Adobe Connected System for **Appian**
V2.0.0

Appian Corporation

Version 2.0.0

Table of Contents

Overview

In the modern day, most businesses store the overwhelming majority of their documents digitally for a wide variety of different purposes, from invoices to tax forms. However, digital documentation storage provides many great advantages, they can be difficult to manage, update, and access. Adobe has been a key solution for mitigating these issues with their widely recognized products, and now with the Adobe Connected System, Appian users can utilize Adobe's modern, cloud-based PDF capabilities.

The key utility provided by using the Adobe PDF Services API is document generation**.** This powerful feature allows users to quickly and easily generate and manage business critical documents. For example, a salesman could create a custom, branded request template for his services that contains static information about his company in addition to dynamic fields that could be filled out by his customers.

Another unique feature users can take advantage of is PDF extraction, a cloud-based web service that uses Adobe's Sensei AI technology to automatically extract content and structural information from PDF documents – native or scanned – and outputs it in a structured JSON format. The service extracts text, complex tables, and figures.

Other practical features include PDF compression, conversion, linearization, and OCR. All these features can be combined to become a critical part of making your workflow more efficient.

For more information, please visit PDF features | Adobe Acrobat.

Adobe PDF Services

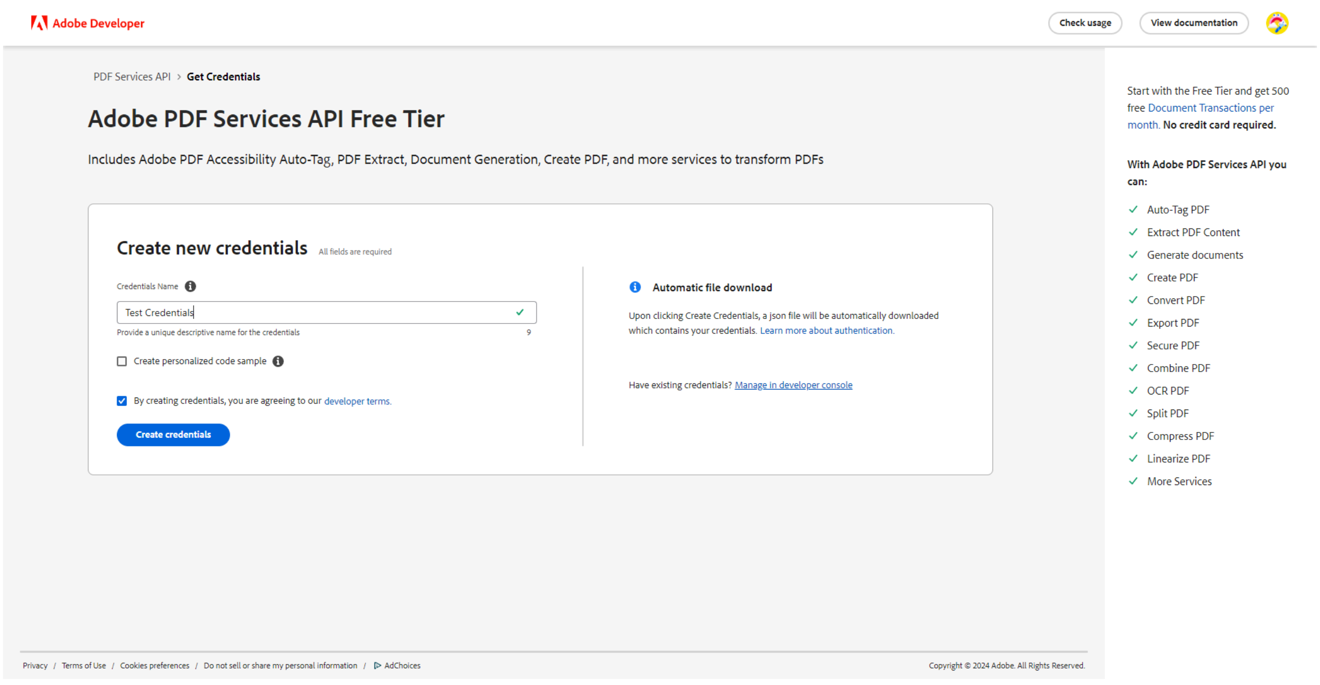
Connected System Configuration

The Adobe Connected System requires the following credentials: Client ID, Client Secret.

- Visit this page to create the credentials for Adobe PDF Services Connected System configuration.

  Important Note: You will be asked to login into your Adobe developer account or to create a new account if you don't have one.

- Provide a name for your credentials and click on Create Credentials and a zip file will be downloaded. The downloaded zip file contains your API credentials. Please store your API credentials securely. You can also access a copy on the dashboard of your Adobe Developer Console.



Adobe PDF Services API Free Tier

Includes Adobe PDF Accessibility Auto-Tag, PDF Extract, Document Generation, Create PDF, and more services to transform PDFs

Create new credentials All fields are required

Credentials Name: Test Credentials

Provide a unique descriptive name for the credentials 9

☐ Create personalized code sample

☑ By creating credentials, you are agreeing to our developer terms.

Automatic file download

Upon clicking Create Credentials, a json file will be automatically downloaded which contains your credentials. Learn more about authentication.

Have existing credentials? Manage in developer console

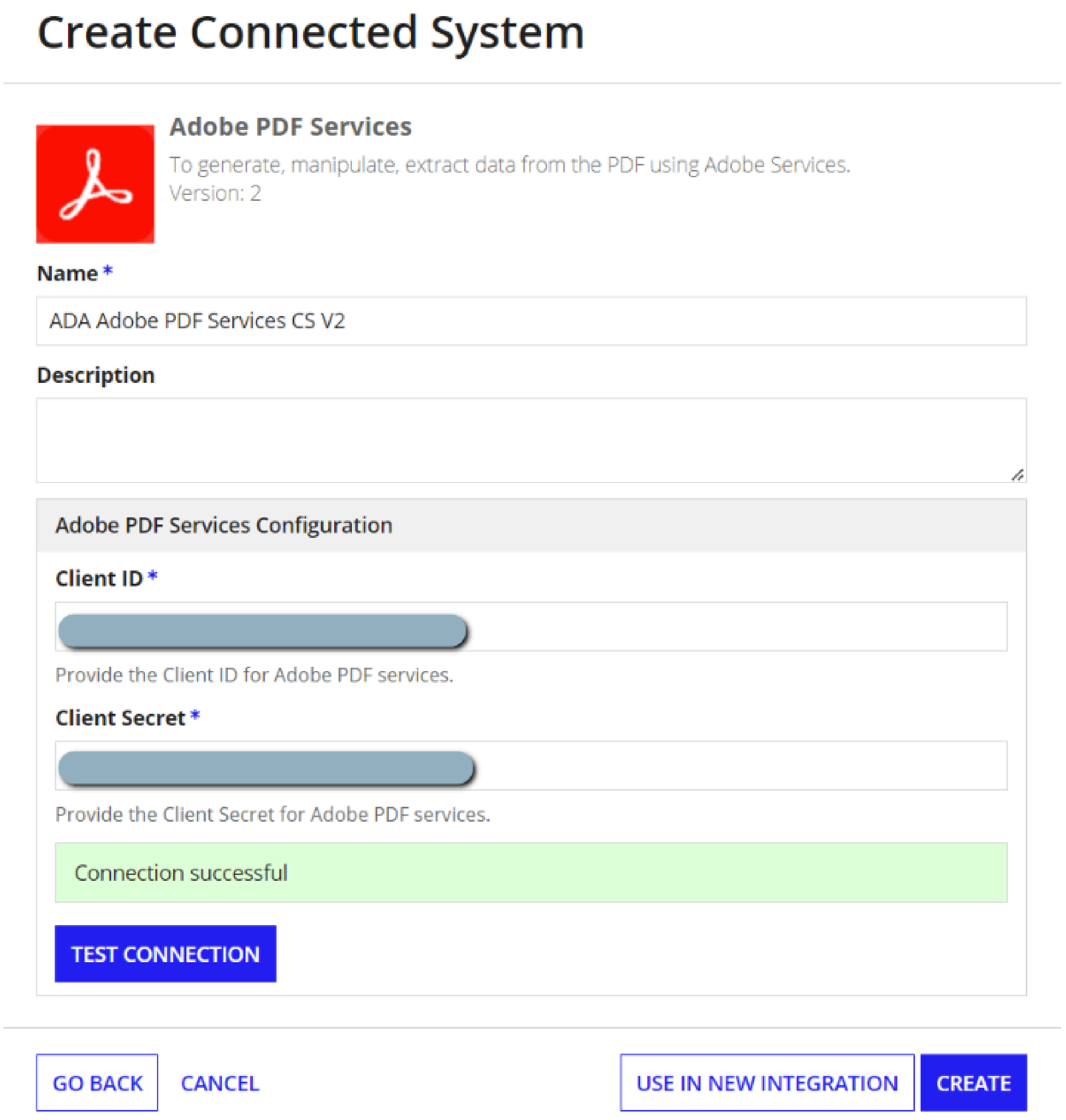
Start with the Free Tier and get 500 free Document Transactions per month. No credit card required.

With Adobe PDF Services API you can:

- ✓ Auto-Tag PDF
- ✓ Extract PDF Content
- ✓ Generate documents
- ✓ Create PDF
- ✓ Convert PDF
- ✓ Export PDF
- ✓ Secure PDF
- ✓ Combine PDF
- ✓ OCR PDF
- ✓ Split PDF
- ✓ Compress PDF
- ✓ Linearize PDF
- ✓ More Services

Privacy / Terms of Use / Cookies preferences / Do not sell or share my personal information / AdChoices
Copyright © 2024 Adobe. All Rights Reserved.

- Extract the zip file and it will contain a json file: ***pdfservices-api-credentials.json*** with the necessary credentials. Copy the ***client_id*** and ***client_secret*** from the json file and paste them in the connected system configuration.


Create Connected System

Adobe PDF Services

To generate, manipulate, extract data from the PDF using Adobe Services.
Version: 2

Name *

ADA Adobe PDF Services CS V2

Description

Adobe PDF Services Configuration

Client ID *

[redacted]

Provide the Client ID for Adobe PDF services.

Client Secret *

[redacted]

Provide the Client Secret for Adobe PDF services.

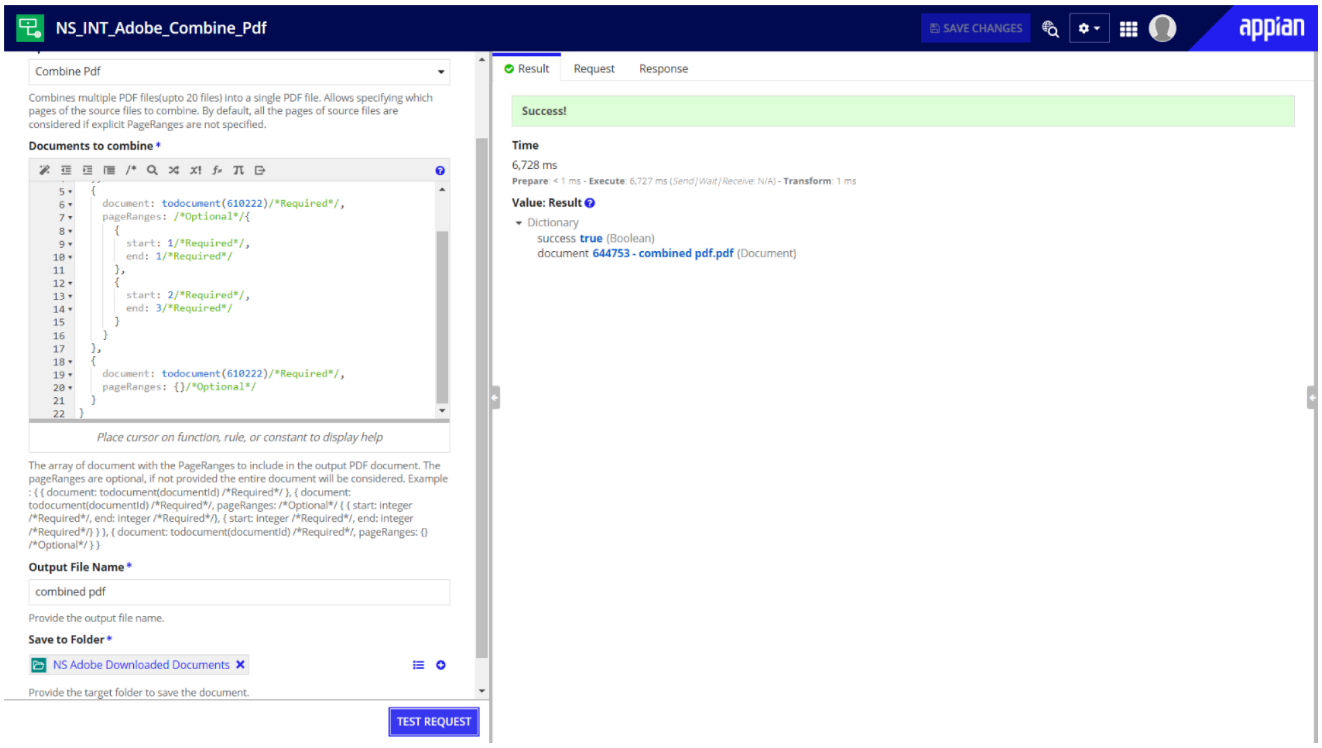
Connection successful

- Click on Test Connection to verify the entered credentials are correct.

Please find pricing information here for the Adobe PDF services API.

Combine PDF

Combines multiple PDF files (up to 20 files) into a single PDF file by specifying which pages of the source files to combine. By default, all the pages of source files are considered if explicit PageRanges are not specified.



Combine Pdf

Combines multiple PDF files(upto 20 files) into a single PDF file. Allows specifying which pages of the source files to combine. By default, all the pages of source files are considered if explicit PageRanges are not specified.

Documents to combine *

```
  {
    document: todocument(610222)/*Required*/,
    pageRanges: /*Optional*/{
      {
        start: 1/*Required*/,
        end: 1/*Required*/
      },
      {
        start: 2/*Required*/,
        end: 3/*Required*/
      }
    }
  },
  {
    document: todocument(610222)/*Required*/,
    pageRanges: {}/*Optional*/
  }
}
```

Place cursor on function, rule, or constant to display help

The array of document with the PageRanges to include in the output PDF document. The pageRanges are optional, if not provided the entire document will be considered. Example : { { document: todocument(documentId) /*Required*/ }, { document: todocument(documentId) /*Required*/, pageRanges: /*Optional*/ { { start: integer /*Required*/, end: integer /*Required*/}, { start: integer /*Required*/, end: integer /*Required*/} } }, { document: todocument(documentId) /*Required*/, pageRanges: {} /*Optional*/ } }

Output File Name *

combined pdf

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time

6,728 ms

Prepare: < 1 ms - **Execute**: 6,727 ms (*Send / Wait / Receive*: N/A) - **Transform**: 1 ms

Value: Result

Dictionary
    success **true** (Boolean)
    document **644753 - combined pdf.pdf** (Document)

Inputs:

<u>**Documents to combine**</u> (List of Dictionary) - Required

Example:

```
Unset
  {
    {
       document: todocument(documentId)/*Required*/
    },
    {
       document: todocument(documentId)/*Required*/,
       pageRanges:{  /*Optional*/
          {
             start: integer /*Required*/,
                end: integer /*Required*/,
            },
            {
                start: integer /*Required*/,
                end: integer /*Required*/,
            }
        }
},
    {
        document: todocument(documentId)/*Required*/,
        pageRanges: {}/*Optional*/
    }
}
```

Description: Each dictionary requires a document. The pageRanges is optional; if provided the start and end inside pageRanges are required. Provide the value for the document using the todocument() function. In pageRanges to pass only a single page, provide the start and end value as the same (ex. start: 3, end: 3. Page numbers are indexed from 1 to N.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

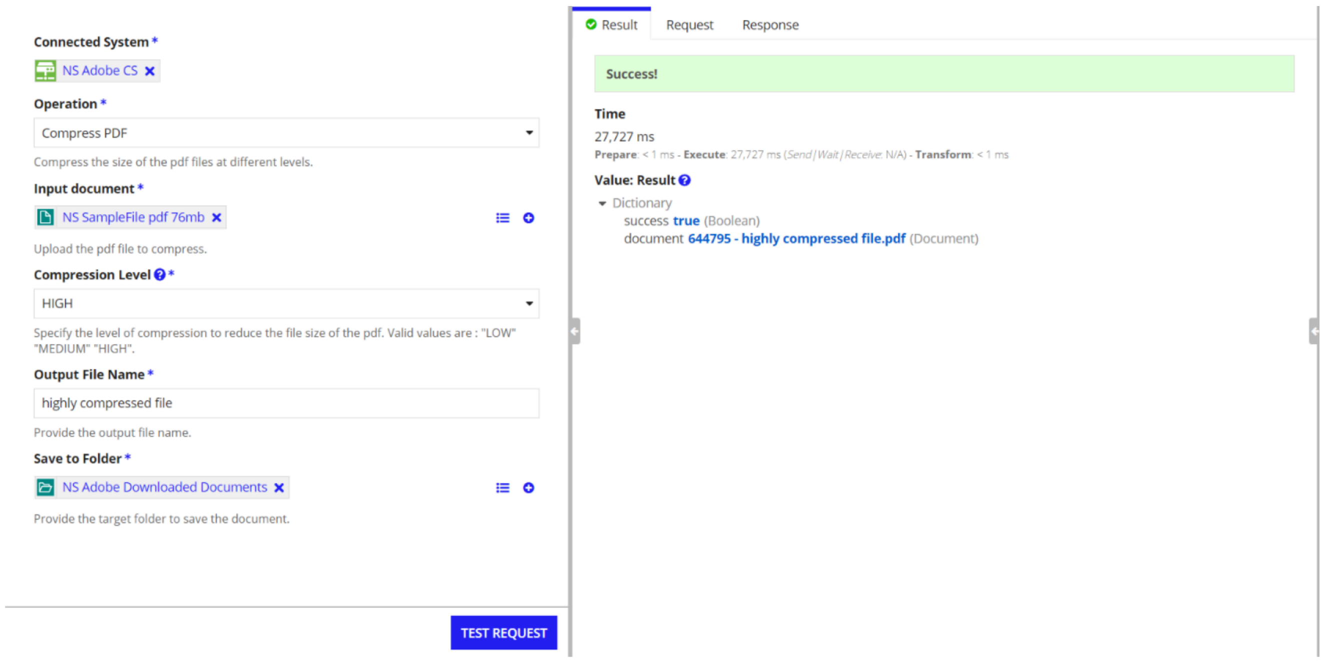
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Compress PDF

Reduces the size of PDF files by compressing to smaller sizes for lower bandwidth viewing, downloading, and sharing. This integration supports multiple compression levels to retain the quality of images and graphics.


