Document AI

Installation, Configuration and Setup

Contents

Azure Service Installations

Azure Web App

Select Resource as Web App

Subscription: prod

RG: docai-prod

Name: docai-web-prod

Unique Default Host Name: off

Publish: Code

Runtime stack: Java 17

Java web server stack: Java SE (Embedded Web Server)

OS: Linux

Region: East US 2

Linux Plan: linux-app-service-plan-prod

Pricing Plan:

Pricing Plan: Premium V3 P1V3 (195 minimum ACU/vCPU, 8 GB memory, 2 vCPU)

Application Insights: No

Microsoft Defender for Cloud: Enable

Rest select default

Azure Functions - Python

Hosting Option: Flex Consumption

Subscription: prod

RG: docai-prod

Name: docai-python-func-prod

Region: East US

Runtime stack: Python

Version:3.11

Instance size: 2048 MB

Storage account: docai8func8adls8prod

App Insights: No

Rest select default

Azure Functions - C#

Hosting Option: Flex Consumption

Subscription: prod

RG: docai-prod

Name: docai-csharp-func-prod

Region: East US

Runtime stack: .NET

Version: 8 (LTS), isolated worker model

Instance size: 2048 MB

Storage account: docai8func8adls8prod

App Insights: No

Rest select default

Azure Logic App

Hosting option: Workflow Service Plan

Subscription: prod

RG: docai-prod

Name: docai-logic-app-prod

Region: East US

Windows Plan: windows-app-service-plan-prod

Storage account: docai8logicapp8adls8prod

App Insights: No

Rest select default

Azure Document Intelligence Service

Subscription: non-prod

RG: docai-non-prod

Name: doc-intel-non-prod

Region: East US 2

Pricing Tier: Standard S0

Azure Storage

Subscription: prod

RG: docai-prod

Name: docai8adls8prod

Region: East US

Primary Service: Azure Blob Storage or Azure Data Lake Storage Gen 2

Redundancy: LRS

Hierarchical Namespace: Yest

Soft Deletes: disable for blobs, containers and file shares

Azure Storage (for Document Intelligence)

Subscription: non-prod

RG: docai-non-prod

Name: docai8doc8intel8blob

Region: East US

Primary Service: Azure Blob Storage or Azure Data Lake Storage Gen 2

Redundancy: LRS

Hierarchical Namespace: Yest

Soft Deletes: disable for blobs, containers and file shares

Azure AI Service

Subscription: non-prod

RG: docai-non-prod

Name: ai-non-prod

Region: East US

Pricing Tier: Standard S0

Azure Open AI

Subscription: non-prod

RG: docai-non-prod

Name: aoai-non-prod

Region: East US

Pricing Tier: Standard S0

Azure Open AI (for Video)

Subscription: non-prod

RG: docai-non-prod

Name: aoai-vision-non-prod

Region: West US

Pricing Tier: Standard S0

Configuration

Azure Logic App

1. Find out the storage account for this Logic App setup during creation. In this case it is docai8logicapp8adls8prod
2. Go that Storage account in the Portal, and then go to Settings->Configuration and enable 'Allow storage account key access'

Azure Function App

1. Find out the storage account for this Function App setup during creation. In this case it is docai8func8adls8prod

2. Go that Storage account in the Portal, and then go to Settings->Configuration and enable 'Allow storage account key access'

Azure Open AI

From Azure Open AI Studio, deploy

Model: gpt-4o

Name: gpt-4o

Model Version: 2024-08-06

Deployment Type: Standard

Capacity: 150k TPM

Azure Open AI (for video)

From Azure Open AI Studio, deploy

Model: gpt-4 vision-preview

Name: gpt-4-vision

Model Version: vision-preview

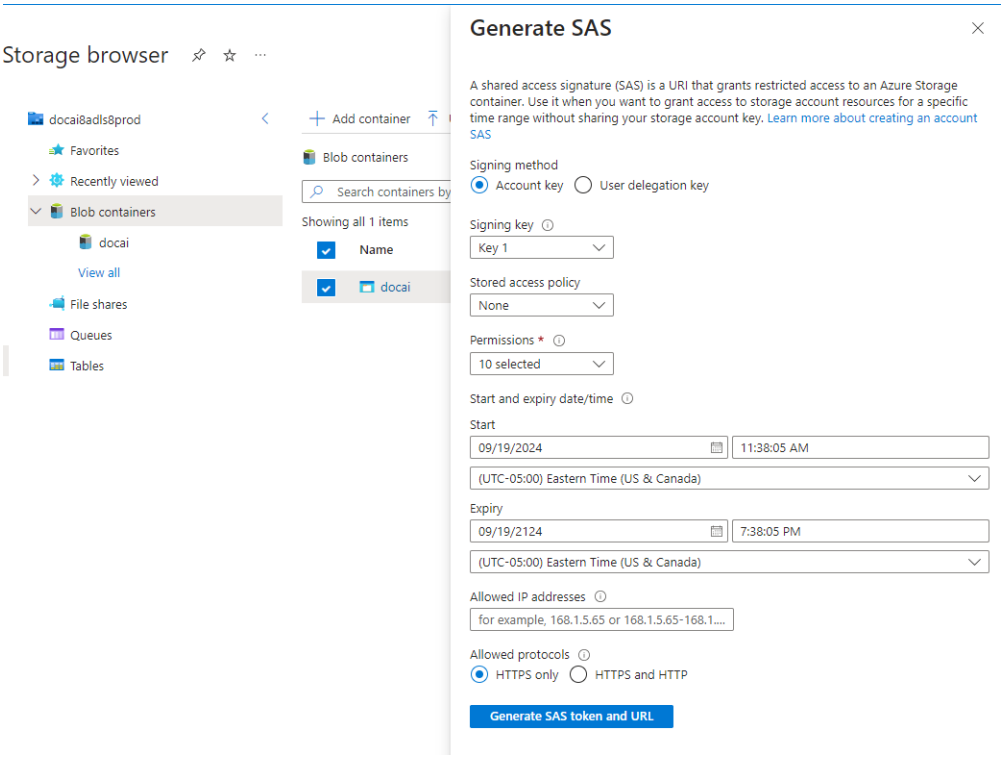
Deployment Type: Standard

Capacity: 80k TPM

API version: 2024-02-15-preview (get this from the Sample Code in Azure Open AI Studio for this model)

