Get started with Document Intelligence

Article • 12/24/2023

ⓘ **Important**

- Azure Cognitive Services Form Recognizer is now Azure AI Document Intelligence.
- Some platforms are still awaiting the renaming update.
- All mention of Form Recognizer or Document Intelligence in our documentation refers to the same Azure service.

This content applies to: ✓ **v4.0 (preview) Earlier versions:** ✓ v3.1 (GA) ✓ v3.0 (GA)

- Get started with Azure AI Document Intelligence latest preview version (2023-10-31-preview).

- Azure AI Document Intelligence / Form Recognizer is a cloud-based Azure AI service that uses machine learning to extract key-value pairs, text, tables and key data from your documents.

- You can easily integrate document processing models into your workflows and applications by using a programming language SDK or calling the REST API.

- For this quickstart, we recommend that you use the free service while you're learning the technology. Remember that the number of free pages is limited to 500 per month.

To learn more about the API features and development options, visit our Overview page.

Form Recognizer → to → Document Intelligence migration guide Client library |SDK reference | REST API reference | Package (PyPi) | Samples | Supported REST API versions

In this quickstart you'll, use the following features to analyze and extract data and values from forms and documents:

- **Layout**—Analyze and extract tables, lines, words, and selection marks like radio buttons and check boxes, and key-value pairs, without the need to train a model.

- **Prebuilt Invoice**—Analyze and extract common fields from specific document types using a pretrained model.