Connected System *
NS Adobe CS
Operation *
Compress PDF
Compress the size of the pdf files at different levels.
Input document *
NS SampleFile pdf 76mb
Upload the pdf file to compress.
Compression Level *
HIGH
Specify the level of compression to reduce the file size of the pdf. Valid values are : "LOW" "MEDIUM" "HIGH".
Output File Name *
highly compressed file
Provide the output file name.
Save to Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

Success!
Time
27,727 ms
Prepare: < 1 ms - Execute: 27,727 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms
Value: Result
Dictionary
success true (Boolean)
document 644795 - highly compressed file.pdf (Document)

Inputs:

<u>Input document</u> (Document) - Required

Provide the input document.

<u>Compression Level</u> (Text) - Required
Specify the level of compression to apply to the PDF.

Valid values: LOW, MEDIUM, HIGH.

LOW: Reduces resolution of the coloured and grayscale images above 250 dpi to 200 dpi. This option uses JP2K high quality compression.

MEDIUM: Reduces resolution of the coloured and grayscale images above 200 dpi to 144 dpi. This option uses JP2K medium quality compression.

HIGH: Reduces resolution of the coloured and grayscale images above 100 dpi to 72 dpi. This option uses JPEG medium quality compression. Output PDF will not contain hidden layers, document structure, metadata, javascript, user properties and print settings.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

Description: Provide the target save to the folder where the output file needs to be saved.

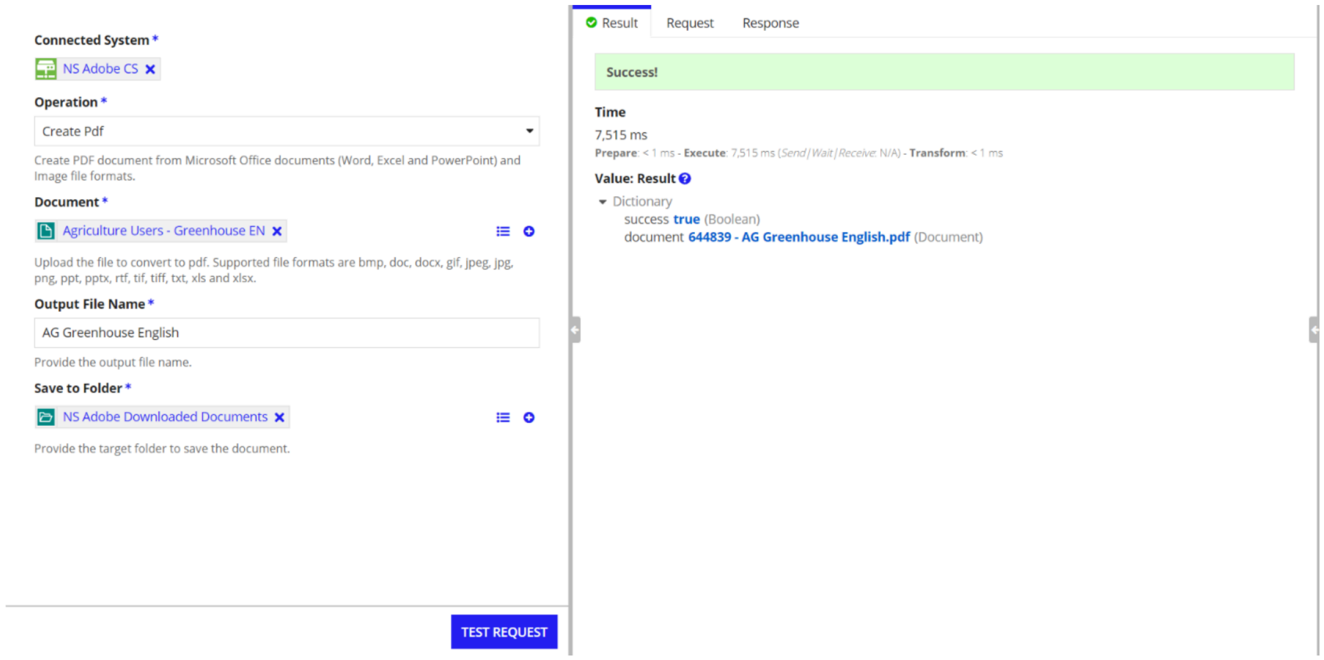
Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Create PDF

Creates PDFs from a variety of formats such as Microsoft Word, PowerPoint, and Excel; as well as text, image, Zip, and URL.

Support for HTML to PDF, DOC to PDF, DOCX to PDF, PPT to PDF, PPTX to PDF, XLS to PDF, XLSX to PDF, TXT to PDF, RTF to PDF, BMP to PDF, JPEG to PDF, GIF to PDF, TIFF to PDF, PNG to PDF.



Connected System * NS Adobe CS

Operation * Create Pdf

Create PDF document from Microsoft Office documents (Word, Excel and PowerPoint) and Image file formats.

Document * Agriculture Users - Greenhouse EN

Upload the file to convert to pdf. Supported file formats are bmp, doc, docx, gif, jpeg, jpg, png, ppt, pptx, rtf, tif, tiff, txt, xls and xlsx.

Output File Name * AG Greenhouse English

Provide the output file name.

Save to Folder * NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time
7,515 ms
Prepare: < 1 ms - Execute: 7,515 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value: Result
Dictionary
success true (Boolean)
document 644839 - AG Greenhouse English.pdf (Document)

Inputs:

<u>Input document</u> (Document) - Required

Provide the input document.

<u>Output File Name</u> (Text) - Required

Description: Provide the name of the output file.

<u>Save to Folder</u> (Folder) - Required

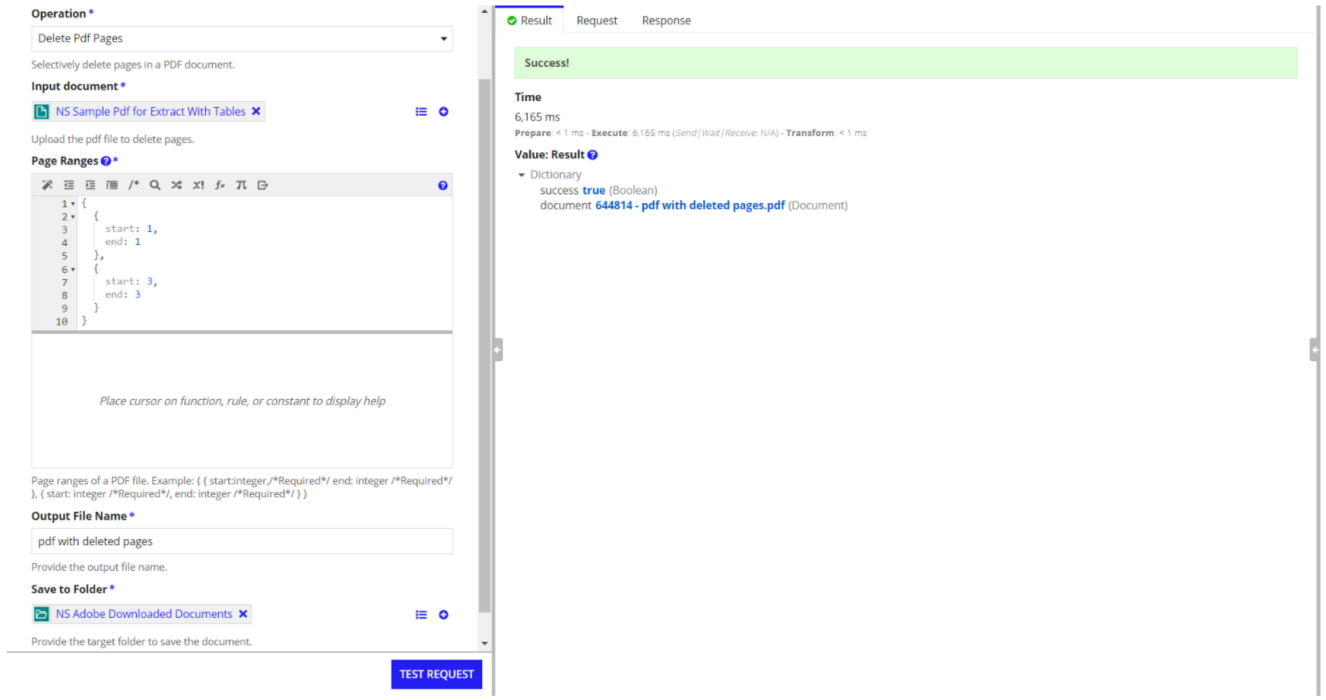
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Delete PDF Pages

Deletes one or more pages from a document. The delete pages operation selectively removes pages from a PDF file.



Operation *

Delete Pdf Pages

Selectively delete pages in a PDF document.

Input document *

NS Sample Pdf for Extract With Tables

Upload the pdf file to delete pages.

Page Ranges *

```
{
  {
    start: 1,
    end: 1
  },
  {
    start: 3,
    end: 3
  }
}
```

Place cursor on function, rule, or constant to display help

Page ranges of a PDF file. Example: { { start:integer,/*Required*/ end: integer /*Required*/ }, { start: integer /*Required*/, end: integer /*Required*/ } }

Output File Name *

pdf with deleted pages

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time

6,165 ms

Prepare: < 1 ms - **Execute**: 6,165 ms (*Send | Wait | Receive*: N/A) - **Transform**: < 1 ms

Value: Result

Dictionary
    success **true** (Boolean)
    document **644814 - pdf with deleted pages.pdf** (Document)

Inputs:

<u>Input document </u>(Document) - Required

Provide the input document.

<u>Page Ranges:</u> (List of Dictionary) - Required

Page ranges of the PDF file. Each dictionary confirms the start and end.
Example :

```
Unset
{
{
start: integer /*Required*/,
end: integer /*Required*/
}
}
```

In pageRanges to pass only a single page, provide the start and end value as the same.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

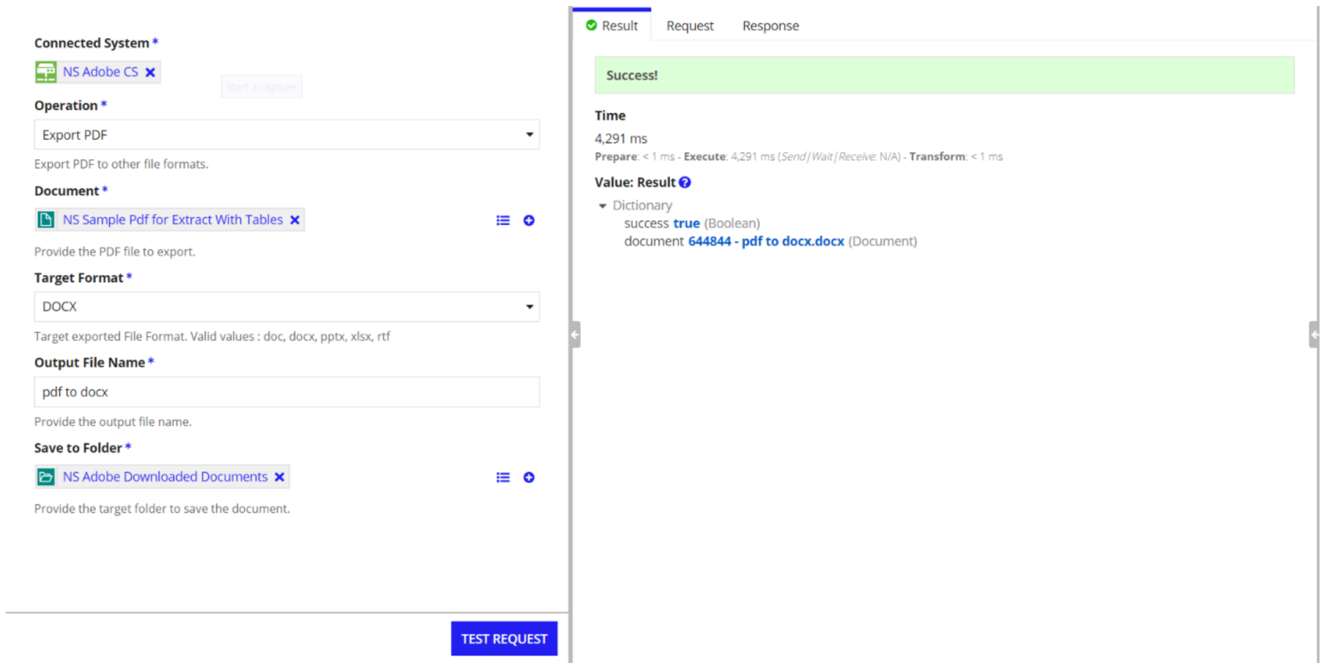
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Export PDF

Exports a source PDF file into a doc, docx, pptx, rtf, or xlsx file.



Connected System *
NS Adobe CS
Operation *
Export PDF
Export PDF to other file formats.
Document *
NS Sample Pdf for Extract With Tables
Provide the PDF file to export.
Target Format *
DOCX
Target exported File Format. Valid values : doc, docx, pptx, xlsx, rtf
Output File Name *
pdf to docx
Provide the output file name.
Save to Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

Result | Request | Response

Success!

Time
4,291 ms
Prepare: < 1 ms - Execute: 4,291 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value: Result
Dictionary
success true (Boolean)
document 644844 - pdf to docx.docx (Document)

Inputs:

Input document (Document) - Required

Provide the input document.

Target Format (Text) - Required

Specifies the output file format. Valid values: doc, docx, pptx, xlsx, rtf.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

Description: Provide the target save to the folder where the output file needs to be saved.

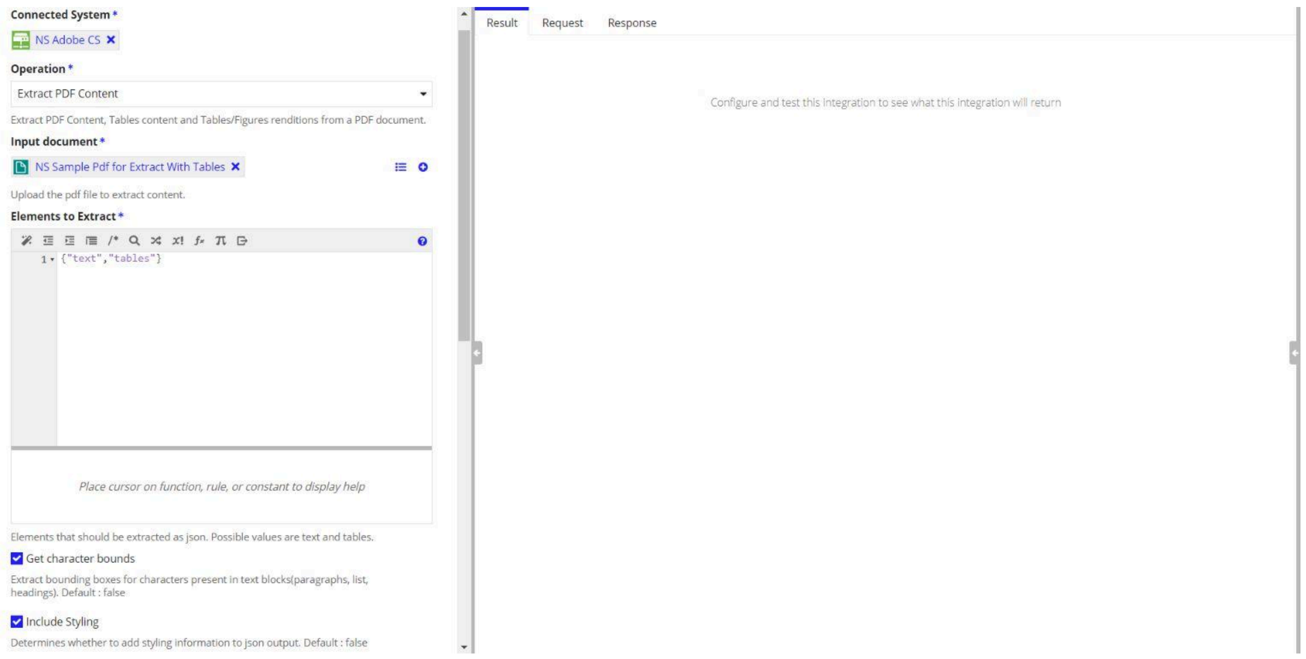
Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

12

Extract PDF Content

Extracts PDF Content, tables content, character bounds, styling info and tables or figures from a PDF document.



