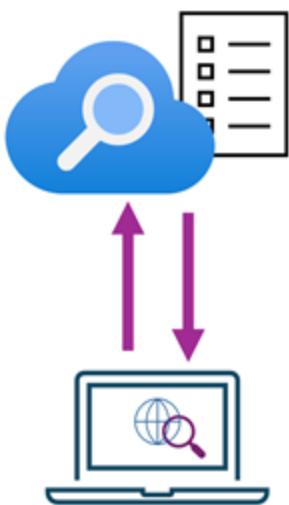
Custom skills

You can further extend the enrichment capabilities of your index by creating custom skills. As the name suggests, custom skills perform custom logic on input data from your index document to return new field values that can be incorporated into the index. Often, **custom skills are "wrappers" around services that are specifically designed to extract data from documents**. For example, you could implement a custom skill as an Azure Function, and use it to pass data from your index document to an **Azure AI Document Intelligence** model, which can extract fields from a form.

Tip: To learn more about using custom skills with Azure AI Search, see [Add a custom skill to an Azure AI Search enrichment pipeline](#) in the Azure AI Search documentation.

Search an index

The index is the searchable result of the indexing process. It consists of a collection of JSON documents, with fields that contain the values extracted during indexing. Client applications can query the index to retrieve, filter, and sort information.



Each index field can be configured with the following attributes:

- **key**: Fields that define a unique key for index records.
- **searchable**: Fields that can be queried using full-text search.

- **filterable**: Fields that can be included in filter expressions to return only documents that **match specified constraints**.
- **sortable**: Fields that can be used to **order** the results.
- **facetable**: Fields that can be used to determine values for facets (user interface elements used to filter the results based on a list of **known field values**).
- **retrievable**: Fields that can be included in search results (by default, all fields are retrievable unless this attribute is explicitly removed).

Full-text search

While you could retrieve index entries based on simple field value matching, most search solutions use full-text search semantics to query an index.

Full-text search describes search solutions that parse text-based document contents to find query terms. Full-text search queries in Azure AI Search are based on the [Lucene query syntax](#), which provides a rich set of query operations for searching, filtering, and sorting data in indexes. Azure AI Search supports two variants of the Lucene syntax:

- **Simple** - An intuitive syntax that makes it easy to perform basic searches that match literal query terms submitted by a user.
- **Full** - An extended syntax that supports complex filtering, regular expressions, and other more sophisticated queries.

Client applications submit queries to Azure AI Search by specifying a search expression along with other parameters that determine how the expression is evaluated and the results returned. Some common parameters submitted with a query include:

- **search** - A search expression that includes the terms to be found.
- **queryType** - The [Lucene syntax](#) to be evaluated (simple or full).
- **searchFields** - The index fields to be searched.
- **select** - The fields to be included in the results.
- **searchMode** - Criteria for including results based on multiple search terms. For example, suppose you search an index of travel-related documents for comfortable hotel. A searchMode value of Any returns documents that contain "comfortable", "hotel", or both; while a searchMode value of All restricts results to documents that contain both "comfortable" and "hotel".

Query processing consists of four stages:

1. **Query parsing**. The search expression is *evaluated and reconstructed as a tree of appropriate subqueries*. Subqueries might include **term queries** (finding specific individual words in the

search expression - for example hotel), **phrase queries** (finding multi-term phrases specified in quotation marks in the search expression - for example, "free parking"), and **prefix queries** (finding terms with a specified prefix - for example air*, which would match airway, air-conditioning, and airport).

2. **Lexical analysis** - The query terms are analyzed and refined based on linguistic rules. For example, *text is converted to lower case and nonessential stopwords (such as "the", "a", "is", and so on) are removed*. Then words are *converted to their root form* (for example, "comfortable" might be simplified to "comfort") and *composite words are split into their constituent terms*.
3. **Document retrieval** - The *query terms are matched against the indexed terms*, and the set of matching documents is identified.
4. **Scoring** - A *relevance score* is assigned to each result based on a term [frequency/inverse document frequency \(TF/IDF\) calculation](#).