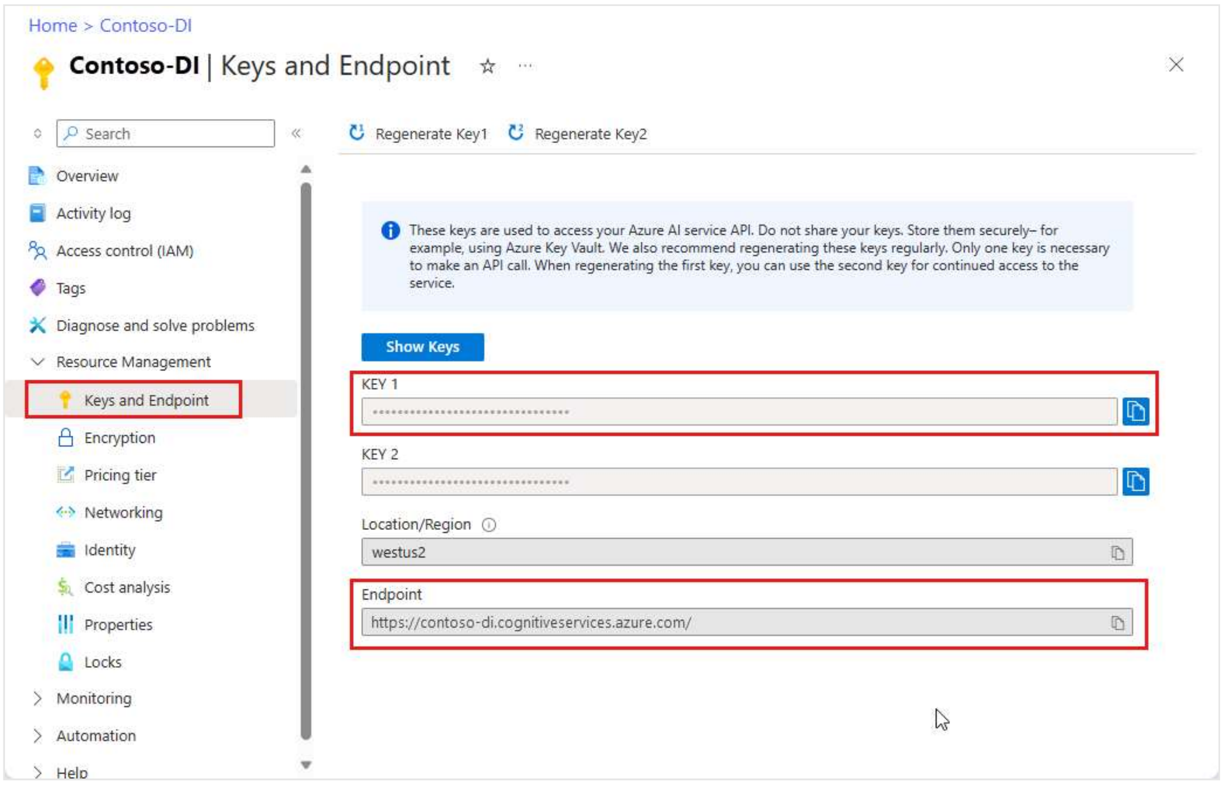
Prerequisites

- Azure subscription - Create one for free
- Python 3.7 or later
  - Your Python installation should include pip. You can check if you have pip installed by running `pip --version` on the command line. Get pip by installing the latest version of Python.
- The latest version of Visual Studio Code or your preferred IDE. For more information, *see* Getting Started with Python in Visual Studio Code.
- An Azure AI services or Document Intelligence resource. Once you have your Azure subscription, create a single-service or multi-service Document Intelligence resource, in the Azure portal, to get your key and endpoint. You can use the free pricing tier (`F0`) to try the service, and upgrade later to a paid tier for production.

Tip

Create an Azure AI services resource if you plan to access multiple Azure AI services under a single endpoint/key. For Document Intelligence access only, create a Document Intelligence resource. Please note that you'll need a single-service resource if you intend to use **Microsoft Entra authentication**.

- After your resource deploys, select **Go to resource**. You need the key and endpoint from the resource you create to connect your application to the Document Intelligence API. You paste your key and endpoint into the code later in the quickstart:



These keys are used to access your Azure AI service API. Do not share your keys. Store them securely– for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.

KEY 1: ••••••••••••••••••••••••••••••••

KEY 2: ••••••••••••••••••••••••••••••••

Location/Region: westus2

Endpoint: https://contoso-di.cognitiveservices.azure.com/

Set up

Open a terminal window in your local environment and install the Azure AI Document Intelligence client library for Python with pip:

Console

```
pip install azure-ai-documentintelligence==1.0.0b1
```

Create your Python application

To interact with the Document Intelligence service, you need to create an instance of the `DocumentIntelligenceClient` class. To do so, you create an `AzureKeyCredential` with your `key` from the Azure portal and a `DocumentIntelligenceClient` instance with the `AzureKeyCredential` and your Document Intelligence `endpoint`.

1. Create a new Python file called **doc_intel_quickstart.py** in your preferred editor or IDE.

2. Open the **doc_intel_quickstart.py** file and select one of the following code samples to copy and paste into your application:

- Layout
- Prebuilt Invoice

ⓘ **Important**

Remember to remove the key from your code when you're done, and never post it publicly. For production, use a secure way of storing and accessing your credentials like **Azure Key Vault**. For more information, *see* Azure AI services **security**.

Layout model

Extract text, selection marks, text styles, table structures, and bounding region coordinates from documents.

- ✓ For this example, you'll need a **document file from a URL**. You can use our sample document for this quickstart.
- ✓ We've added the file URL value to the `formUrl` variable in the `analyze_layout` function.
- ✓ To analyze a given file at a URL, you'll use the `begin_analyze_document_from_url` method and pass in `prebuilt-layout` as the model Id. The returned value is a `result` object containing data about the submitted document.

Add the following code sample to your doc_intel_quickstart.py application. Make sure you update the key and endpoint variables with values from your Azure portal Document Intelligence instance:

Python

```python
# import libraries
import os
from azure.core.credentials import AzureKeyCredential
from azure.ai.documentintelligence import DocumentIntelligenceClient

# set `<your-endpoint>` and `<your-key>` variables with the values from the
Azure portal
endpoint = "<your-endpoint>"
key = "<your-key>"


def analyze_layout():
    # sample document
    formUrl = "https://raw.githubusercontent.com/Azure-Samples/cognitive-ser-
vices-REST-api-samples/master/curl/form-recognizer/sample-layout.pdf"

    document_intelligence_client = DocumentIntelligenceClient(
        endpoint=endpoint, credential=AzureKeyCredential(key)
    )

    poller = document_intelligence_client.begin_analyze_document_from_url(
        "prebuilt-layout", formUrl
    )
    result = poller.result()

    if any([style.is_handwritten for style in result.styles]):
        print("Document contains handwritten content")
    else:
        print("Document does not contain handwritten content")

    for page in result.pages:
        print(f"----Analyzing layout from page #{page.page_number}----")
        print(
            f"Page has width: {page.width} and height: {page.height}, measured
with unit: {page.unit}"
        )

        for line_idx, line in enumerate(page.lines):
            words = get_words(page, line)
            print(
                f"...Line # {line_idx} has word count {len(words)} and text
'{line.content}' "
                f"within bounding polygon '{line.polygon}'"
            )

            for word in words:
                print(
                    f"......Word '{word.content}' has a confidence of
{word.confidence}"
                )

        for selection_mark in page.selection_marks:
            print(
                f"Selection mark is '{selection_mark.state}' within bounding
polygon "
                f"'{selection_mark.polygon}' and has a confidence of
{selection_mark.confidence}"
            )

    for table_idx, table in enumerate(result.tables):
        print(
            f"Table # {table_idx} has {table.row_count} rows and "
            f"{table.column_count} columns"
        )
        for region in table.bounding_regions:
            print(
                f"Table # {table_idx} location on page: {region.page_number} is
{region.polygon}"
            )
        for cell in table.cells:
            print(
                f"...Cell[{cell.row_index}][{cell.column_index}] has text
'{cell.content}'"
            )
            for region in cell.bounding_regions:
                print(
                    f"...content on page {region.page_number} is within bound-
ing polygon '{region.polygon}'"
                )

    print("----------------------------------------")


if __name__ == "__main__":
    analyze_layout()
```

Run the application

After you add a code sample to your application, build and run your program:

1. Navigate to the folder where you have your **doc_intel_quickstart.py** file.
2. Type the following command in your terminal:

Console

```
python doc_intel_quickstart.py
```

Prebuilt model

Analyze and extract common fields from specific document types using a prebuilt model. In this example, we analyze an invoice using the **prebuilt-invoice** model.

Tip

You aren't limited to invoices—there are several prebuilt models to choose from, each of which has its own set of supported fields. The model to use for the analyze operation depends on the type of document to be analyzed. See **model data extraction**.

- ✓ Analyze an invoice using the prebuilt-invoice model. You can use our sample invoice document for this quickstart.
- ✓ We've added the file URL value to the `invoiceUrl` variable at the top of the file.
- ✓ To analyze a given file at a URI, you'll use the `begin_analyze_document_from_url` method and pass `prebuilt-invoice` as the model Id. The returned value is a `result` object containing data about the submitted document.
- ✓ For simplicity, all the key-value pairs that the service returns are not shown here. To see the list of all supported fields and corresponding types, see our Invoice concept page.

Add the following code sample to your doc_intel_quickstart.py application. Make sure you update the key and endpoint variables with values from your Azure portal Document Intelligence instance:

Python

```python
# import libraries
import os
from azure.core.credentials import AzureKeyCredential
from azure.ai.documentintelligence import DocumentIntelligenceClient


# set `<your-endpoint>` and `<your-key>` variables with the values from the
Azure portal
endpoint = "<your-endpoint>"
key = "<your-key>"

def analyze_invoice():
    # sample document

    invoiceUrl = "https://raw.githubusercontent.com/Azure-Samples/cognitive-
services-REST-api-samples/master/curl/form-recognizer/sample-invoice.pdf"

    document_intelligence_client = DocumentIntelligenceClient(
        endpoint=endpoint, credential=AzureKeyCredential(key)
    )

    poller = document_intelligence_client.begin_analyze_document_from_url(
        "prebuilt-invoice", invoiceUrl
    )
    invoices = poller.result()

    for idx, invoice in enumerate(invoices.documents):
        print(f"--------Analyzing invoice #{idx + 1}--------")
        vendor_name = invoice.fields.get("VendorName")
        if vendor_name:
            print(
                f"Vendor Name: {vendor_name.get('content')} has confidence:
{vendor_name.get('confidence')}"
            )
        vendor_address = invoice.fields.get("VendorAddress")
        if vendor_address:
            print(
                f"Vendor Address: {vendor_address.get('content')} has confi-
dence: {vendor_address.get('confidence')}"
            )
        vendor_address_recipient = invoice.fields.get("VendorAddressRecipient")
        if vendor_address_recipient:
            print(
                f"Vendor Address Recipient: {vendor_address_recipient.get('con-
tent')} has confidence: {vendor_address_recipient.get('confidence')}"
            )
        customer_name = invoice.fields.get("CustomerName")
        if customer_name:
            print(
                f"Customer Name: {customer_name.get('content')} has confidence:
{customer_name.get('confidence')}"
            )
        customer_id = invoice.fields.get("CustomerId")
        if customer_id:
            print(
                f"Customer Id: {customer_id.get('content')} has confidence:
{customer_id.get('confidence')}"
            )
        customer_address = invoice.fields.get("CustomerAddress")
        if customer_address:
            print(
                f"Customer Address: {customer_address.get('content')} has con-
fidence: {customer_address.get('confidence')}"
            )
        customer_address_recipient =
invoice.fields.get("CustomerAddressRecipient")
        if customer_address_recipient:
            print(
                f"Customer Address Recipient:
{customer_address_recipient.get('content')} has confidence:
{customer_address_recipient.get('confidence')}"
            )
        invoice_id = invoice.fields.get("InvoiceId")
        if invoice_id:
            print(
                f"Invoice Id: {invoice_id.get('content')} has confidence:
{invoice_id.get('confidence')}"
            )
        invoice_date = invoice.fields.get("InvoiceDate")
        if invoice_date:
            print(
                f"Invoice Date: {invoice_date.get('content')} has confidence:
{invoice_date.get('confidence')}"
            )
        invoice_total = invoice.fields.get("InvoiceTotal")
        if invoice_total:
            print(
                f"Invoice Total: {invoice_total.get('content')} has confidence:
{invoice_total.get('confidence')}"
            )
        due_date = invoice.fields.get("DueDate")
        if due_date:
            print(
                f"Due Date: {due_date.get('content')} has confidence:
{due_date.get('confidence')}"
            )
        purchase_order = invoice.fields.get("PurchaseOrder")
        if purchase_order:
            print(
                f"Purchase Order: {purchase_order.get('content')} has confi-
dence: {purchase_order.get('confidence')}"
            )
        billing_address = invoice.fields.get("BillingAddress")
        if billing_address:
            print(
                f"Billing Address: {billing_address.get('content')} has confi-
dence: {billing_address.get('confidence')}"
            )
        billing_address_recipient =
invoice.fields.get("BillingAddressRecipient")
        if billing_address_recipient:
            print(
                f"Billing Address Recipient:
{billing_address_recipient.get('content')} has confidence:
{billing_address_recipient.get('confidence')}"
            )
        shipping_address = invoice.fields.get("ShippingAddress")
        if shipping_address:
            print(
                f"Shipping Address: {shipping_address.get('content')} has con-
fidence: {shipping_address.get('confidence')}"
            )
        shipping_address_recipient =
invoice.fields.get("ShippingAddressRecipient")
```

```
        if shipping_address_recipient:
            print(
                f"Shipping Address Recipient:
{shipping_address_recipient.get('content')} has confidence:
{shipping_address_recipient.get('confidence')}"
            )
        print("Invoice items:")
        for idx, item in enumerate(invoice.fields.get("Items").get("valueAr-
ray")):
            print(f"...Item #{idx + 1}")
            item_description = item.get("valueObject").get("Description")
            if item_description:
                print(
                    f"......Description: {item_description.get('content')} has