Connected System *
NS Adobe CS
Operation *
Extract PDF Content
Extract PDF Content, Tables content and Tables/Figures renditions from a PDF document.
Input document *
NS Sample Pdf for Extract With Tables
Upload the pdf file to extract content.
Elements to Extract *

```
{"text","tables"}
```

Place cursor on function, rule, or constant to display help
Elements that should be extracted as json. Possible values are text and tables.
☑ Get character bounds
Extract bounding boxes for characters present in text blocks(paragraphs, list, headings). Default : false
☑ Include Styling
Determines whether to add styling information to json output. Default : false

Result
Request
Response
Configure and test this Integration to see what this integration will return

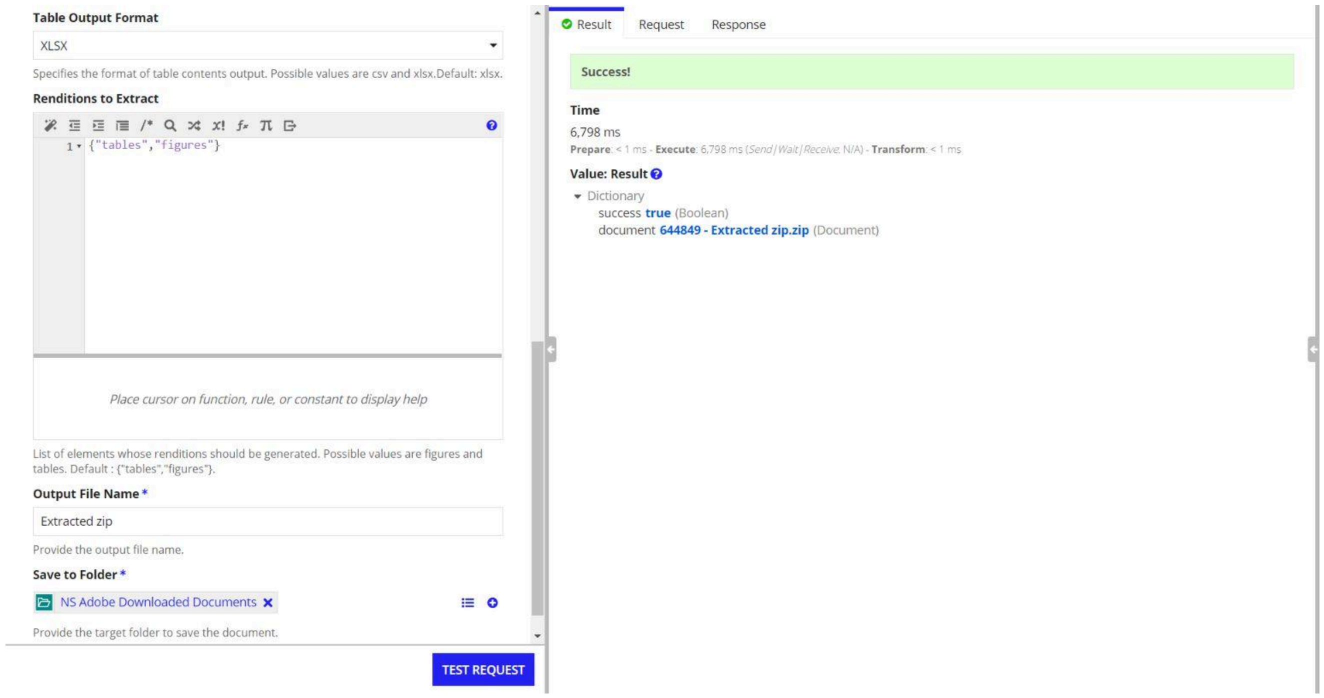


Table Output Format

XLSX

Specifies the format of table contents output. Possible values are csv and xlsx.Default: xlsx.

Renditions to Extract

```
{"tables","figures"}
```

Place cursor on function, rule, or constant to display help

List of elements whose renditions should be generated. Possible values are figures and tables. Default : {"tables","figures"}.

Output File Name *

Extracted zip

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time

6,798 ms

Prepare: < 1 ms - Execute: 6,798 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value: Result

```
Dictionary
    success true (Boolean)
    document 644849 - Extracted zip.zip (Document)
```

Inputs:

Input document (Document) - Required

Description: Provide the input document.

Elements to Extract (List of Text) - Required

Description: Provide the list of elements to extract from the PDF document. Possible values are text and tables.

Example:

```
Unset

{"text","tables"}
```

Get character bounds (Boolean) - Optional

Extract bounding boxes for characters present in text blocks(paragraphs, list, headings). Default: false

Include Styling (Boolean) - Optional

Description: Determines whether to to get styling information for each text element(Bold / Italics / Superscript etc) . Default: false

Table Output Format (Text) - Optional

Description: Specifies the format of table contents output. Possible values are csv and xlsx. Default: xlsx.

Renditions to Extract (List of Text) - Optional

Description: Determines whether to get figure renditions as PNGs and table renditions in PNG and XLSX/CSVformat. Possible values are figures and tables.

Example:

```
Unset
{"figures", "tables"}.
```

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

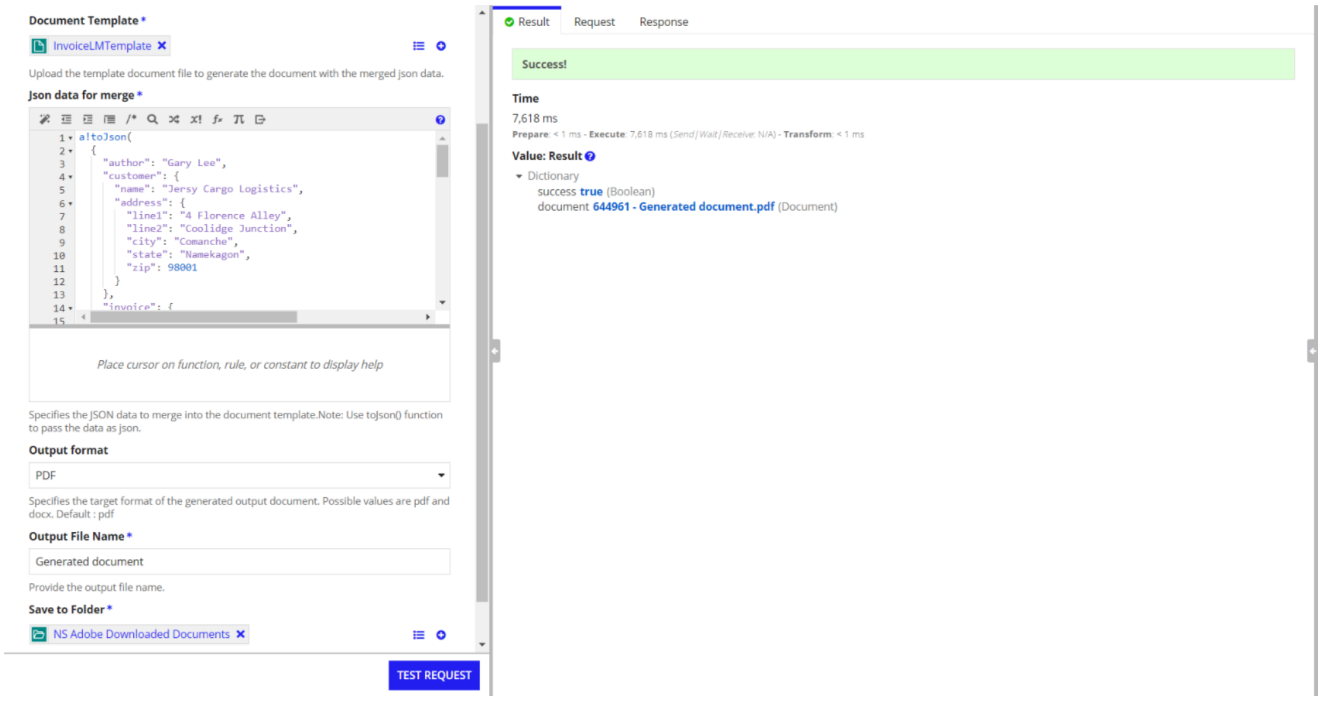
```
Unset
{
success: true,
document: documentId - Output file name
}
```

Please click here for the details on the output.

Generate Document

Merges Word based templates with JSON data to create Word and PDF documents. By using the a!toJson function, user can input JSON data into Word based templates to create dynamic documents.

We can use this to generate documents dynamically from Appian Records using the record data.



Document Template *

InvoiceLMTemplate

Upload the template document file to generate the document with the merged json data.

Json data for merge *

```
a!toJson(
  {
    "author": "Gary Lee",
    "customer": {
      "name": "Jersy Cargo Logistics",
      "address": {
        "line1": "4 Florence Alley",
        "line2": "Coolidge Junction",
        "city": "Comanche",
        "state": "Namekagon",
        "zip": 98001
      }
    },
    "invoice": {
```

Place cursor on function, rule, or constant to display help

Specifies the JSON data to merge into the document template.Note: Use toJson() function to pass the data as json.

Output format

PDF

Specifies the target format of the generated output document. Possible values are pdf and docx. Default : pdf

Output File Name *

Generated document

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Success!

Time

7,618 ms

Prepare: < 1 ms - **Execute**: 7,618 ms (*Send | Wait | Receive*: N/A) - **Transform**: < 1 ms

Value: Result

Dictionary
success **true** (Boolean)
document **644961 - Generated document.pdf** (Document)

Inputs:

Document template: (Document) - Required

Description: Provide the template document file with the template tags.

Json data for merge (JSON) - Required

Description: Provide the corresponding JSON data to merge in the template document. Note: Please use toJson() function to provide the json input as shown in the screenshot below.

Please visit this page to learn more about template tags and the json data to merge and how to use them.

Output format (Text) - Optional

Description: Specifies the output format of the generated document. Possible values are PDF and docx. Default: PDF

<u>Output File Name</u> (Text) - Required

Description: Provide the name of the output file.

<u>Save to Folder</u> (Folder) - Required

Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

HTML To PDF

Converts a Html file with internal styles or URL to a PDF.

Inputs:

Input Type (Text) - Required.

Description: Select the type of input html to convert. Valid values: fileUpload, inputUrl.
Input document (Document) - Required
Description: Provide the input html document.

URL (Text) - Required

Description: Provide the URL that needs to be converted to PDF.

Include header and footer: (Boolean) - Optional

Description: Determine whether to add default headers and footers to the output pages. The default header includes a short date and the contents of the document title. The default footer includes a file name and a page n/m reference. Default: false.

Page Width (Decimal) - Optional

Description: The width (in inches) of the output paper size. Default : 11. 16
Page Height (Decimal) - Optional

Description: The height (in inches) of the output paper size.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

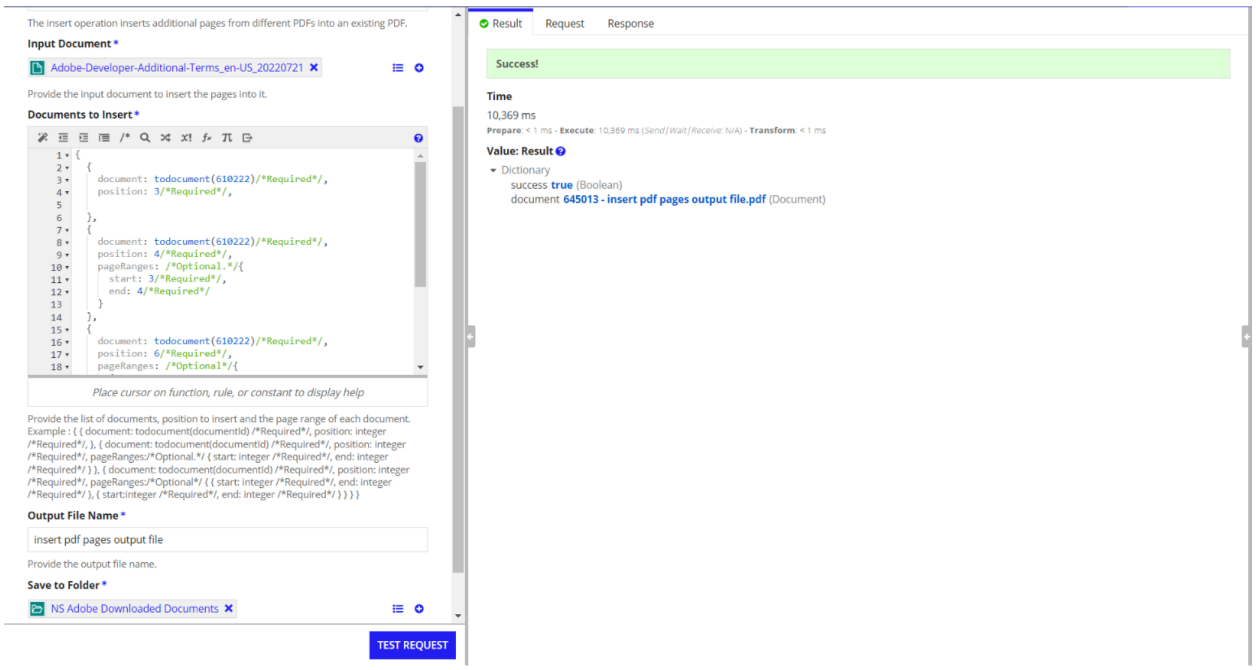
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Insert Pages to PDF

Inserts additional pages from different PDFs into an existing PDF.



The insert operation inserts additional pages from different PDFs into an existing PDF.

Input Document *

Adobe-Developer-Additional-Terms_en-US_20220721

Provide the input document to insert the pages into it.

Documents to Insert *

```
{
  {
    document: todocument(610222)/*Required*/,
    position: 3/*Required*/,

  },
  {
    document: todocument(610222)/*Required*/,
    position: 4/*Required*/,
    pageRanges: /*Optional.*/{
      start: 3/*Required*/,
      end: 4/*Required*/
    }
  },
  {
    document: todocument(610222)/*Required*/,
    position: 6/*Required*/,
    pageRanges: /*Optional*/{
```

Place cursor on function, rule, or constant to display help

Provide the list of documents, position to insert and the page range of each document. Example : { { document: todocument(documentId) /*Required*/, position: integer /*Required*/, }, { document: todocument(documentId) /*Required*/, position: integer /*Required*/, pageRanges:/*Optional.*/ { start: integer /*Required*/, end: integer /*Required*/ } }, { document: todocument(documentId) /*Required*/, position: integer /*Required*/, pageRanges:/*Optional*/ { { start: integer /*Required*/, end: integer /*Required*/ }, { start:integer /*Required*/, end: integer /*Required*/ } } } }

Output File Name *

insert pdf pages output file

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Success!

Time

10,369 ms

Prepare: < 1 ms - **Execute**: 10,369 ms (*Send/Wait/Receive*: N/A) - **Transform**: < 1 ms

Value: Result

Dictionary
success **true** (Boolean)
document **645013 - insert pdf pages output file.pdf** (Document)

Input Document (Document) - Required
Description: Provide the input document to insert the pages into.

Documents to Insert (List of Dictionary) - Required

Example:

```
Unset

{

{

document: todocument(documentId)/*Required*/

position: integer /*Required*/

},

{
```

```
document: todocument(documentId)/*Required*/,

position: integer /*Required*/,

pageRanges: /*Optional*/{

{

start: integer /*Required*/,

end: integer /*Required*/,

},

{

start: integer /*Required*/,

end: integer /*Required*/,

}

}

},

{

document: todocument(documentId)/*Required*/,

position: integer /*Required*/,

pageRanges: {}/*Optional*/

}

}
```

Description: Each dictionary requires a document and a position at where this document needs to be inserted in the input document. The pageRanges is optional, if provided the start and end inside pageRanges are required. Provide the value for the document using the todocument() function. In pageRanges to pass only a single page, provide the start and end value as the same.Page numbers are indexed from 1 to N.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

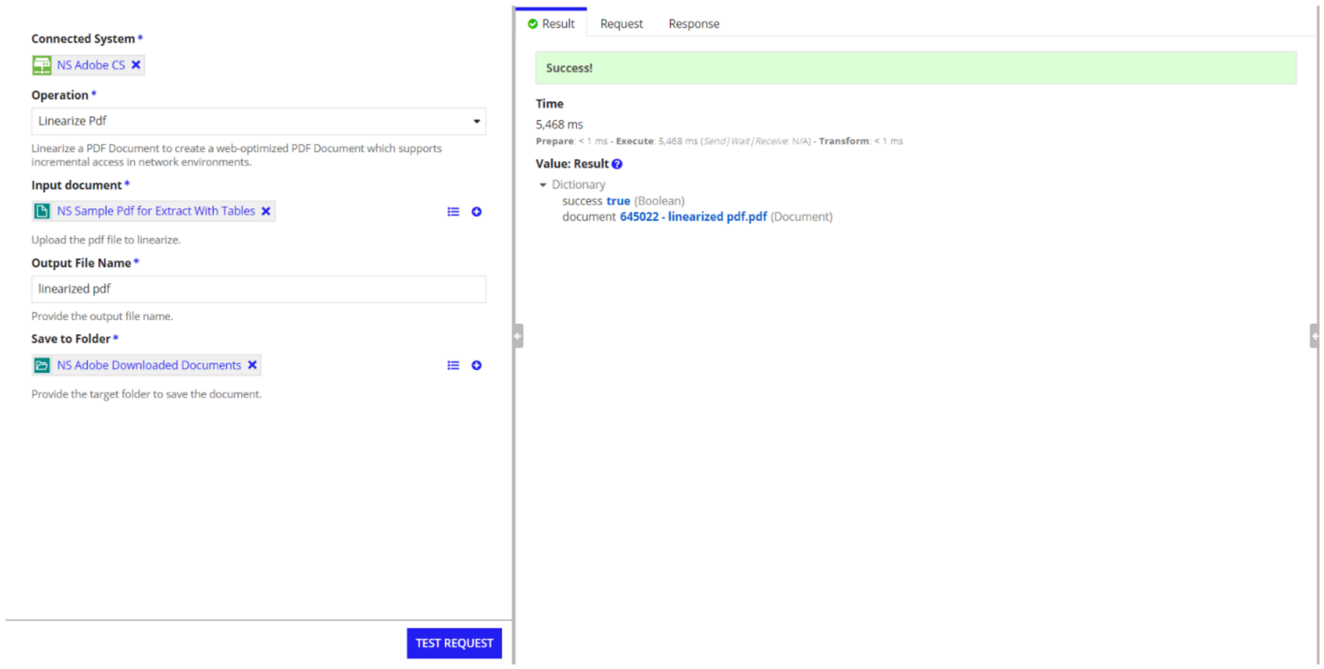
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Linearize PDF

Optimizes PDFs for quick viewing on the web, especially for mobile clients. Linearization allows your end users to view large PDF documents incrementally so that they can view pages much faster in lower bandwidth conditions.