Azure Storage docai8adls8prod

1. Create a blob container named **docai**
2. Generate the SAS token



3. From MSDOS command line go to the DocAI\code\python folder and run the following:
    - env.bat
    - cd cli
    - python base64Tool.py -e "<the Blob SAS Token>"
4. Save the base 64 encoded token returned and use it as the value for the BLOB_STORE_SAS_TOKEN environment variable for the Azure Web App and the Azure Python Function App
5. In the Portal, go to Access Control (IAM) and add 'Storage Blob Data Owner' role to your admin user. This will allow the admin to view files using the Storage browser from the Portal.
6. In Portal, go to Settings->Configuration and enable 'Allow storage account key access'

Azure Document Intelligence Service

1. In the Storage Account docai8doc8intel8blob, create a Blob container named doc-intel-models
2. Upload all the files and folder in <GitHub DocAI project root>/data/doc-intel-models into the doc-intel-models blob container

3. Go to Document Intelligence Studio
4. Select Custom classification models
5. Create 1 new Project
    - Name docai-classifier
    - Service resource
        - Subscription: non-prod
        - RG: docai-non-prod
        - Document Intelligence Service: doc-intel-non-prod
        - API version 2024-07-31 (Preview)
    - Training Data Source
        - Subscription: non-prod
        - RG: docai-non-prod
        - Storage account: docai8doc8intel8blob
        - Blob container: doc-intel-models
        - Folder path: blank
6. Now Train the model
    - Model Name: docai-classifier-v1
7. Select Custom extraction models
8. Create 3 new Projects
    - Name: AutoInsuranceClaim (or CommercialInsuranceApplication-AcordForm or WorkersCompensationApplication-AcordForm)
    - Service resource
        - Subscription: non-prod
        - RG: docai-non-prod
        - Document Intelligence Service: doc-intel-non-prod
        - API version 2024-07-31 (Preview)
    - Training Data Source
        - Subscription: non-prod
        - RG: docai-non-prod

Storage account: docai8doc8intel8blob

Blob container: doc-intel-models

Folder path: sample-auto-insurance-claims-docs/training (or sample-commercial-insurance-applications/training or sample-workers-compensation-applications/training)

9. Now Train each one of the models

   Model Names:

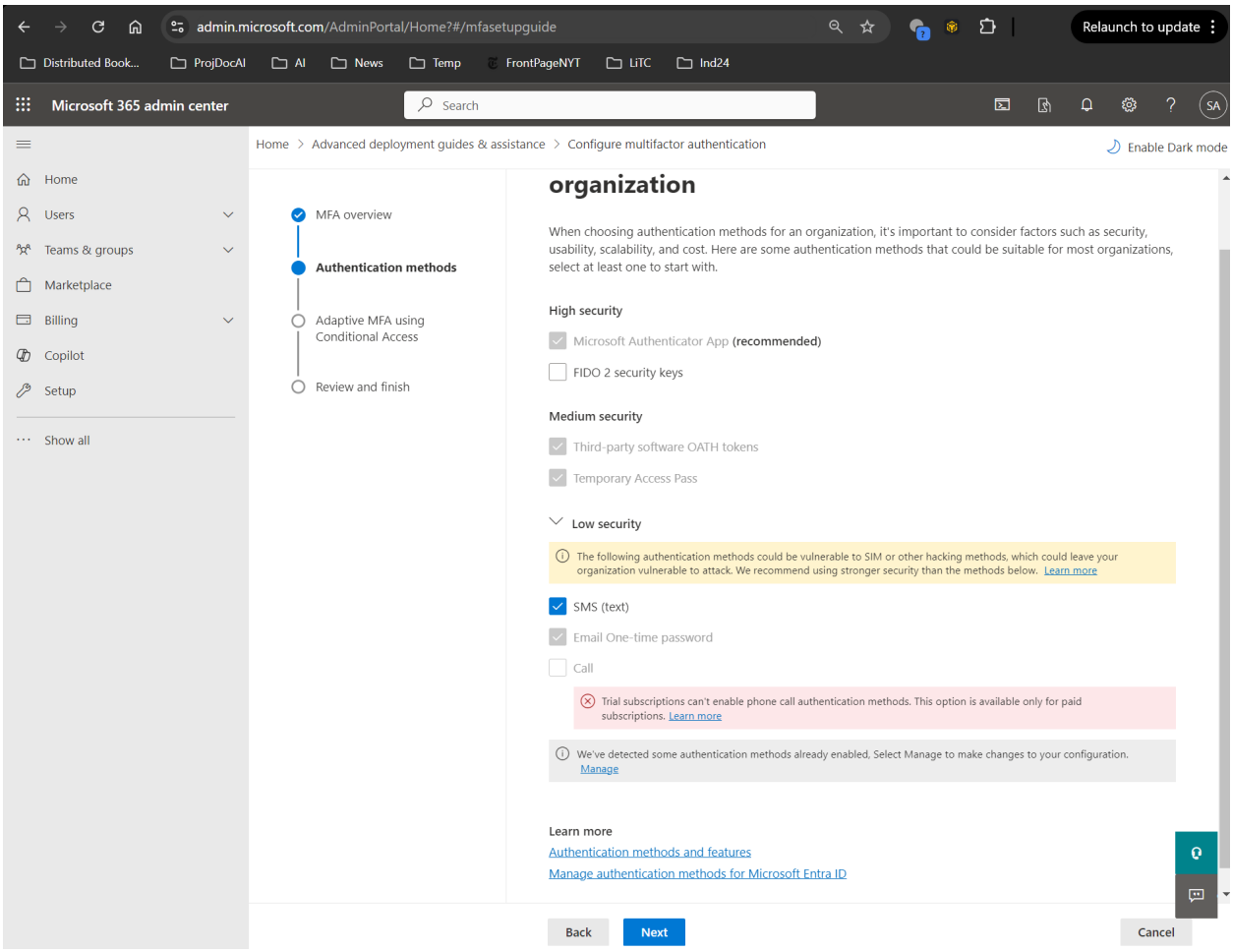
   a. autoInsuranceClaimExtraction-v1

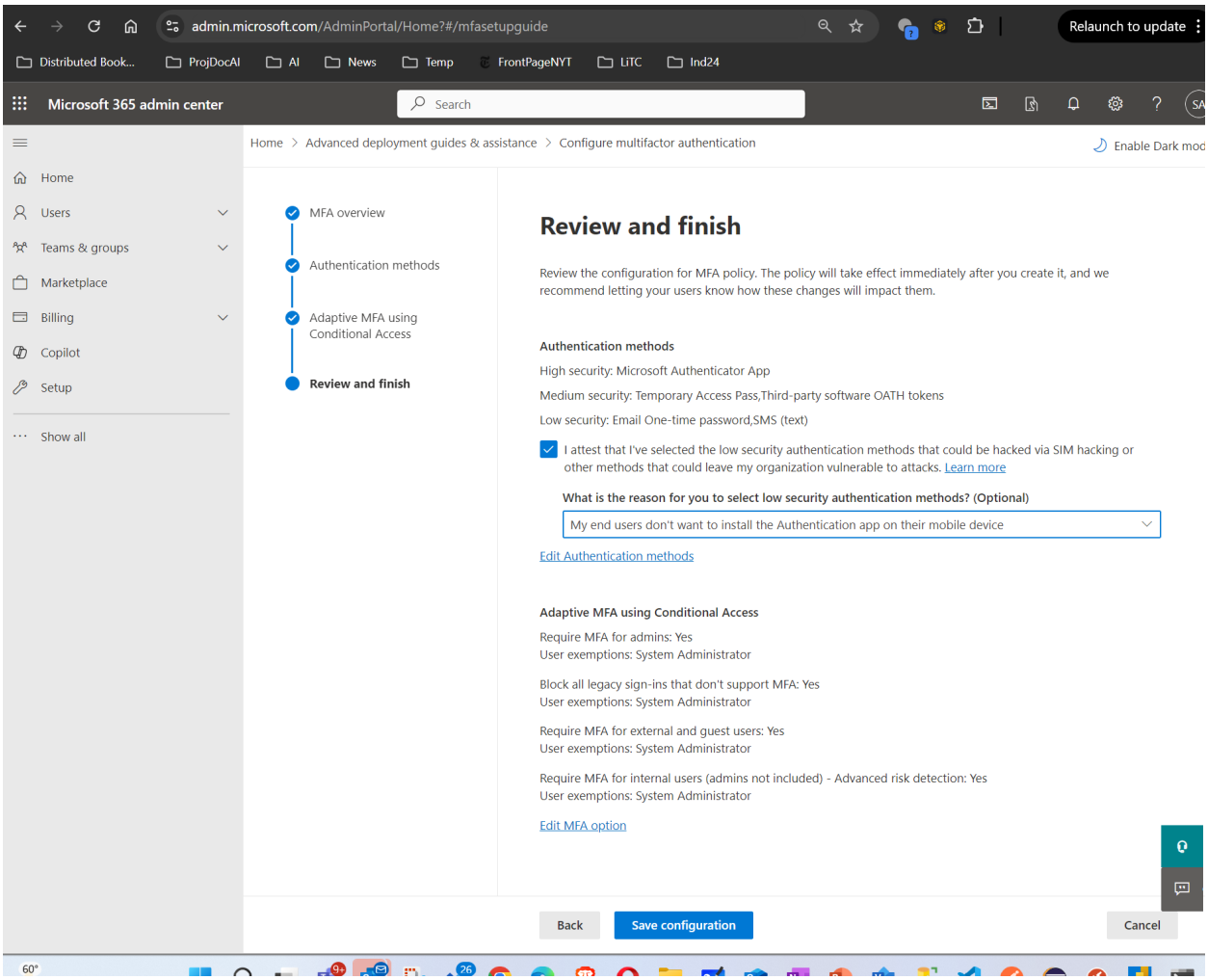
   b. commercialInsuranceApplicationExtraction-v1