confidence: {item_description.get('confidence')}"
                )
            item_quantity = item.get("valueObject").get("Quantity")
            if item_quantity:
                print(
                    f"......Quantity: {item_quantity.get('content')} has confi-
dence: {item_quantity.get('confidence')}"
                )
            unit = item.get("valueObject").get("Unit")
            if unit:
                print(
                    f"......Unit: {unit.get('content')} has confidence:
{unit.get('confidence')}"
                )
            unit_price = item.get("valueObject").get("UnitPrice")
            if unit_price:
                unit_price_code = (
                    unit_price.get("valueCurrency").get("currencyCode")
                    if unit_price.get("valueCurrency").get("currencyCode")
                    else ""
                )
                print(
                    f"......Unit Price: {unit_price.get('content')}
{unit_price_code} has confidence: {unit_price.get('confidence')}"
                )
            product_code = item.get("valueObject").get("ProductCode")
            if product_code:
                print(
                    f"......Product Code: {product_code.get('content')} has
confidence: {product_code.get('confidence')}"
                )
            item_date = item.get("valueObject").get("Date")
            if item_date:
                print(
                    f"......Date: {item_date.get('content')} has confidence:
{item_date.get('confidence')}"
                )
            tax = item.get("valueObject").get("Tax")
            if tax:
                print(
                    f"......Tax: {tax.get('content')} has confidence:
{tax.get('confidence')}"
                )
            amount = item.get("valueObject").get("Amount")
            if amount:
                print(
                    f"......Amount: {amount.get('content')} has confidence:
{amount.get('confidence')}"
                )
        subtotal = invoice.fields.get("SubTotal")
        if subtotal:
            print(
                f"Subtotal: {subtotal.get('content')} has confidence:
{subtotal.get('confidence')}"
            )
        total_tax = invoice.fields.get("TotalTax")
        if total_tax:
            print(
                f"Total Tax: {total_tax.get('content')} has confidence:
{total_tax.get('confidence')}"
            )
        previous_unpaid_balance = invoice.fields.get("PreviousUnpaidBalance")
        if previous_unpaid_balance:
            print(
                f"Previous Unpaid Balance: {previous_unpaid_balance.get('con-
tent')} has confidence: {previous_unpaid_balance.get('confidence')}"
            )
        amount_due = invoice.fields.get("AmountDue")
        if amount_due:
            print(
                f"Amount Due: {amount_due.get('content')} has confidence:
{amount_due.get('confidence')}"
            )
        service_start_date = invoice.fields.get("ServiceStartDate")
        if service_start_date:
            print(
                f"Service Start Date: {service_start_date.get('content')} has
confidence: {service_start_date.get('confidence')}"
            )
        service_end_date = invoice.fields.get("ServiceEndDate")
        if service_end_date:
            print(
                f"Service End Date: {service_end_date.get('content')} has con-
fidence: {service_end_date.get('confidence')}"
            )
        service_address = invoice.fields.get("ServiceAddress")
        if service_address:
            print(
                f"Service Address: {service_address.get('content')} has confi-
dence: {service_address.get('confidence')}"
            )
        service_address_recipient =
invoice.fields.get("ServiceAddressRecipient")
        if service_address_recipient:
            print(
                f"Service Address Recipient:
{service_address_recipient.get('content')} has confidence:
{service_address_recipient.get('confidence')}"
            )
        remittance_address = invoice.fields.get("RemittanceAddress")
        if remittance_address:
            print(
                f"Remittance Address: {remittance_address.get('content')} has
confidence: {remittance_address.get('confidence')}"
            )
        remittance_address_recipient =
invoice.fields.get("RemittanceAddressRecipient")
        if remittance_address_recipient:
            print(
                f"Remittance Address Recipient:
{remittance_address_recipient.get('content')} has confidence:
{remittance_address_recipient.get('confidence')}"
            )

        print("----------------------------------------")


if __name__ == "__main__":
    analyze_invoice()
```

Run the application

After you add a code sample to your application, build and run your program:

1. Navigate to the folder where you have your **doc_intel_quickstart.py** file.

2. Type the following command in your terminal:

Console

```
python doc_intel_quickstart.py
```

That's it, congratulations!

In this quickstart, you used a document Intelligence model to analyze various forms and documents. Next, explore the Document Intelligence Studio and reference documentation to learn about Document Intelligence API in depth.

Next steps

For an enhanced experience and advanced model quality, try Document Intelligence Studio