Connected System *
NS Adobe CS

Operation *
Linearize Pdf

Linearize a PDF Document to create a web-optimized PDF Document which supports incremental access in network environments.

Input document *
NS Sample Pdf for Extract With Tables

Upload the pdf file to linearize.

Output File Name *
linearized pdf

Provide the output file name.

Save to Folder *
NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time
5,468 ms
Prepare: < 1 ms - Execute: 5,468 ms (Send | Wait | Receive: N/A) - Transform: < 1 ms

Value: Result
Dictionary
success true (Boolean)
document 645022 - linearized pdf.pdf (Document)

Inputs:
Input document (Document) - Required

Provide the input document to linearize it.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

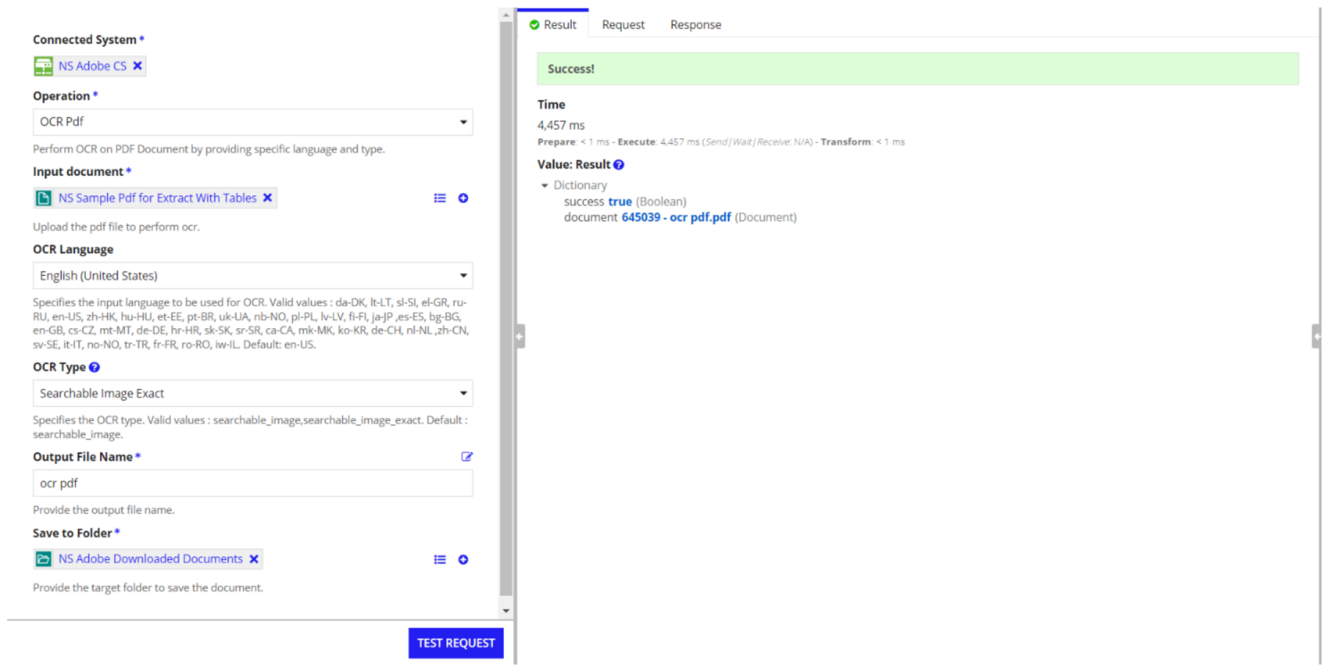
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

OCR PDF

Uses built-in optical character recognition (OCR) to convert images to text and enable fully text searchable documents for archiving and creation of searchable indexes. OCR converts images to text so that you and your users can fully interact with the PDF file. After performing OCR, the PDF may be fully editable and searchable.



Connected System *
NS Adobe CS

Operation *
OCR Pdf
Perform OCR on PDF Document by providing specific language and type.

Input document *
NS Sample Pdf for Extract With Tables
Upload the pdf file to perform ocr.

OCR Language
English (United States)
Specifies the input language to be used for OCR. Valid values : da-DK, lt-LT, sl-SI, el-GR, ru-RU, en-US, zh-HK, hu-HU, et-EE, pt-BR, uk-UA, nb-NO, pl-PL, lv-LV, fi-FI, ja-JP ,es-ES, bg-BG, en-GB, cs-CZ, mt-MT, de-DE, hr-HR, sk-SK, sr-SR, ca-CA, mk-MK, ko-KR, de-CH, nl-NL ,zh-CN, sv-SE, it-IT, no-NO, tr-TR, fr-FR, ro-RO, iw-IL. Default: en-US.

OCR Type
Searchable Image Exact
Specifies the OCR type. Valid values : searchable_image,searchable_image_exact. Default : searchable_image.

Output File Name *
ocr pdf
Provide the output file name.

Save to Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

Success!

Time
4,457 ms
Prepare: < 1 ms - Execute: 4,457 ms (Send | Wait | Receive: N/A) - Transform: < 1 ms

Value: Result
Dictionary
success true (Boolean)
document 645039 - ocr pdf.pdf (Document)

Inputs:

Input document (Document) - Required

Provide the input document.

OCR Language (Text) - Optional

Description: Specifies the input language to be used for OCR. Default: en-US. 20

OCR Type (Text) - Optional

Description: Specifies OCR Type. Valid Values: "searchable_image" "searchable_image_exact". Default: "searchable_image".

Searchable Image: This type ensures that text is searchable and selectable, but modifies the original image during the cleanup process (for example, deskews it) before placing an invisible text layer over it. This type removes unwanted artifacts and may result in a more readable document in some scenarios.

Searchable Image Exact: This type overlays a searchable text layer over the original image, but in this case, the original image is unchanged. This type produces maximum fidelity to the original image.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

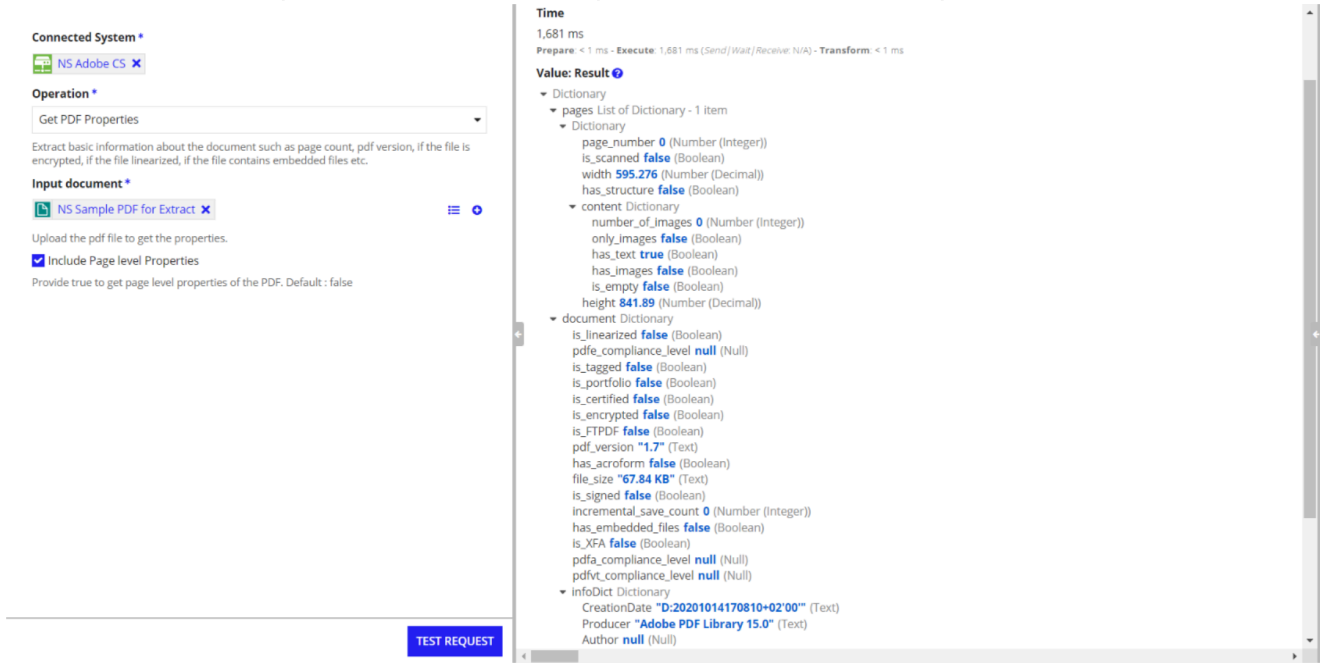
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Get PDF Properties

Gets the metadata properties of a PDF. When used, the metadata including page count, PDF version, file size, compliance levels, font info, permissions and more are provided.



Connected System *

NS Adobe CS

Operation *

Get PDF Properties

Extract basic information about the document such as page count, pdf version, if the file is encrypted, if the file linearized, if the file contains embedded files etc.

Input document *

NS Sample PDF for Extract

Upload the pdf file to get the properties.

[✓] Include Page level Properties

Provide true to get page level properties of the PDF. Default : false

Time

1,681 ms

Prepare: < 1 ms - **Execute**: 1,681 ms (*Send | Wait | Receive*: N/A) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ pages List of Dictionary - 1 item
    ▾ Dictionary
        page_number 0 (Number (Integer))
        is_scanned false (Boolean)
        width 595.276 (Number (Decimal))
        has_structure false (Boolean)
      ▾ content Dictionary
          number_of_images 0 (Number (Integer))
          only_images false (Boolean)
          has_text true (Boolean)
          has_images false (Boolean)
          is_empty false (Boolean)
        height 841.89 (Number (Decimal))
  ▾ document Dictionary
      is_linearized false (Boolean)
      pdfe_compliance_level null (Null)
      is_tagged false (Boolean)
      is_portfolio false (Boolean)
      is_certified false (Boolean)
      is_encrypted false (Boolean)
      is_FTPDF false (Boolean)
      pdf_version "1.7" (Text)
      has_acroform false (Boolean)
      file_size "67.84 KB" (Text)
      is_signed false (Boolean)
      incremental_save_count 0 (Number (Integer))
      has_embedded_files false (Boolean)
      is_XFA false (Boolean)
      pdfa_compliance_level null (Null)
      pdfvt_compliance_level null (Null)
    ▾ infoDict Dictionary
        CreationDate "D:20201014170810+02'00'" (Text)
        Producer "Adobe PDF Library 15.0" (Text)
        Author null (Null)
```

Inputs:

Input document (Document) - Required

Provide the input document.

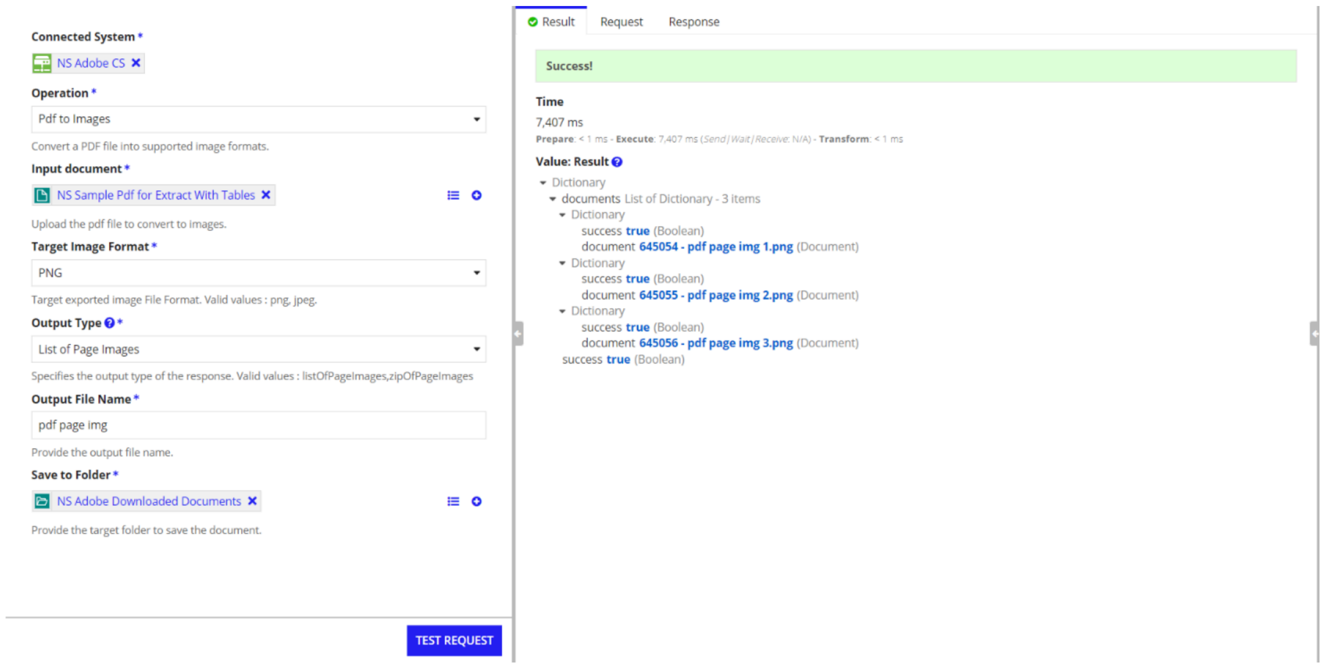
Include Page level Properties (Boolean) - Optional

Provide true to get page level properties of the PDF. Default : false.

Output : (Dictionary) - It contains the information about the pages, document and security in a dictionary format.

PDF to Images

Converts a PDF file into supported image formats(jpeg and png).



Connected System *
NS Adobe CS

Operation *
Pdf to Images
Convert a PDF file into supported image formats.

Input document *
NS Sample Pdf for Extract With Tables
Upload the pdf file to convert to images.

Target Image Format *
PNG
Target exported image File Format. Valid values : png, jpeg.

Output Type *
List of Page Images
Specifies the output type of the response. Valid values : listOfPageImages,zipOfPageImages

Output File Name *
pdf page img
Provide the output file name.

Save to Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

TEST REQUEST

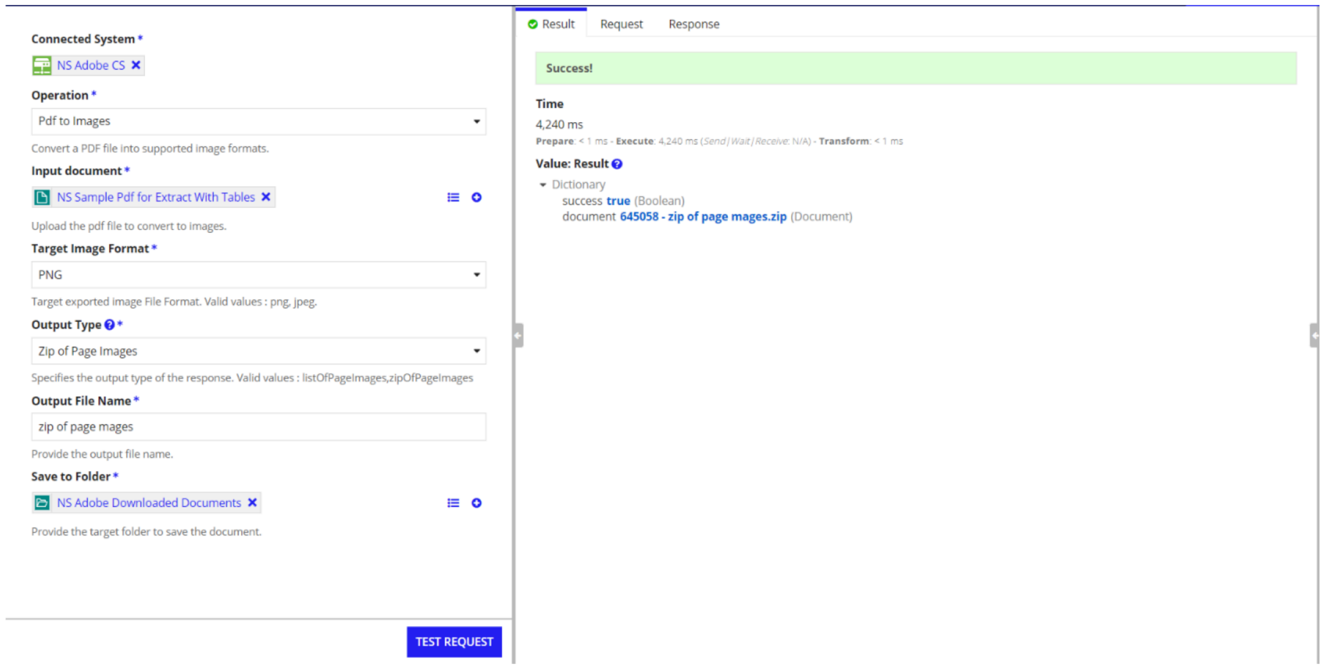
Result | Request | Response

Success!

Time
7,407 ms
Prepare: < 1 ms - Execute: 7,407 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value: Result