   c. workersCompensationApplicationExtraction-v1

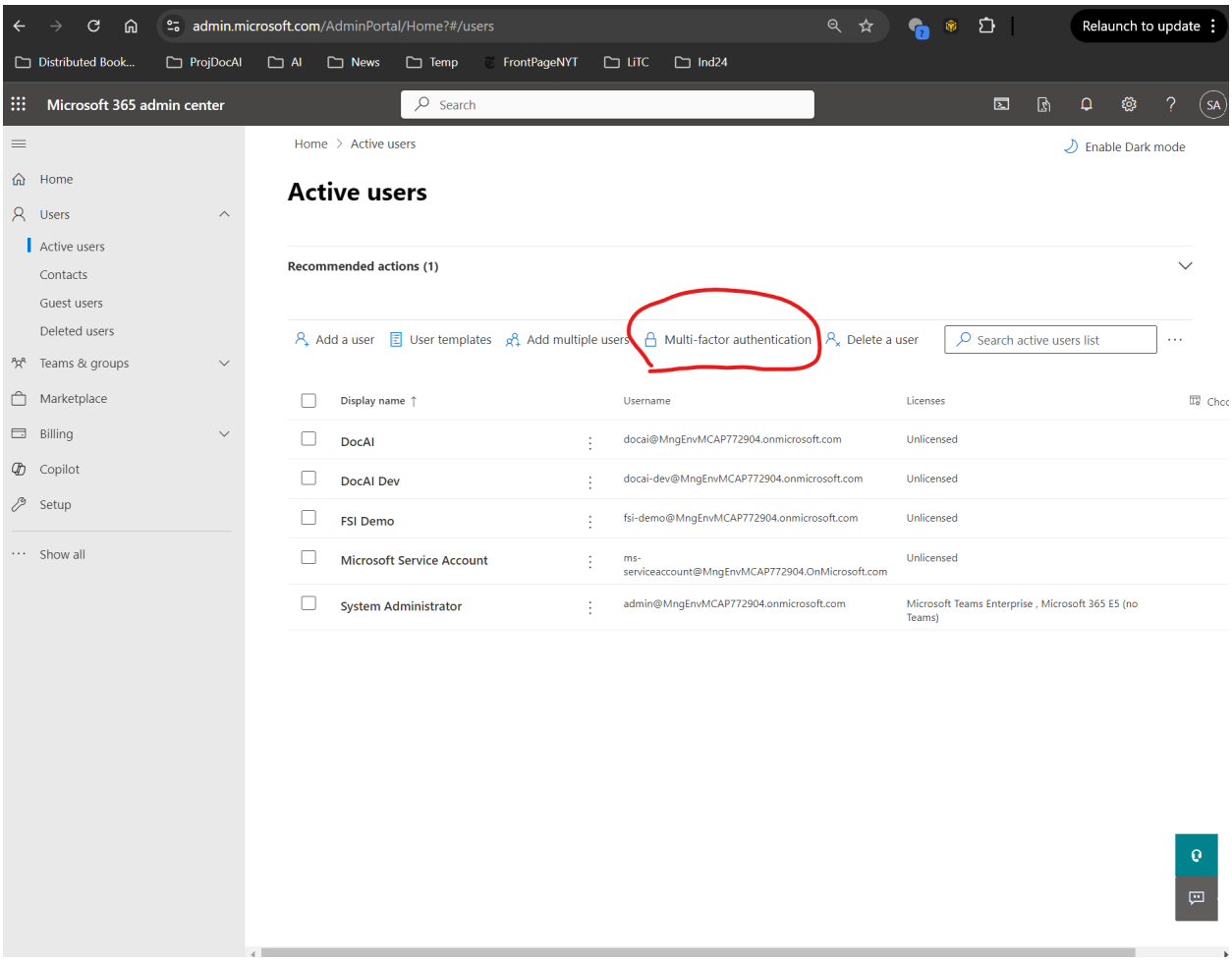
Microsoft Graph API

1. Login as admin in your tenant to the Add User website - https://portal.azure.com/#view/Microsoft_AAD_UsersAndTenants/UserManagementMenuBlade/~/AllUsers
2. Add the users
   - fsi-demo@<your FQDN>
   - docai@<your FQDN>
   - docai-dev@<your FQDN>
3. Remember the passwords
4. Enable MFA for the above users - https://admin.microsoft.com/AdminPortal/Home?#/mfasetupguide