Tip: For more information about querying an index, and details about simple and full syntax, see [Query types and composition in Azure AI Search](#) in the Azure AI Search documentation.

It's common in a search solution for users to want to refine query results by filtering and sorting based on field values. Azure AI Search supports both of these capabilities through the search query API.

Filtering results

You can apply filters to queries in two ways:

- By including filter criteria in a simple search expression.
- By providing an OData filter expression as a **\$filter** parameter with a *full* syntax search expression.

You can apply a filter to any *filterable* field in the index.

For example, suppose you want to find documents containing the text *London* that have an **author** field value of *Reviewer*.

You can achieve this result by submitting the following simple search expression:

```
search=London+author='Reviewer'  
queryType=Simple
```

Alternatively, you can use an OData filter in a **\$filter** parameter with a **full Lucene search expression** like this:

```
search=London  
$filter=author eq 'Reviewer'
```

```
queryType=Full
```

Note: OData **\$filter** expressions are case-sensitive!

Filtering with facets

Facets are a useful way to present users with filtering criteria based on field values in a result set. They work best when a field has a small number of discrete values that can be displayed as links or options in the user interface.

To use facets, you must specify **facetable** fields for which you want to retrieve the possible values in an initial query. For example, you could use the following parameters to return all of the possible values for the author field:

```
search=*  
facet=author
```

The results from this query include a collection of discrete facet values that you can display in the user interface for the user to select. Then in a subsequent query, you can use the selected facet value to filter the results:

```
search=*  
$filter=author eq 'selected-facet-value-here'
```

Sorting results

By default, results are sorted based on the relevancy score assigned by the query process, with the highest scoring matches listed first. However, you can override this sort order by including an OData **orderby** parameter that specifies one or more **sortable** fields and a **sort order (asc or desc)**.

For example, to sort the results so that the most recently modified documents are listed first, you could use the following parameter values:

```
search=*  
$orderby=last_modified desc
```

Tip: For more information about using filters, see [Filters in Azure AI Search](#) in the Azure AI Search documentation.

Persist extracted information in a knowledge store

While the index might be considered the primary output from an indexing process, the enriched data it contains might also be useful in other ways. For example:

- Since the index is essentially a collection of JSON objects, each representing an indexed record, it might be useful to *export the objects as JSON files for integration into a data orchestration process for extract, transform, and load (ETL) operations.*
- You may want to *normalize the index records into a relational schema of tables* for analysis and reporting.
- Having *extracted embedded images from documents during the indexing process*, you might want to save those images as files.

Azure AI Search supports these scenarios by enabling you to define a **knowledge** store in the skillset that encapsulates your enrichment pipeline. The knowledge store consists of *projections of the enriched data*, which can be *JSON objects, tables, or image files*. When an indexer runs the pipeline to create or update an index, the projections are generated and persisted in the knowledge store.

Tip: To learn more about using a knowledge store, see [Knowledge store in Azure AI Search](#) in the Azure AI Search documentation.

Exercise - Create a knowledge mining solution

It's time to put what you've learned into practice!

In this exercise, you use Azure AI Search to extract and enrich information from documents into a searchable index and a knowledge store.

Create an knowledge mining solution

In this exercise, you use AI Search to index a set of documents maintained by Margie's Travel, a fictional travel agency. The indexing process involves using AI skills to extract key information to make them searchable, and generating a knowledge store containing data assets for further analysis.

Module assessment

1. Which component of an Azure AI Search solution is scheduled to extract and enrich data to populate an index? **Indexer**.
2. Which service supports built-in AI skills in Azure AI Search? **Azure AI Services**
3. Which kind of projection results in a relational data schema for extracted fields? **Table**

Summary

In this module, you've learned how Azure AI Search enables you to build **intelligent search and knowledge mining** solutions by **indexing and enriching data** from various sources. You explored *the indexing process, the use of AI skills for data enrichment, and how to persist enriched data in a knowledge store for further analysis and integration*.

With these skills, you're now equipped to design and implement solutions that unlock valuable insights from your data using Azure AI Search.

Tip: To learn more about Azure AI Search, see the [Azure AI Search documentation](#).