```
Dictionary
  documents List of Dictionary - 3 items
    Dictionary
      success true (Boolean)
      document 645054 - pdf page img 1.png (Document)
    Dictionary
      success true (Boolean)
      document 645055 - pdf page img 2.png (Document)
    Dictionary
      success true (Boolean)
      document 645056 - pdf page img 3.png (Document)
  success true (Boolean)
```



Connected System *
NS Adobe CS

Operation *
Pdf to Images
Convert a PDF file into supported image formats.

Input document *
NS Sample Pdf for Extract With Tables
Upload the pdf file to convert to images.

Target Image Format *
PNG
Target exported image File Format. Valid values : png, jpeg.

Output Type *
Zip of Page Images
Specifies the output type of the response. Valid values : listOfPageImages,zipOfPageImages

Output File Name *
zip of page mages
Provide the output file name.

Save to Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

TEST REQUEST

Result | Request | Response

Success!

Time
4,240 ms
Prepare: < 1 ms - Execute: 4,240 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value: Result

```
Dictionary
  success true (Boolean)
  document 645058 - zip of page mages.zip (Document)
```

<u>Input document</u> (Document) - Required
Provide the input document.

<u>Target Image Format</u> (Text) - Required

Description: Target exported image File Format. Valid values : png, jpeg.

Output Type (Text) - Required

Description: Specifies the output type of the response. Valid values: listOfPageImages, zipOfPageImages.

If it is set to zipOfPageImages then the response will be provided as a zip response, otherwise if set as listOfPageImages the response will be provided as a list of images as specified in the targetFormat.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

Description: Provide the target save to the folder where the output file needs to be saved.

Output (List of Images) - Dictionary

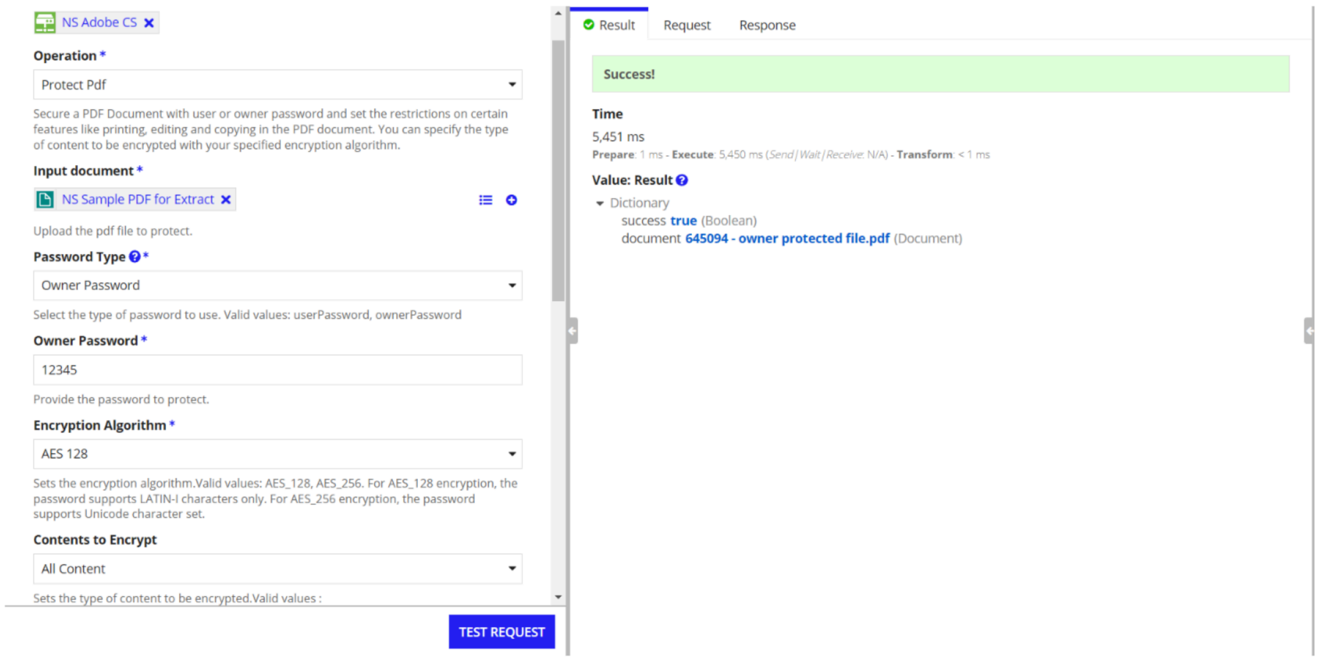
```
Unset
{
success: true,
documents: { (List of Dictionary)
{
success:true,
document : documentId
}
}
}
```

Output (Zip)- Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Protect PDF

Use this integration to secure a PDF file with a password to encrypt the document. Set an owner password and restrictions on certain features like printing, editing and copying in the PDF document to prevent end users from modifying it. You can specify the type of content to be encrypted along with your specified encryption algorithm.



NS Adobe CS

Operation *
Protect Pdf
Secure a PDF Document with user or owner password and set the restrictions on certain features like printing, editing and copying in the PDF document. You can specify the type of content to be encrypted with your specified encryption algorithm.

Input document *
NS Sample PDF for Extract
Upload the pdf file to protect.

Password Type *
Owner Password
Select the type of password to use. Valid values: userPassword, ownerPassword

Owner Password *
12345
Provide the password to protect.

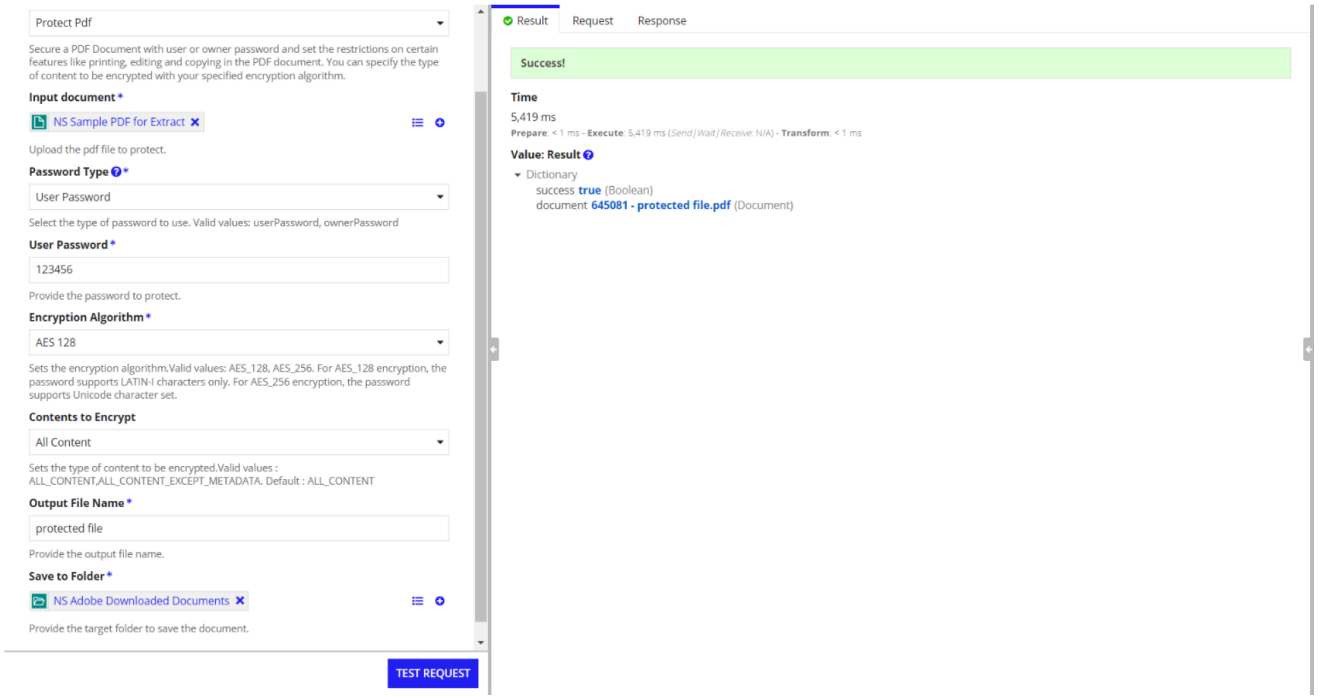
Encryption Algorithm *
AES 128
Sets the encryption algorithm.Valid values: AES_128, AES_256. For AES_128 encryption, the password supports LATIN-I characters only. For AES_256 encryption, the password supports Unicode character set.

Contents to Encrypt
All Content
Sets the type of content to be encrypted.Valid values :

Success!

Time
5,451 ms
Prepare: 1 ms - **Execute**: 5,450 ms (*Send/Wait/Receive*: N/A) - **Transform**: < 1 ms

Value: Result
Dictionary
success **true** (Boolean)
document **645094 - owner protected file.pdf** (Document)


Protect Pdf

Secure a PDF Document with user or owner password and set the restrictions on certain features like printing, editing and copying in the PDF document. You can specify the type of content to be encrypted with your specified encryption algorithm.

Input document *

NS Sample PDF for Extract

Upload the pdf file to protect.

Password Type *

User Password

Select the type of password to use. Valid values: userPassword, ownerPassword

User Password *

123456

Provide the password to protect.

Encryption Algorithm *

AES 128

Sets the encryption algorithm.Valid values: AES_128, AES_256. For AES_128 encryption, the password supports LATIN-I characters only. For AES_256 encryption, the password supports Unicode character set.

Contents to Encrypt

All Content

Sets the type of content to be encrypted.Valid values : ALL_CONTENT,ALL_CONTENT_EXCEPT_METADATA. Default : ALL_CONTENT

Output File Name *

protected file

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

TEST REQUEST

Result | Request | Response

Success!

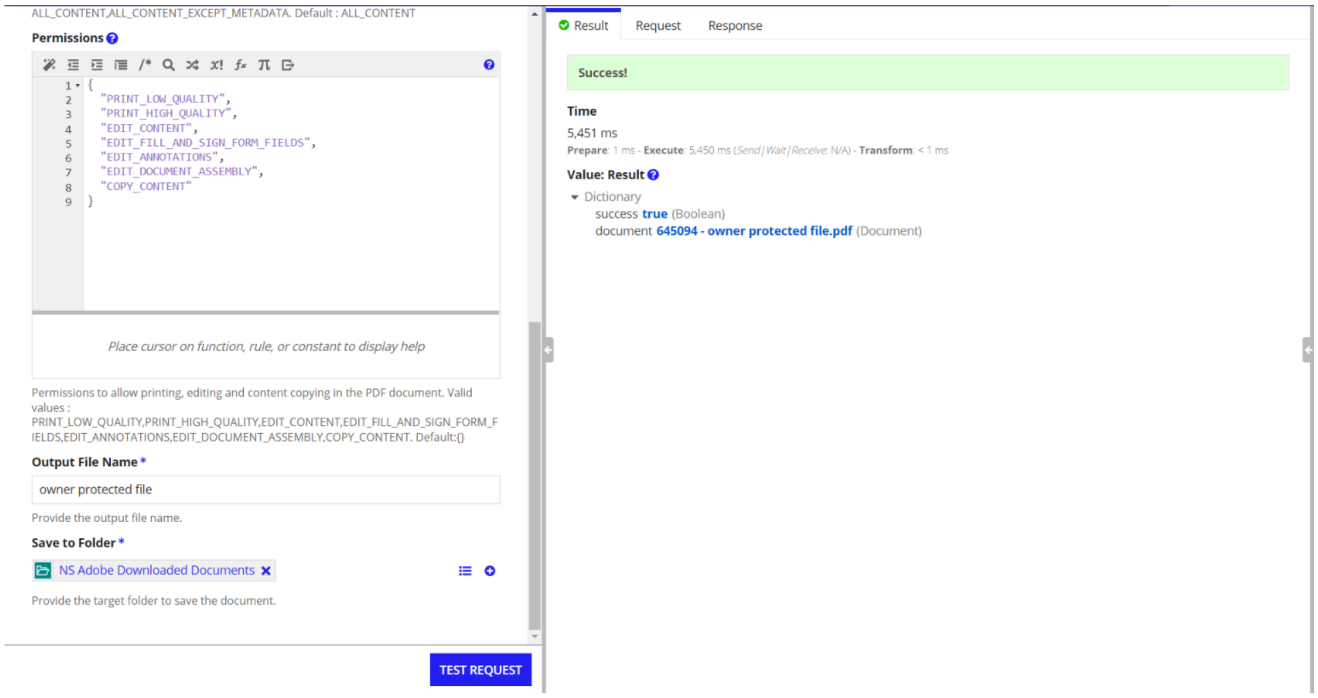
Time

5,419 ms

Prepare: < 1 ms - Execute: 5,419 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value: Result

Dictionary
success **true** (Boolean)
document **645081 - protected file.pdf** (Document)


ALL_CONTENT,ALL_CONTENT_EXCEPT_METADATA. Default : ALL_CONTENT

Permissions