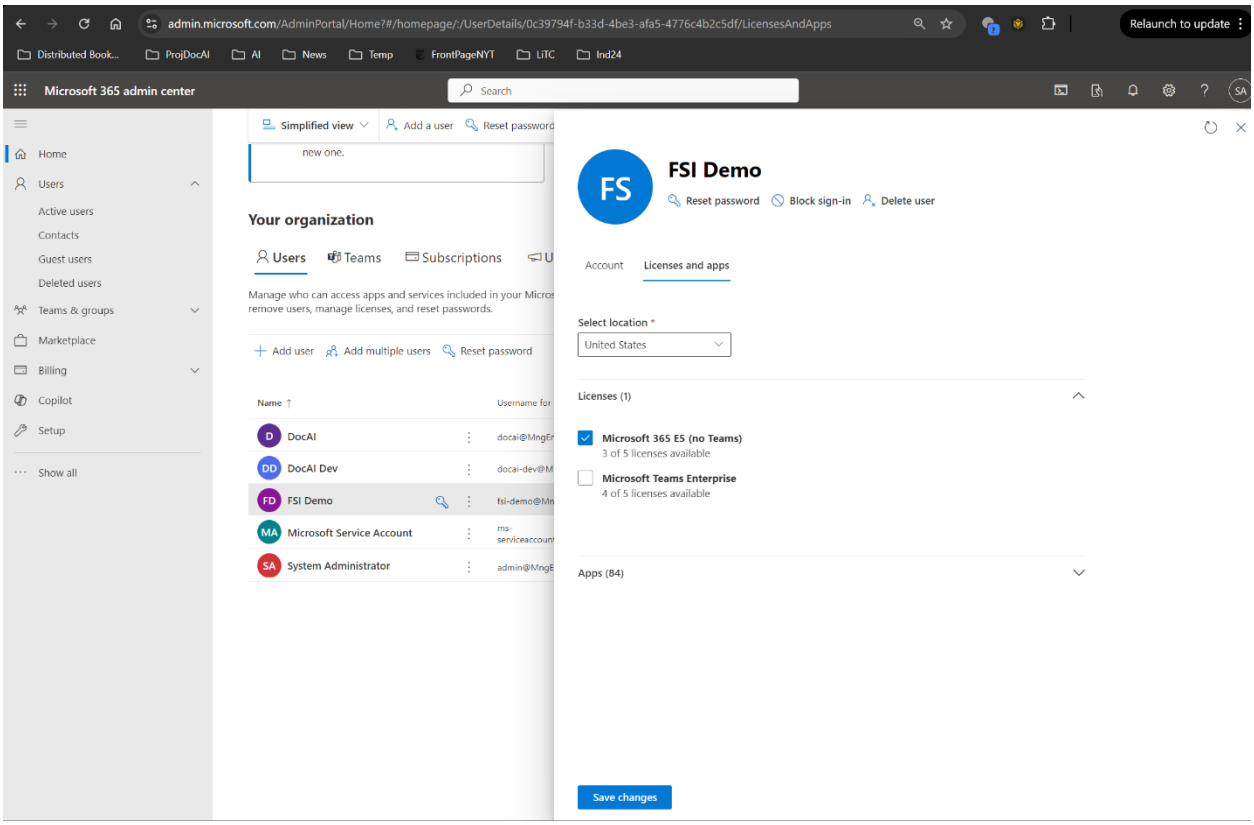

admin.microsoft.com/AdminPortal/Home?#/mfasetupguide
Relaunch to update
Distributed Book...
ProjDocAI
AI
News
Temp
FrontPageNYT
LiTC
Ind24
Microsoft 365 admin center
Search
SA
Home
Users
Teams & groups
Marketplace
Billing
Copilot
Setup
Show all
Home > Advanced deployment guides & assistance > Configure multifactor authentication
Enable Dark mode
MFA overview
Authentication methods
Adaptive MFA using Conditional Access
Review and finish
organization
When choosing authentication methods for an organization, it's important to consider factors such as security, usability, scalability, and cost. Here are some authentication methods that could be suitable for most organizations, select at least one to start with.
High security
Microsoft Authenticator App (recommended)
FIDO 2 security keys
Medium security
Third-party software OATH tokens
Temporary Access Pass
Low security
The following authentication methods could be vulnerable to SIM or other hacking methods, which could leave your organization vulnerable to attack. We recommend using stronger security than the methods below. Learn more
SMS (text)
Email One-time password
Call
Trial subscriptions can't enable phone call authentication methods. This option is available only for paid subscriptions. Learn more
We've detected some authentication methods already enabled, Select Manage to make changes to your configuration. Manage
Learn more
Authentication methods and features
Manage authentication methods for Microsoft Entra ID
Back
Next
Cancel



5. On the same above web page above, on the left side menu items go to Users -> Active Users and select Multi Factor Authentication button

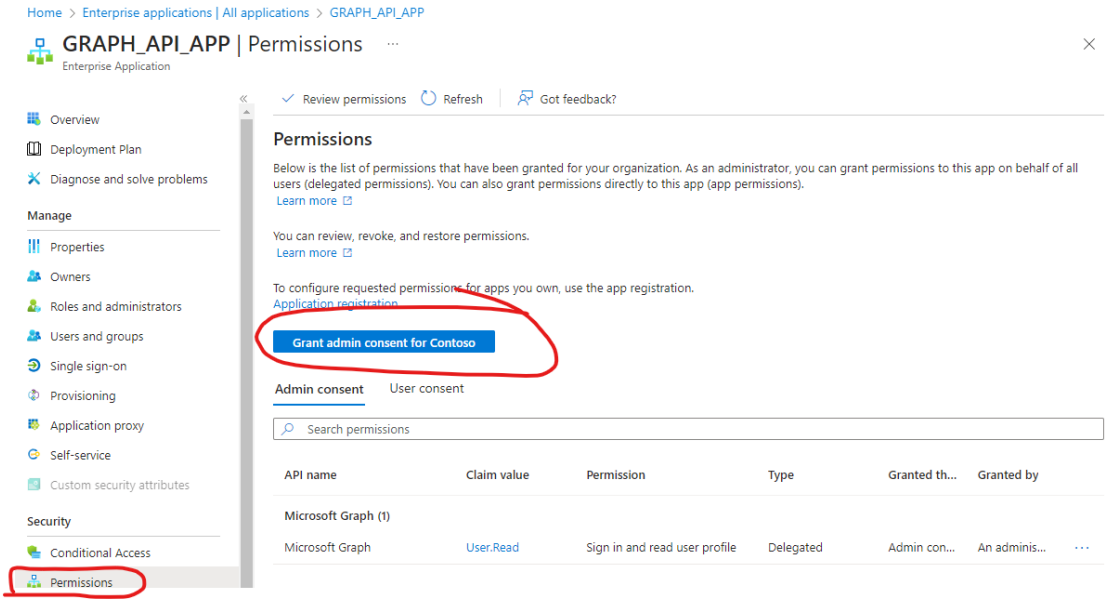


6. Enable M365 license for the fsi-demo user as below:



7. Register your App so the java code in your DocAI Web App can send email using the Graph API. The link is here https://learn.microsoft.com/en-us/graph/auth-register-app-v2