```
{
  "PRINT_LOW_QUALITY",
  "PRINT_HIGH_QUALITY",
  "EDIT_CONTENT",
  "EDIT_FILL_AND_SIGN_FORM_FIELDS",
  "EDIT_ANNOTATIONS",
  "EDIT_DOCUMENT_ASSEMBLY",
  "COPY_CONTENT"
}
```

Place cursor on function, rule, or constant to display help

Permissions to allow printing, editing and content copying in the PDF document. Valid values : PRINT_LOW_QUALITY,PRINT_HIGH_QUALITY,EDIT_CONTENT,EDIT_FILL_AND_SIGN_FORM_FIELDS,EDIT_ANNOTATIONS,EDIT_DOCUMENT_ASSEMBLY,COPY_CONTENT. Default:{}

Output File Name *

owner protected file

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

TEST REQUEST

Result | Request | Response

Success!

Time

5,451 ms

Prepare: 1 ms - Execute: 5,450 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value: Result

Dictionary
success **true** (Boolean)
document **645094 - owner protected file.pdf** (Document)

Inputs:

Input document (Document) - Required

Description: Provide the input document to linearize it.

Password Type (Text) - Required

Description: Provide the type of password to use. Valid values: userPassword, ownerPassword.

User Password: Password used to open an encrypted PDF document. When this property is included and non-empty, the use of a password is necessary to open or view the document. If this password is empty or omitted the document can be opened automatically by conforming PDF viewers.

Owner Password: Password used to control permissions (does not add password to view the PDF) in a PDF document. Conforming PDF viewers require this password to change the permissions. This password can also be used to open/view the PDF document.

User Password (Text) - Required (or) **Owner Password** (Text) - Required
Description: Provide the password value to protect with.
Encryption Algorithm (Text) - Required

Description: Sets the encryption algorithm.Valid values: AES_128, AES_256. For AES_128 encryption, the password supports LATIN-I characters only. For AES_256 encryption, the password supports the Unicode character set.

Content to Encrypt: (Text) - Optional

Description: Sets the type of content to be encrypted. Valid values: ALL_CONTENT, ALL_CONTENT_EXCEPT_METADATA. Default: ALL_CONTENT.

Permissions (List of Text) - Optional - Available only when type is Owner Password.

Description: Permissions to allow printing, editing and content copying in the PDF document. Valid values: PRINT_LOW_QUALITY, PRINT_HIGH_QUALITY, EDIT_CONTENT, EDIT_FILL_AND_SIGN_FORM_FIELDS, EDIT_ANNOTATIONS, EDIT_DOCUMENT_ASSEMBLY, COPY_CONTENT. Default:{}

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

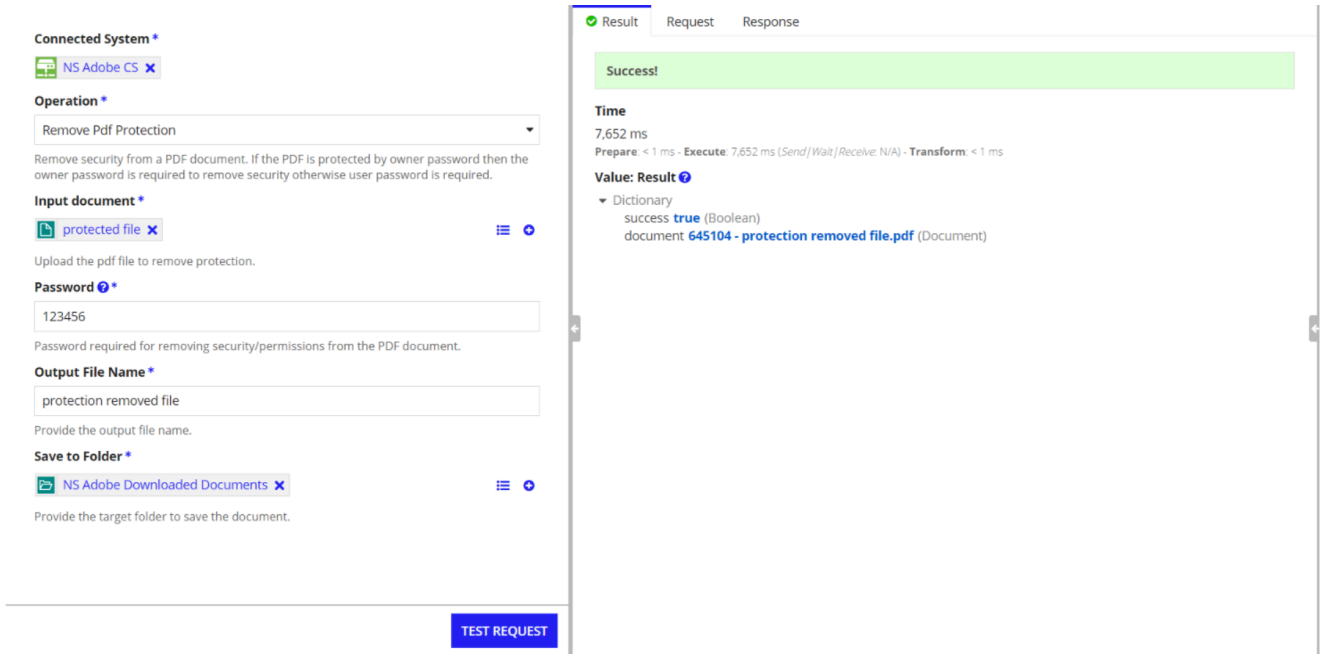
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Remove PDF Protection

Use this integration to remove security configurations from a PDF document. If the PDF is protected by an owner password then the owner password is required to remove security otherwise user password is required.


Connected System *
NS Adobe CS
Operation *
Remove Pdf Protection
Remove security from a PDF document. If the PDF is protected by owner password then the owner password is required to remove security otherwise user password is required.
Input document *
protected file
Upload the pdf file to remove protection.
Password *
123456
Password required for removing security/permissions from the PDF document.
Output File Name *
protection removed file
Provide the output file name.
Save to Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

Success!
Time
7,652 ms
Prepare: < 1 ms - Execute: 7,652 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms
Value: Result
Dictionary
success true (Boolean)
document 645104 - protection removed file.pdf (Document)

Inputs:

Input document (Document) - Required

Description: Provide the input document with password protected.

Password (Text) - Required

Description: Password required for removing security/permissions from the PDF document.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

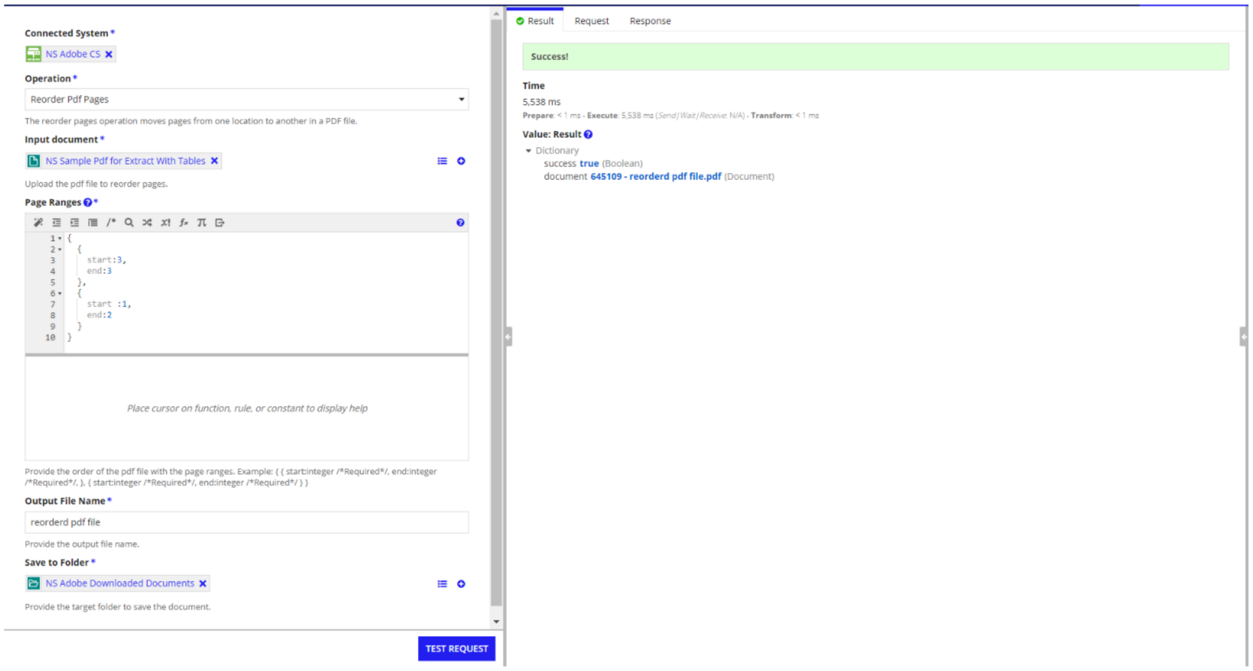
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Reorder PDF Pages

Reorders pages in a PDF by moving pages from one position to another.



Connected System *
NS Adobe CS

Operation *
Reorder Pdf Pages
The reorder pages operation moves pages from one location to another in a PDF file.

Input document *
NS Sample Pdf for Extract With Tables
Upload the pdf file to reorder pages.

Page Ranges *
```
{
  {
    start:3,
    end:3
  },
  {
    start :1,
    end:2
  }
}
```
Place cursor on function, rule, or constant to display help

Provide the order of the pdf file with the page ranges. Example: { { start:integer /*Required*/, end:integer /*Required*/, }, { start:integer /*Required*/, end:integer /*Required*/ } }

Output File Name *
reorderd pdf file
Provide the output file name.

Save to Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

Success!

Time
5,538 ms
Prepare: < 1 ms - Execute: 5,538 ms (*Send/Wait/Receive*: N/A) - Transform: < 1 ms

Value: Result
```
Dictionary
    success true (Boolean)
    document 645109 - reorderd pdf file.pdf (Document)
```

Inputs:

<u>Input document</u> (Document) - Required

Provide the input document.

<u>Page Ranges:</u> (List of Dictionary) - Required

Page ranges of the PDF file. Each dictionary confirms the start and end.

Example :

```
Unset
{
{
start: integer /*Required*/,
end: integer /*Required*/
}
}
```

In pageRanges to pass only a single page, provide the start and end value as the same. 29
Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

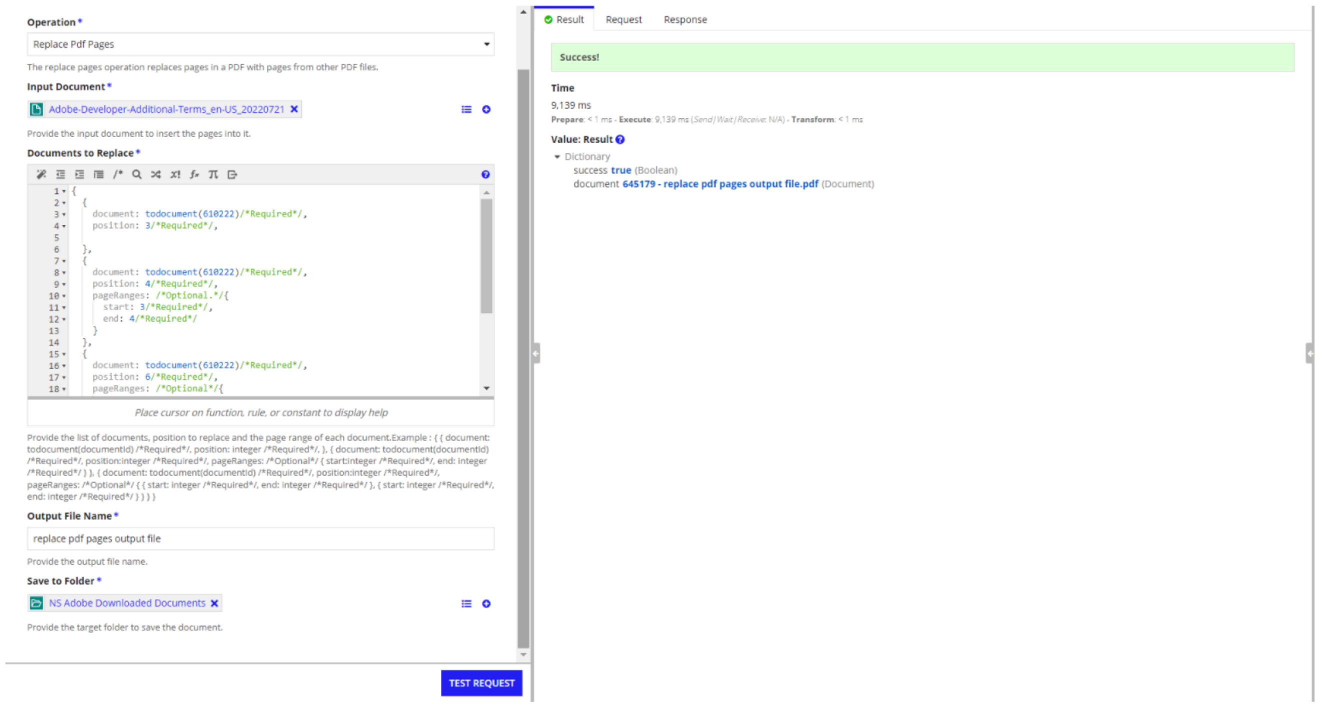
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Replace PDF Pages

Replaces pages in a PDF with pages from other PDF files.



Operation *

Replace Pdf Pages

The replace pages operation replaces pages in a PDF with pages from other PDF files.

Input Document *

Adobe-Developer-Additional-Terms_en-US_20220721

Provide the input document to insert the pages into it.

Documents to Replace *

```
{
  {
    document: todocument(610222)/*Required*/,
    position: 3/*Required*/,

  },
  {
    document: todocument(610222)/*Required*/,
    position: 4/*Required*/,
    pageRanges: /*Optional.*/{
      start: 3/*Required*/,
      end: 4/*Required*/
    }
  },
  {
    document: todocument(610222)/*Required*/,
    position: 6/*Required*/,
    pageRanges: /*Optional*/{
```

Place cursor on function, rule, or constant to display help

Provide the list of documents, position to replace and the page range of each document.Example : { { document: todocument(documentId) /*Required*/, position: integer /*Required*/, }, { document: todocument(documentId) /*Required*/, position:integer /*Required*/, pageRanges: /*Optional*/ { start:integer /*Required*/, end: integer /*Required*/ } }, { document: todocument(documentId) /*Required*/, position:integer /*Required*/, pageRanges: /*Optional*/ { { start: integer /*Required*/, end: integer /*Required*/ }, { start: integer /*Required*/, end: integer /*Required*/ } } } }

Output File Name *

replace pdf pages output file

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time

9,139 ms

Prepare: < 1 ms - **Execute**: 9,139 ms (*Send/Wait/Receive*: N/A) - **Transform**: < 1 ms

Value: Result

Dictionary
success **true** (Boolean)
document **645179 - replace pdf pages output file.pdf** (Document)

Input:
Input Document (Document) - Required
Description: Provide the input document to replace the pages.

Documents to Insert (List of Dictionary) - Required

Example:

```
Unset

{

{

document: todocument(documentId)/*Required*/

position: integer /*Required*/

},

{
```

```
document: todocument(documentId)/*Required*/,

position: integer /*Required*/,

pageRanges: /*Optional*/{

{

start: integer /*Required*/,

end: integer /*Required*/,

},

{

start: integer /*Required*/,

end: integer /*Required*/,

}

}

},

{

document: todocument(documentId)/*Required*/,

position: integer /*Required*/,

pageRanges: {}/*Optional*/

}

}
```

Description: Each dictionary requires a document and a position at where the document needs to be replaced in the input document section. The pageRanges is optional, however, if provided the start and end inside pageRanges are required. Provide the value for the document using the todocument() function. In pageRanges to pass only a single page, provide the start and end value as the same. Page numbers are indexed from 1 to N.

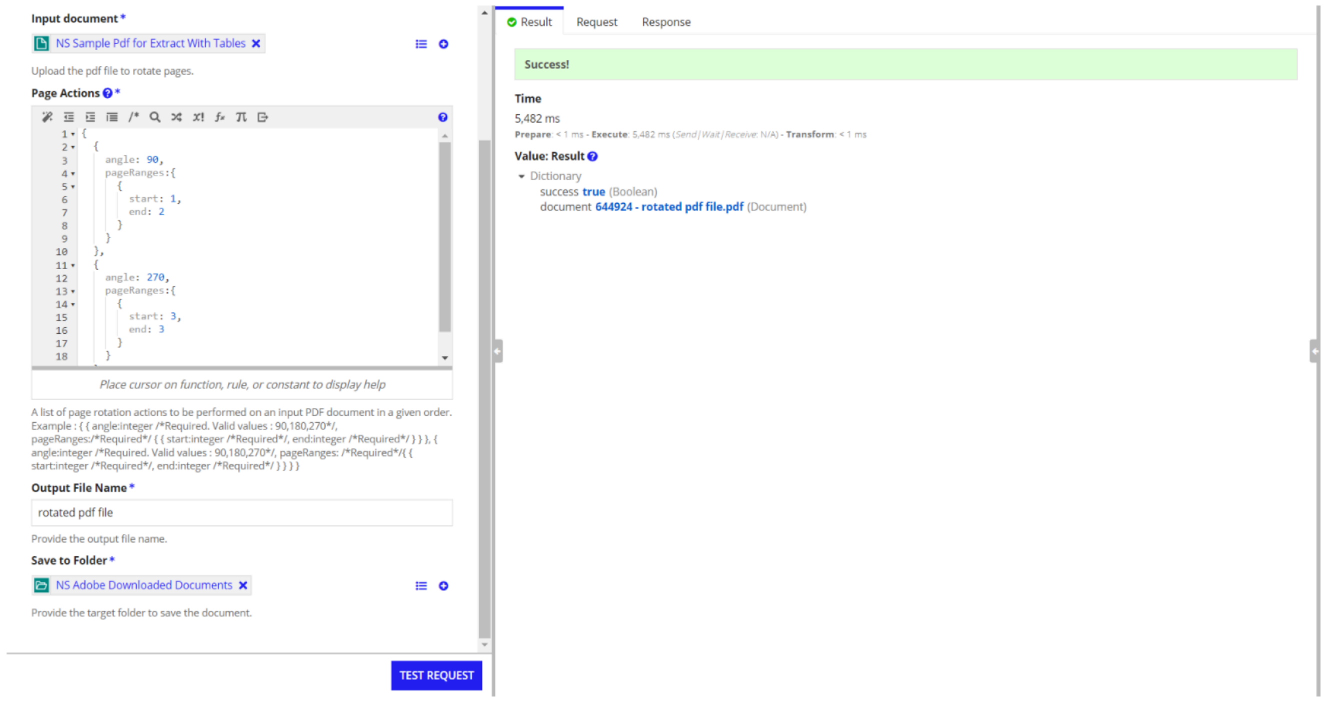
Output File Name (Text) - Required
Description: Provide the name of the output file.

Save to Folder (Folder) - Required
Description: Provide the target save to folder where the output file needs to be saved.
Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Rotate PDF Pages

Selectively rotates pages in a PDF document. For example, you can change portrait view to landscape view of certain pages from a PDF document.


Input document *

NS Sample Pdf for Extract With Tables

Upload the pdf file to rotate pages.

Page Actions *

```
{
  {
    angle: 90,
    pageRanges:{
      {
        start: 1,
        end: 2
      }
    }
  },
  {
    angle: 270,
    pageRanges:{
      {
        start: 3,
        end: 3
      }
    }
```

Place cursor on function, rule, or constant to display help

A list of page rotation actions to be performed on an input PDF document in a given order. Example : { { angle:integer /*Required. Valid values : 90,180,270*/, pageRanges:/*Required*/ { { start:integer /*Required*/, end:integer /*Required*/ } } }, { angle:integer /*Required. Valid values : 90,180,270*/, pageRanges: /*Required*/{ { start:integer /*Required*/, end:integer /*Required*/ } } } }

Output File Name *

rotated pdf file

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time

5,482 ms

Prepare: < 1 ms - Execute: 5,482 ms (*Send / Wait / Receive*: N/A) - Transform: < 1 ms

Value: Result

Dictionary
success **true** (Boolean)
document **644924 - rotated pdf file.pdf** (Document)

Inputs:

Input document (Document) - Required
Provide the input document.

Page Actions (List of Dictionary) - Required

A list of page actions to be performed on an input PDF document in the given order. Example

```
Unset
{
{
angle:integer /*Required. Valid values : 90,180,270*/, pageRanges:/*Required*/ {
{
start:integer /*Required*/,
end:integer /*Required*/
}
}
},
{
```