    - Sign in to the MS Entra admin center - Microsoft Entra admin center
    - Go to Identity > Applications > App registrations and select New registration
        I. Name : GRAPH_API_APP
        II. Select Accounts in any organizational directory
    - Hit Register
    - Go to Applications > Enterprise applications
        I. Select the Application GRAPH_API_APP
        II. A new page shows up. This page displays the details of the GRAPH_API_APP
        III. Go to Permissions under Security from the left side menu



IV. Hit Grant admin consent for Contoso

- Go to Applications > App registrations > All applications

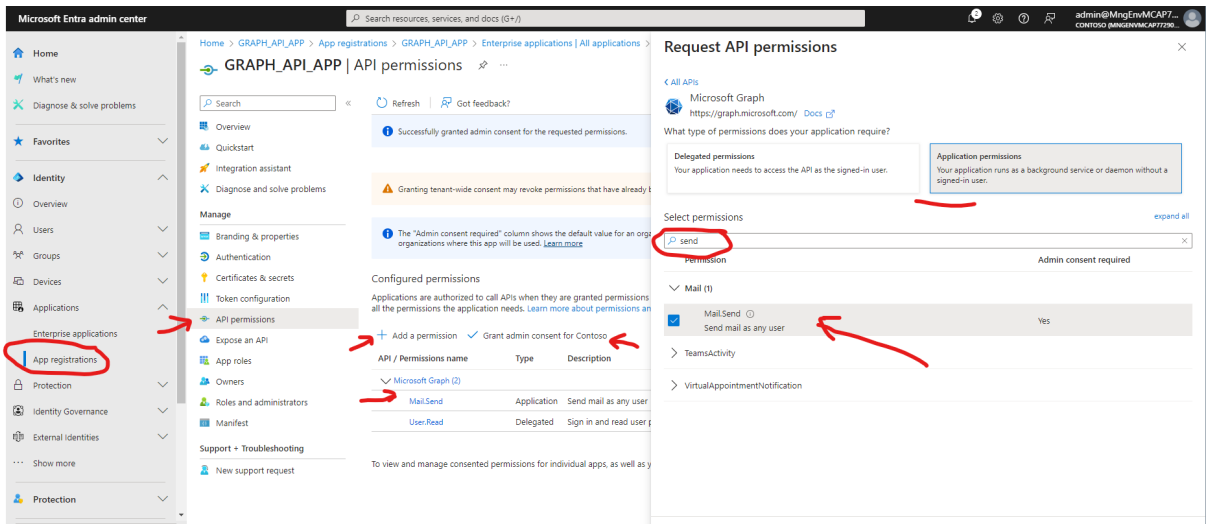
I. Click on GRAPH_API_APP in the list

II. Click on API permissions menu item from the left side of the page

III. Now you should see the Microsoft Graph User.Read in the list. If not, hit refresh on the dialog box in the image above

IV. You need to add the Microsoft Graph Mail.Send permission now, from Add a permission button

V. Finally, you need to click 'Grant admin consent for Contoso' to grant permission for Mail.Send



8. Now from the M365 admin portal (https://admin.microsoft.com/adminportal/home#/homepage) open a Support ticket with the below message, so the fsi-demo user can send emails

    "Release tenant IP not accepting traffic"

9. Under Certificates & secrets, add a Client secret

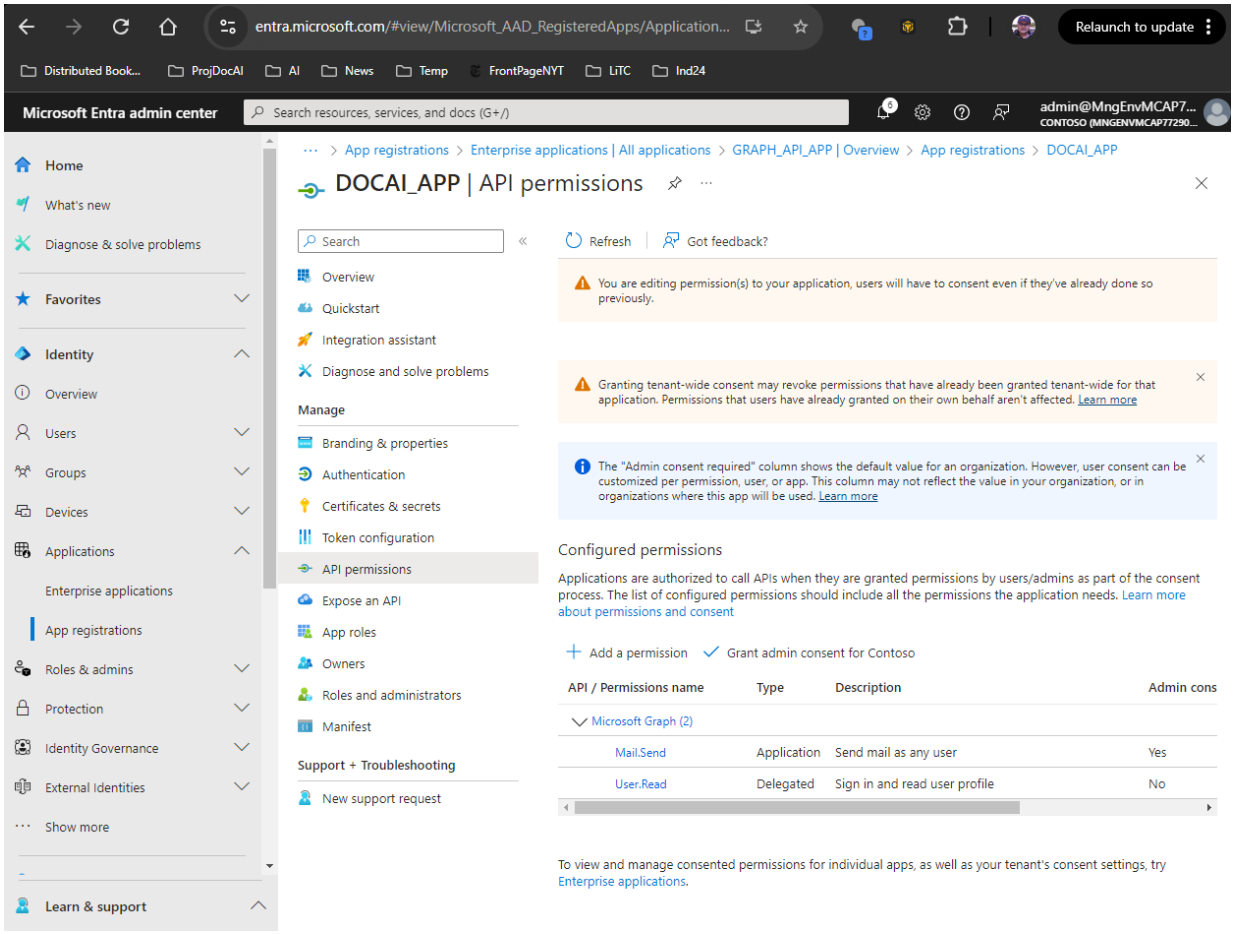
Azure Web App

1. Create the App Registration, so users who login to the web app are authenticated properly. The link is here https://learn.microsoft.com/en-us/graph/auth-register-app-v2

    - Sign in to the MS Entra admin center - Microsoft Entra admin center
    - Go to Identity > Applications > App registrations and select New registration
        I. Name : DOCAI_APP
        II. Select Accounts in any organizational directory
        III. Make sure you add the web url

            (https://<hostname>.azurewebsites.net/login/oauth2/code/)

        IV. Add http://localhost:8080/login/oauth2/code/ (needed when running in your dev machine)
    - Hit Register
    - Now add the Microsoft Graph Mail.Send permission under API Permissions



2. Under Certificates & secrets, add a Client Secret
3. Change all the Environment variables

Azure Bing Search 7 API

Subscription: non-prod

RG: docai-non-prod

Name: bing-search-non-prod

Region: Global

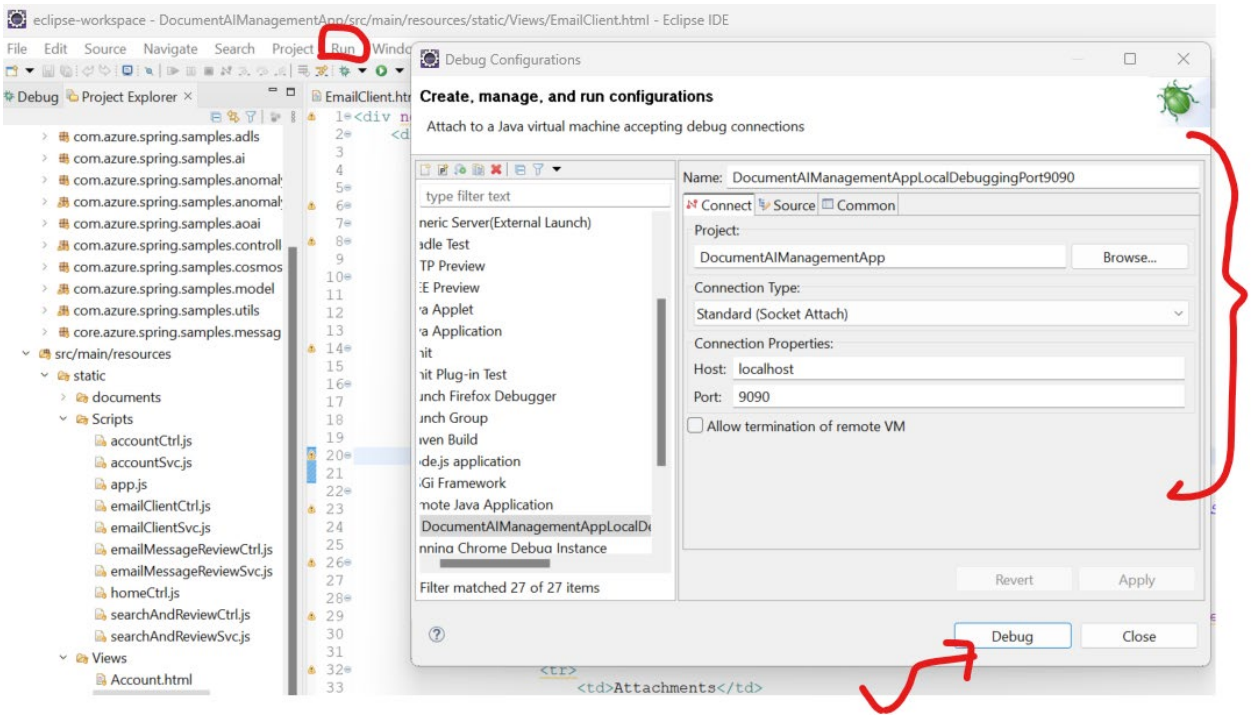
Pricing Tier: Free F1

Deployment

Azure Web App

Go to DocAI\code\web-apps\DocumentAIManagementApp and run:

- env.bat prod
- copy pom.xml.runLocal pom.xml
- mvn package spring-boot:run
- In Eclipse IDE go to Run > Debug Configurations > and hit Debug



Then from the browser go to localhost:8080

If the web app is working, then you are ready to deploy in production

Then run the following to deploy in production

- copy pom.xml.runLocal pom.xml
- mvn install "-DRESOURCEGROUP_NAME=DocAI" "-DWEBAPP_NAME=tr-docai-web" "-DREGION=East US" azure-webapp:deploy
- This should pop up an Azure Login browser window. Login to your Azure portal.
- This will now deploy the web app in production

Once complete, make sure you configure all the environment variables for the Web App correctly

Azure Function App (python)

Start VSCode and open folder DocAI\code\python\function

Make sure the variables in local.settings.json are correct. Example below:

```
{
  "IsEncrypted": false,
  "Values": {
    "DOCUMENT_EXTRACTION_MODEL_CLASS_MAP": "[{'unknown':'unknown'},{'auto-insurance-claim':'autoInsuranceClaimExtraction-v1'},{'commercial-insurance-application':'commercialInsuranceApplicationExtraction-v1'},{'workers-compensation-application':'workersCompensationApplicationExtraction-v1'}]",
    "DOCUMENT_CLASSIFIER_ID": "docai-classifier-v1",
    "DOCUMENT_CONFIDENCE_THRESHOLD": "0.9",
    "CosmosDbConnectionString": "AccountEndpoint=https://<fqdn for cosmosdb>:443/;AccountKey=<key>;",
    "COGNITIVE_SERVICE_ENDPOINT": "https://ai-vision-non-prod.cognitiveservices.azure.com/",
    "COGNITIVE_SERVICE_KEY": "<key>",
    "FUNCTIONS_WORKER_RUNTIME": "python"
  }
}
```

Hit F5 to run the functions locally.

Once it works, deploy the functions in Azure.

Now, make sure all the environment variables are set in Azure Functions instance

Azure Function App (python)

Start VSCode and open folder DocAI\code\csharp\function

Hit f5 to test locally

If it works, then deploy in Azure.

Azure Logic App

1. From portal expand the Workflows menu and click on [@] Parameters and add the following:

```
{
    "SUBJECT_PREFIX_LOGIC_APP_TRIGGER": {
        "type": "String",
        "value": "docai"
    }
}
```

2. Save Parameter
3. Create a workflow named test
4. Select Stateful
5. Add Office 365 when new email arrives as the trigger and set the connection to
    - docai@...
6. Add an action to setup the connection to the Bob Store from above. Use Managed Identity
7. Create a workflow named *docAIEmailProcessingLogicApp*
8. Select Stateful
9. Open it under workflow -> *docAIEmailProcessingLogicApp -> Code ->*
10. Copy and paste the code from the github repo under <repo home>\code\logic-apps\docAIEmailProcessingLogicApp\workflow.json
11. Replace in the workflow.json the name of the placeholder blob store with *docai8adls8prod*
12. Create the connections for Office365, Blob and Functions and fix the workflow in designer
13. Check all the functions and make sure the urls for blob store have the right hostname
14. Make sure the trigger condition for New Email module matches what is in the Web APP application.properties (or the respective environment variable)

Setup Managed Identity

1. Azure Open AI Services (aoai-non-prod):

    Add **Cognitive Services OpenAI Contributor role** for the Management Identity for

    - Azure Function App (docai-python-func-prod)

2. Azure Open AI Services (aoai-vision-non-prod):

   Add **Cognitive Services OpenAI Contributor role** for the Management Identity for

   - Azure Function App (docai-python-func-prod)

3. Azure Storage (docai8adls8prod):

   Add **Storage Blob Data Contributor role** for the Management Identity for

   - Azure Open AI (aoai-non-prod)
   - Azure Document Intelligence Service (doc-intel-non-prod)
   - Azure Logic App (docai-logic-app-prod)

4. Azure Cosmos DB (docai-cosmosdb-prod):

   - Create the System assigned MI from Settings -> Identity
   - Get the Object id for Azure Function App (docai-python-func-prod) & Azure Web App (docai-web-prod)
   - From command line, using Azure CLI login to the admin account in your Azure tenant

     ```
     az login

     az account set --subscription <subscription name for the Azure Cosmos DB>
     ```

   - For each of the Object ids from above run the following from command line

     ```
     az cosmosdb sql role assignment create --account-name "docai-cosmosdb-prod" --resource-group "docai-prod" --scope "/" --principal-id <Object id from above e.g. xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx> --subscription "prod" --role-definition-id "00000000-0000-0000-0000-000000000002"
     ```

     This will enable the code in Azure Functions and Azure Web App to perform read/write in Cosmos DB

   - For the Web App:
     Now you can set the COSMOSDB_API_KEY to " " (or any whitespace) from Web App environment variables list. Note: The COSMOSDB_API_KEY variable must still exist for the App to start and inject it, but you do not need to add any value to it.

     This way if you need to switch back and forth between MI and Secret key (like running the web app from your laptop it is easy to use the key instead of MI) you can just do it by configuring the environment variable and do not need to make code changes.

   - For the Function App:
     Set the following environment variables:
     ```
     CosmosDbConnectionString__accountEndpoint=<cosmosdb endpoint>
     CosmosDbConnectionString__credential=managedidentity
     ```