```
angle:integer /*Required. Valid values : 90,180,270*/, pageRanges: /*Required*/{
{
start:integer /*Required*/,
end:integer /*Required*/
}
}
}
}
```

Description:
angle (Integer) - Required: It specifies the clockwise rotation angle relative to the starting orientation of the page. e.g. if a page is already rotated 90 degrees (landscape), specifying a rotation of 90 degrees will rotate it a further 90 degrees. The valid rotation angles are: 90, 180, 270.
pageRanges (List of dictionary) - Required: Each dictionary contains start and end values. To pass only a single page, provide the start and end value as the same.Page numbers are indexed from 1 to N.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Save to Folder (Folder) - Required

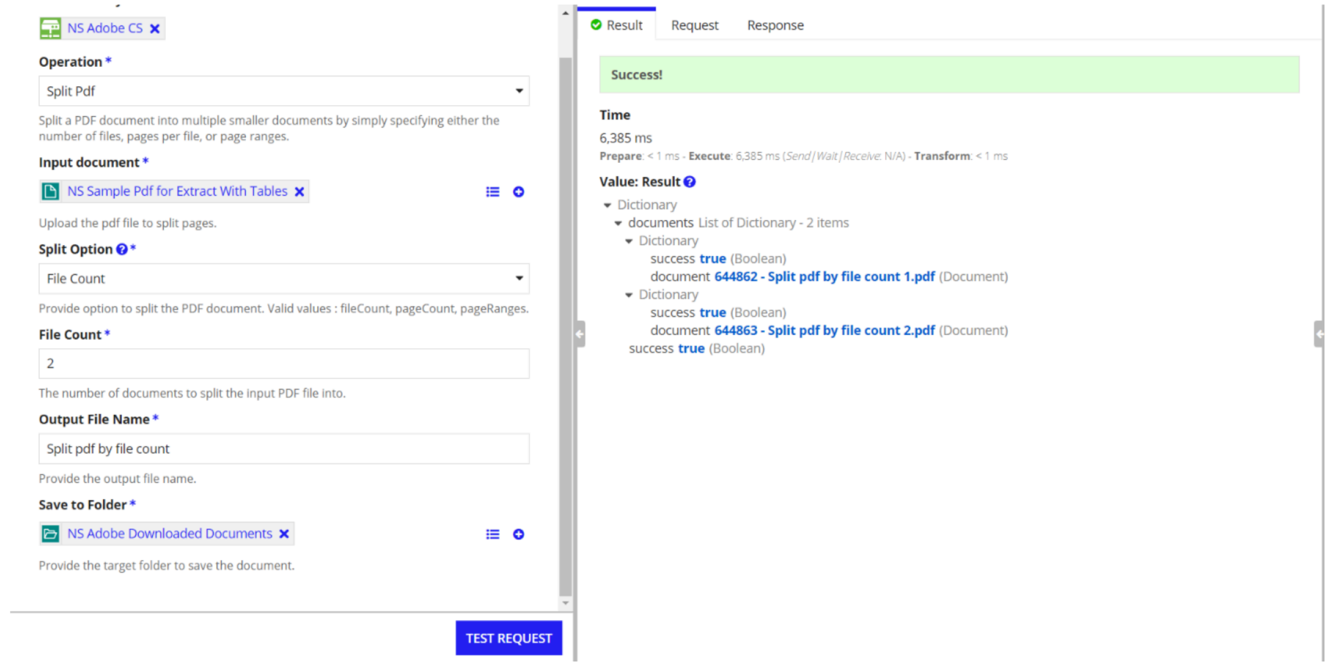
Description: Provide the target save to the folder where the output file needs to be saved.

Output - Dictionary

```
Unset
{
success: true,
document: documentId - Output file name
}
```

Split PDF

Splits a PDF document into multiple smaller documents by simply specifying either the number of files, pages per file, or page ranges.



NS Adobe CS

Operation *

Split Pdf

Split a PDF document into multiple smaller documents by simply specifying either the number of files, pages per file, or page ranges.

Input document *

NS Sample Pdf for Extract With Tables

Upload the pdf file to split pages.

Split Option *

File Count

Provide option to split the PDF document. Valid values : fileCount, pageCount, pageRanges.

File Count *

2

The number of documents to split the input PDF file into.

Output File Name *

Split pdf by file count

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

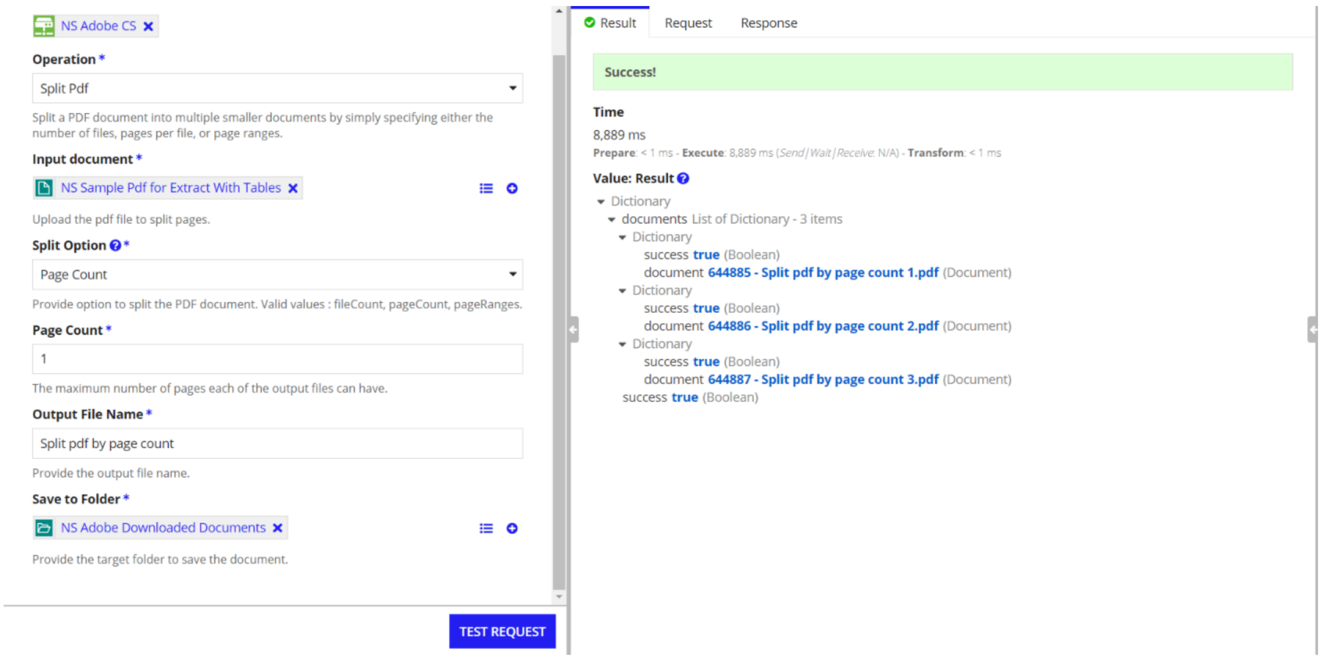
Time

6,385 ms

Prepare: < 1 ms - **Execute**: 6,385 ms (*Send/Wait/Receive*: N/A) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ documents List of Dictionary - 2 items
    ▾ Dictionary
        success true (Boolean)
        document 644862 - Split pdf by file count 1.pdf (Document)
    ▾ Dictionary
        success true (Boolean)
        document 644863 - Split pdf by file count 2.pdf (Document)
  success true (Boolean)
```



NS Adobe CS

Operation *

Split Pdf

Split a PDF document into multiple smaller documents by simply specifying either the number of files, pages per file, or page ranges.

Input document *

NS Sample Pdf for Extract With Tables

Upload the pdf file to split pages.

Split Option *

Page Count

Provide option to split the PDF document. Valid values : fileCount, pageCount, pageRanges.

Page Count *

1

The maximum number of pages each of the output files can have.

Output File Name *

Split pdf by page count

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

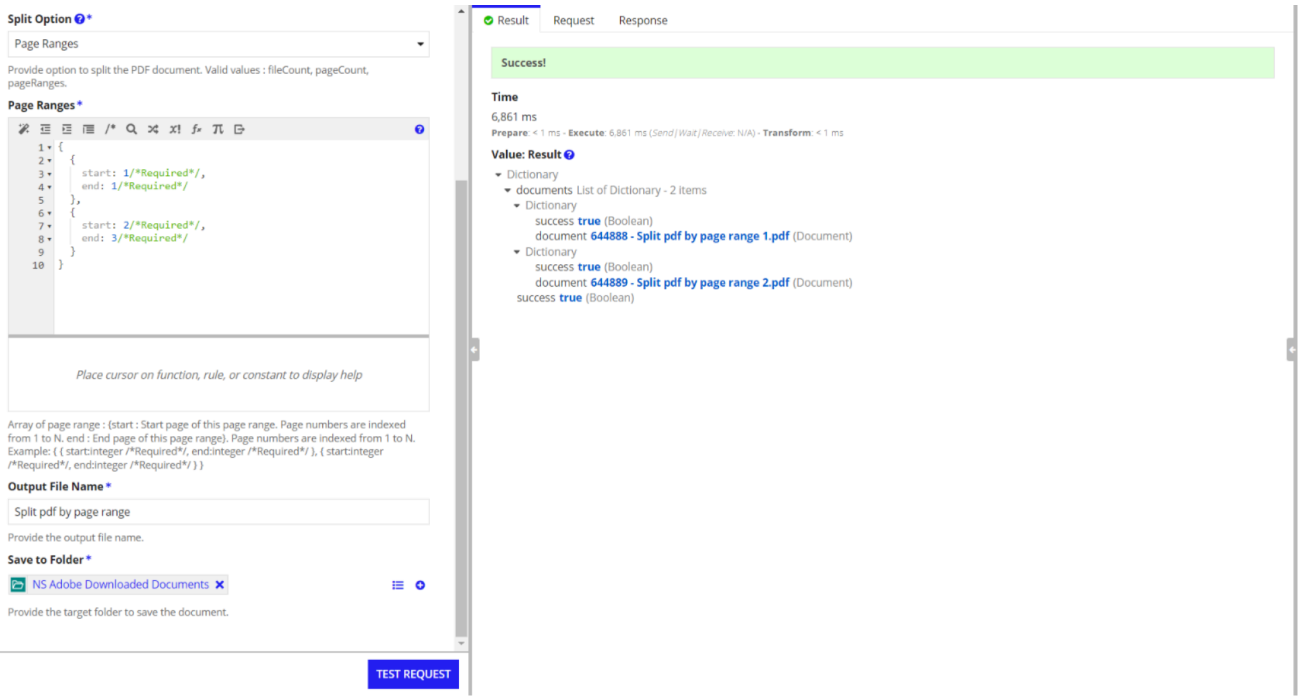
Time

8,889 ms

Prepare: < 1 ms - **Execute**: 8,889 ms (*Send / Wait / Receive*: N/A) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ documents List of Dictionary - 3 items
    ▾ Dictionary
        success true (Boolean)
        document 644885 - Split pdf by page count 1.pdf (Document)
    ▾ Dictionary
        success true (Boolean)
        document 644886 - Split pdf by page count 2.pdf (Document)
    ▾ Dictionary
        success true (Boolean)
        document 644887 - Split pdf by page count 3.pdf (Document)
  success true (Boolean)
```



Split Option *

Page Ranges

Provide option to split the PDF document. Valid values : fileCount, pageCount, pageRanges.

Page Ranges *

```
{
  {
    start: 1/*Required*/,
    end: 1/*Required*/
  },
  {
    start: 2/*Required*/,
    end: 3/*Required*/
  }
}
```

Place cursor on function, rule, or constant to display help

Array of page range : {start : Start page of this page range. Page numbers are indexed from 1 to N. end : End page of this page range}. Page numbers are indexed from 1 to N. Example: { { start:integer /*Required*/, end:integer /*Required*/ }, { start:integer /*Required*/, end:integer /*Required*/ } }

Output File Name *

Split pdf by page range

Provide the output file name.

Save to Folder *

NS Adobe Downloaded Documents

Provide the target folder to save the document.

Success!

Time

6,861 ms

Prepare: < 1 ms - **Execute**: 6,861 ms (*Send / Wait / Receive*: N/A) - **Transform**: < 1 ms

Value: Result

```
▾ Dictionary
  ▾ documents List of Dictionary - 2 items
    ▾ Dictionary
        success true (Boolean)
        document 644888 - Split pdf by page range 1.pdf (Document)
    ▾ Dictionary
        success true (Boolean)
        document 644889 - Split pdf by page range 2.pdf (Document)
  success true (Boolean)
```

Inputs:

<u>Input document</u> (Document) - Required
Provide the input document.

<u>Split Option</u> (Text) - Required.
Provide an option to split the PDF document. Valid values : fileCount, pageCount, pageRanges.

fileCount : The number of documents to split the input PDF file into.
(Integer) - Required

pageCount : The maximum number of pages each of the output files can have.
(Integer) - Required

pageRanges: (List of dictionary) - Required. Each dictionary contains start and end values.
To pass only a single page, provide the start and end value as the same.Page numbers are indexed from 1 to N.
Example:

```
Unset
{
{
start: integer/*Required*/,
end: integer/*Required*/
},
{
start: integer/*Required*/, end: integer/*Required*/ }
}
```

Output File Name (Text) - Required

Description: Provide the name of the output file.

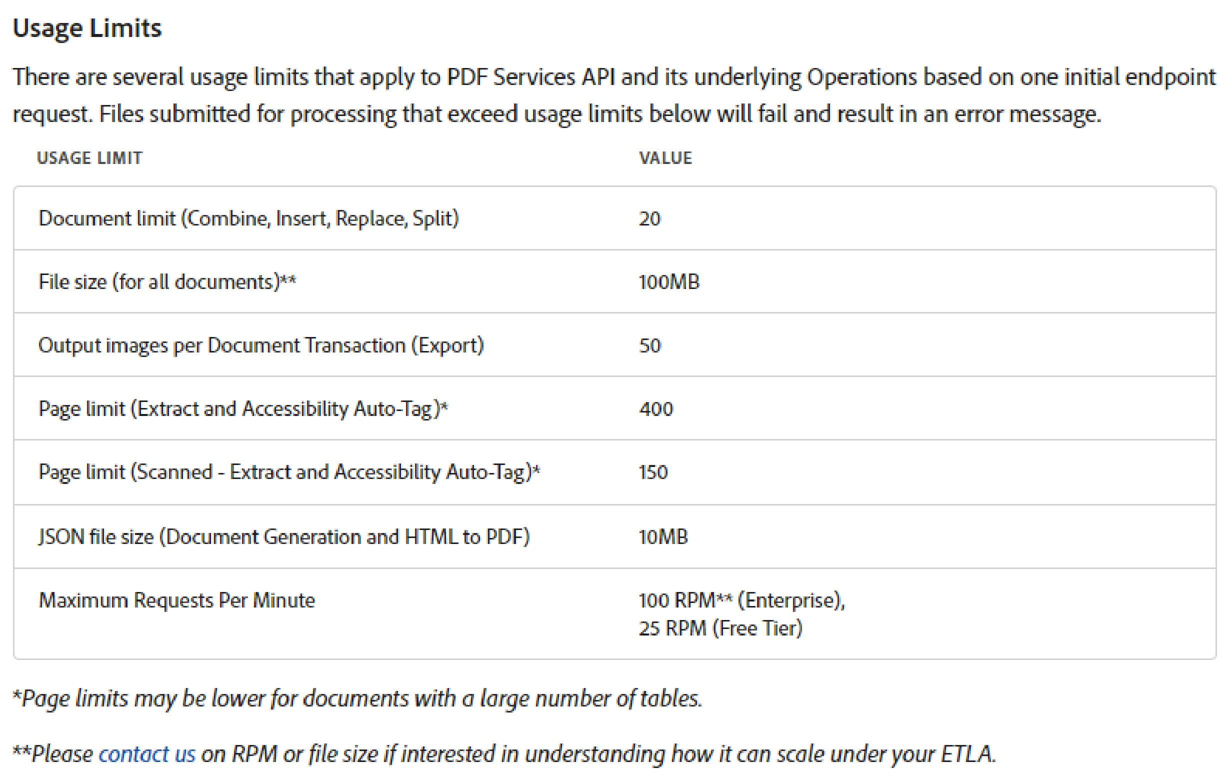
Save to Folder (Folder) - Required

Description: Provide the target save to the folder where the output file needs to be saved. **Output** - Dictionary

```
Unset
{
success: true,
documents: { (List of Dictionary)
{
success:true,
document : documentId
}
}
}
```

Important Note: If a set of PDF pages cannot be evenly separated in files then this integration will split them by the requested number of pages and put remaining pages into a separate file.

Usage Limits



Usage Limits

There are several usage limits that apply to PDF Services API and its underlying Operations based on one initial endpoint request. Files submitted for processing that exceed usage limits below will fail and result in an error message.

USAGE LIMIT	VALUE
Document limit (Combine, Insert, Replace, Split)	20
File size (for all documents)**	100MB
Output images per Document Transaction (Export)	50
Page limit (Extract and Accessibility Auto-Tag)*	400
Page limit (Scanned - Extract and Accessibility Auto-Tag)*	150
JSON file size (Document Generation and HTML to PDF)	10MB
Maximum Requests Per Minute	100 RPM** (Enterprise), 25 RPM (Free Tier)

**Page limits may be lower for documents with a large number of tables.*

***Please contact us on RPM or file size if interested in understanding how it can scale under your ETLA.*

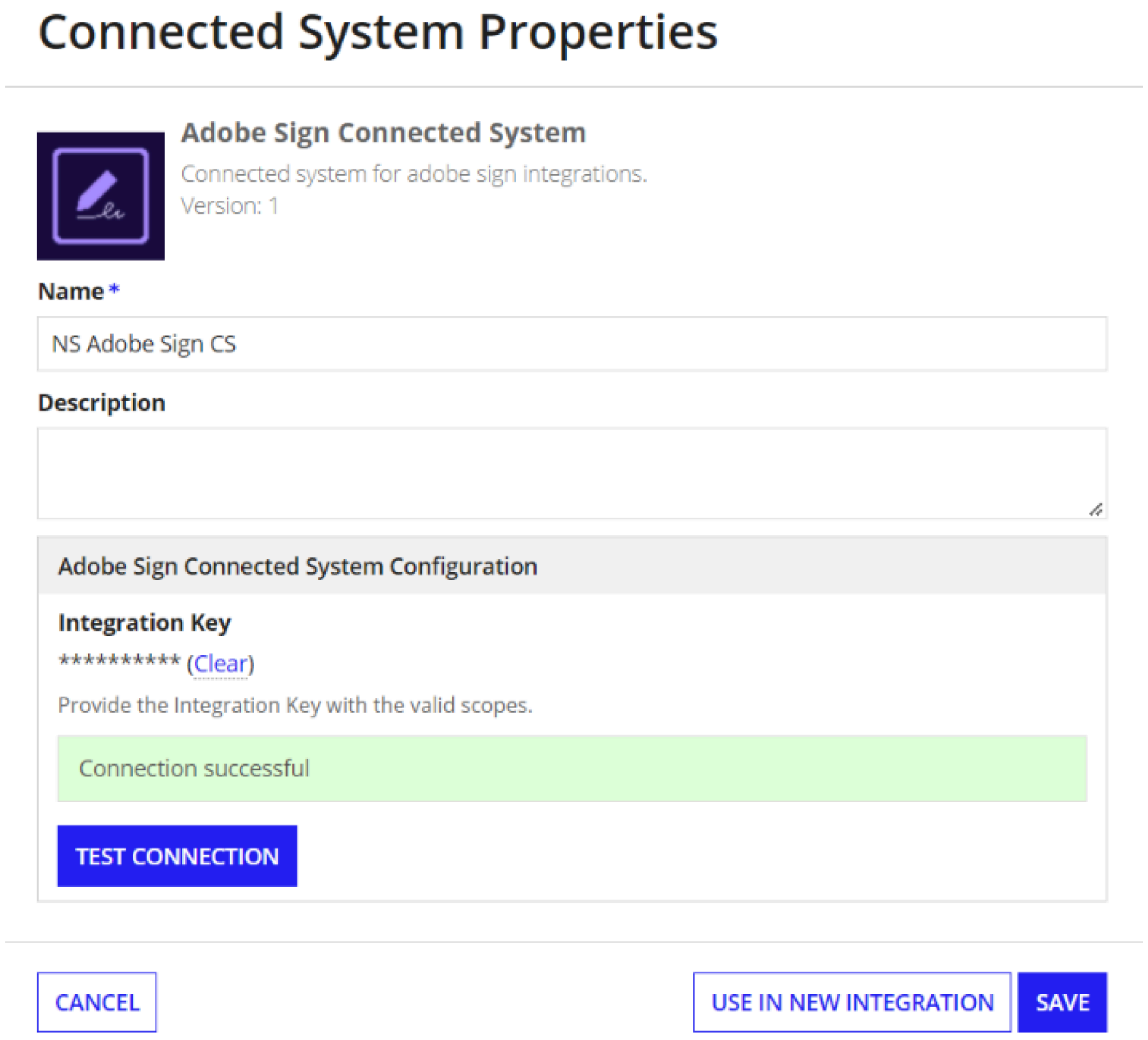
Please refer here for the usage limits of the Adobe PDF Services API.

Adobe Sign

Overview

Adobe Sign API is a great way to enhance how you manage signed agreements. You can easily integrate with Sign API which provides a reliable, easy, and quick way to upload and manage documents, send documents for signing, send reminders to signatories, and manage e-signatures.

Connected System Configuration



Connected System Properties

Adobe Sign Connected System

Connected system for adobe sign integrations.

Version: 1

Name *

NS Adobe Sign CS

Description

Adobe Sign Connected System Configuration

Integration Key

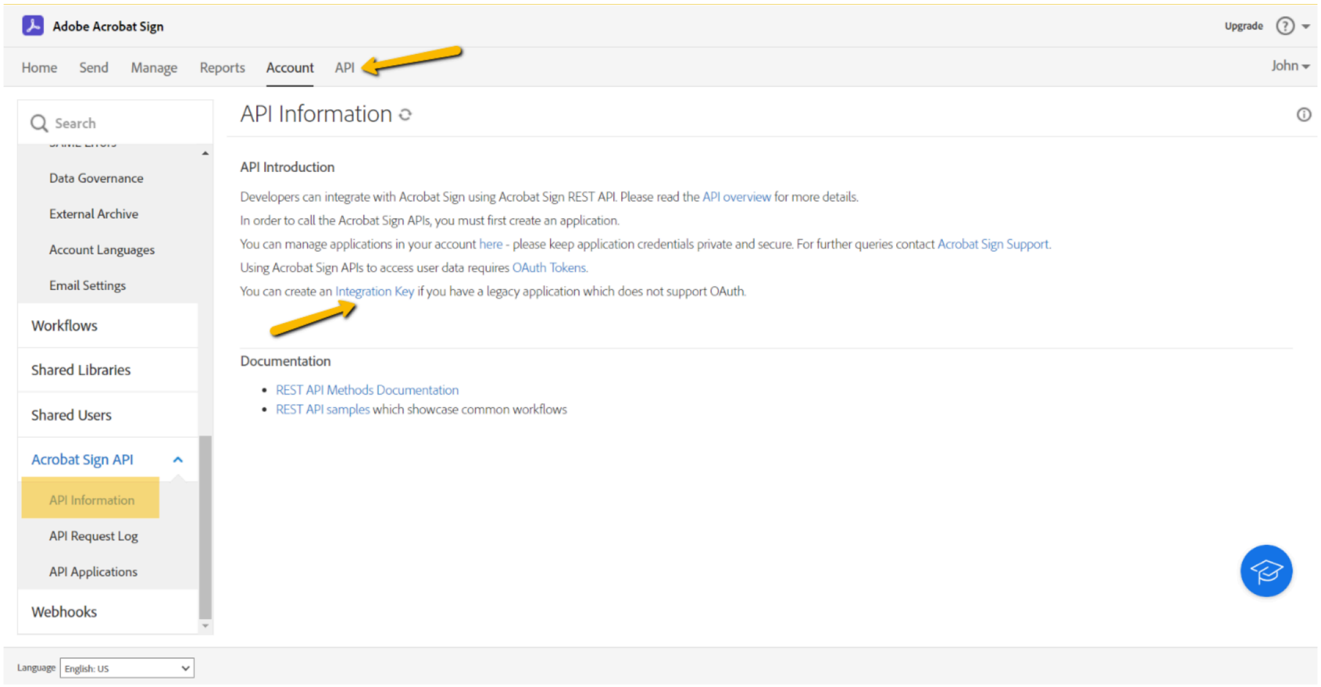
*********** (Clear)

Provide the Integration Key with the valid scopes.

Connection successful

The Adobe Sign Connected System requires an Integration Key for its authentication.

1. To get the Integration Key, you need an Adobe Developer/Enterprise account.
2. Click here to create a free Adobe Sign Developer Account or login to your account if you already have one.
3. After account creation, please login to your Adobe Account. Navigate to API > Acrobat Sign API > API Information and click on Integration Key link.



API Information

API Introduction

Developers can integrate with Acrobat Sign using Acrobat Sign REST API. Please read the API overview for more details.

In order to call the Acrobat Sign APIs, you must first create an application.

You can manage applications in your account here - please keep application credentials private and secure. For further queries contact Acrobat Sign Support.

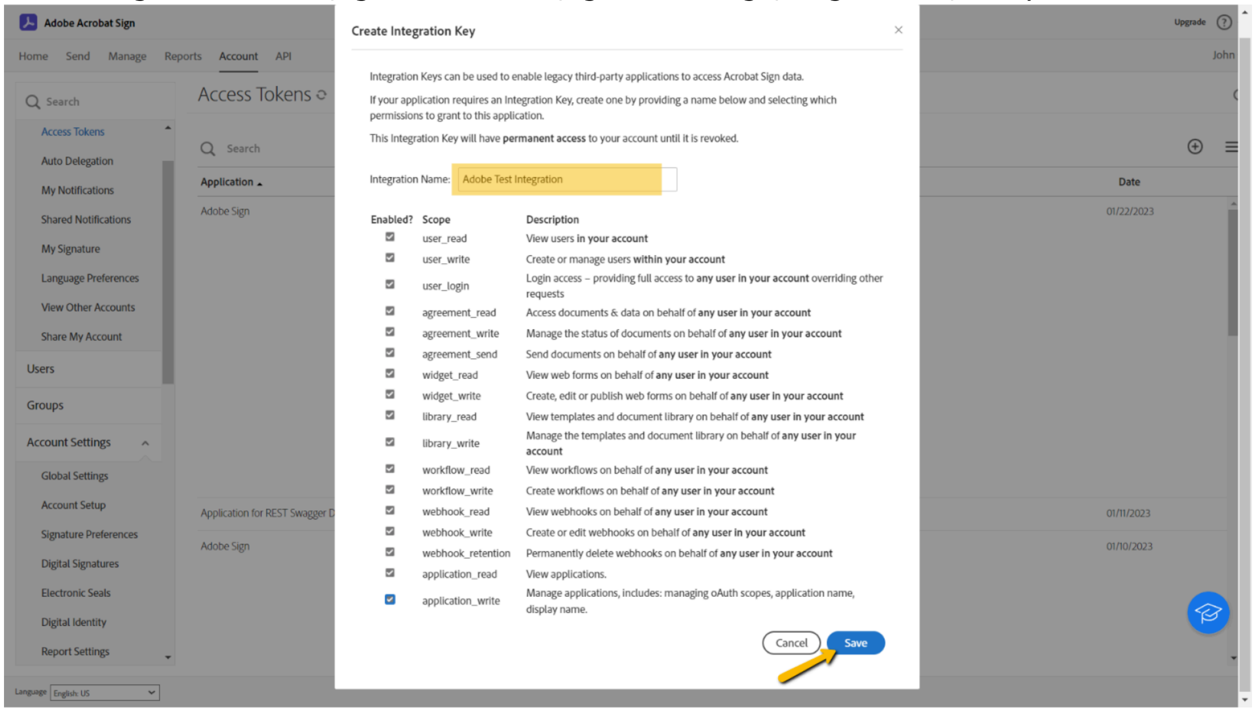
Using Acrobat Sign APIs to access user data requires OAuth Tokens.

You can create an Integration Key if you have a legacy application which does not support OAuth.

Documentation

- REST API Methods Documentation
- REST API samples which showcase common workflows

4. Provide a name for your Integration and select the necessary scopes and select Save. Please make sure the following scopes are selected: agreement_read,agreement_write,agreement_sign,widget_write,library_write.



Create Integration Key

Integration Keys can be used to enable legacy third-party applications to access Acrobat Sign data.

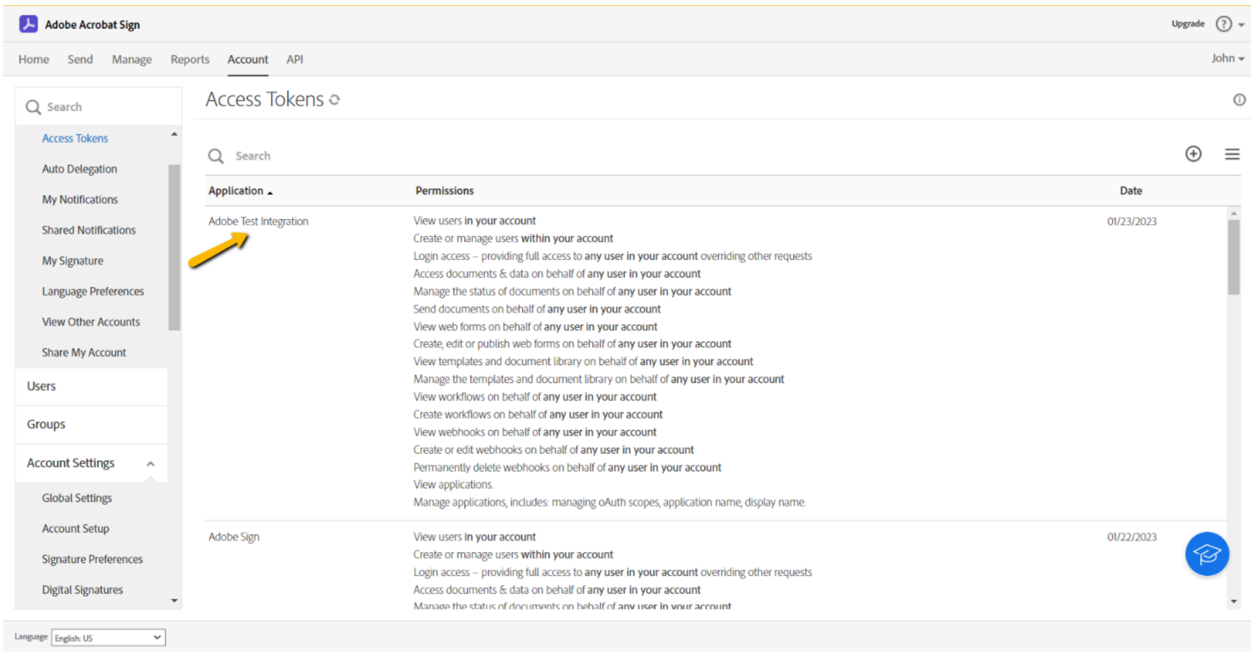
If your application requires an Integration Key, create one by providing a name below and selecting which permissions to grant to this application.

This Integration Key will have **permanent access** to your account until it is revoked.

Integration Name: Adobe Test Integration

Enabled?	Scope	Description
✓	user_read	View users **in your account**
✓	user_write	Create or manage users **within your account**
✓	user_login	Login access – providing full access to **any user in your account** overriding other requests
✓	agreement_read	Access documents & data on behalf of **any user in your account**
✓	agreement_write	Manage the status of documents on behalf of **any user in your account**
✓	agreement_send	Send documents on behalf of **any user in your account**
✓	widget_read	View web forms on behalf of **any user in your account**
✓	widget_write	Create, edit or publish web forms on behalf of **any user in your account**
✓	library_read	View templates and document library on behalf of **any user in your account**
✓	library_write	Manage the templates and document library on behalf of **any user in your account**
✓	workflow_read	View workflows on behalf of **any user in your account**
✓	workflow_write	Create workflows on behalf of **any user in your account**
✓	webhook_read	View webhooks on behalf of **any user in your account**
✓	webhook_write	Create or edit webhooks on behalf of **any user in your account**
✓	webhook_retention	Permanently delete webhooks on behalf of **any user in your account**
✓	application_read	View applications.
✓	application_write	Manage applications, includes: managing oAuth scopes, application name, display name.

Now you can see the List of Applications in your account along with the permissions.



Adobe Acrobat Sign — Upgrade — John

Home | Send | Manage | Reports | Account | API

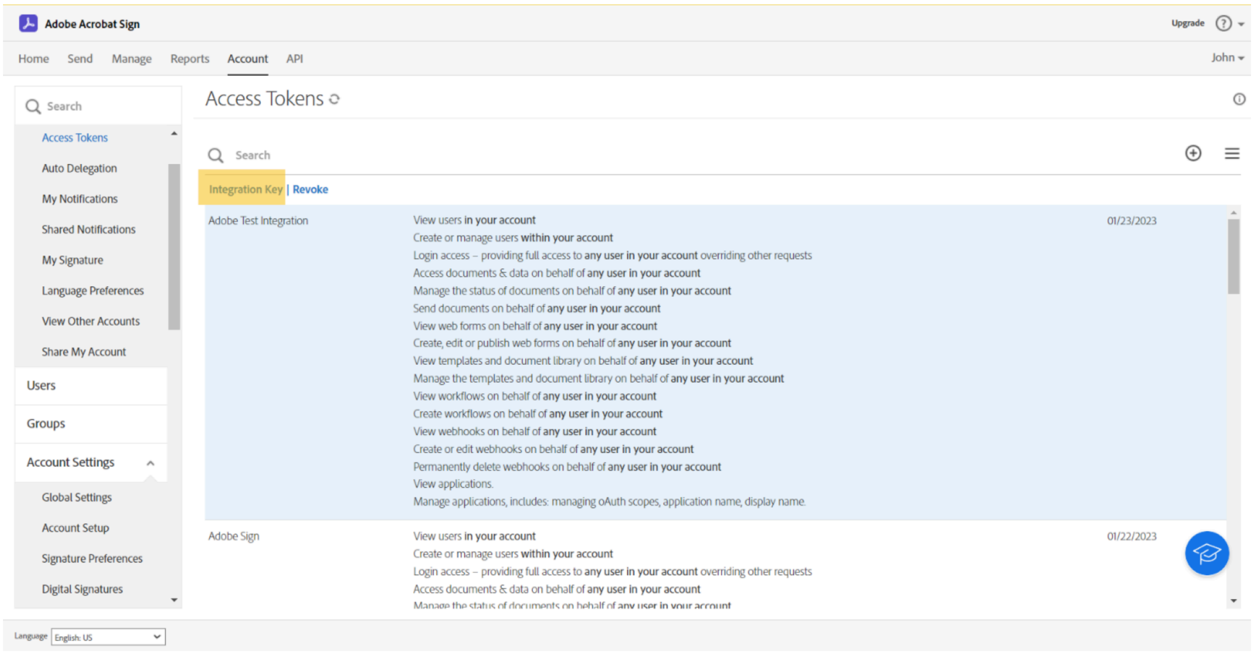
Search: Access Tokens, Auto Delegation, My Notifications, Shared Notifications, My Signature, Language Preferences, View Other Accounts, Share My Account; Users; Groups; Account Settings: Global Settings, Account Setup, Signature Preferences, Digital Signatures

Access Tokens

Application	Permissions	Date
Adobe Test Integration	View users in your account Create or manage users within your account Login access – providing full access to any user in your account overriding other requests Access documents & data on behalf of any user in your account Manage the status of documents on behalf of any user in your account Send documents on behalf of any user in your account View web forms on behalf of any user in your account Create, edit or publish web forms on behalf of any user in your account View templates and document library on behalf of any user in your account Manage the templates and document library on behalf of any user in your account View workflows on behalf of any user in your account Create workflows on behalf of any user in your account View webhooks on behalf of any user in your account Create or edit webhooks on behalf of any user in your account Permanently delete webhooks on behalf of any user in your account View applications. Manage applications, includes: managing oAuth scopes, application name, display name.	01/23/2023
Adobe Sign	View users in your account Create or manage users within your account Login access – providing full access to any user in your account overriding other requests Access documents & data on behalf of any user in your account Manage the status of documents on behalf of any user in your account	01/22/2023

Language: English: US

5. Select the Application that you have created.



Adobe Acrobat Sign — Upgrade — John

Home | Send | Manage | Reports | Account | API

Search: Access Tokens, Auto Delegation, My Notifications, Shared Notifications, My Signature, Language Preferences, View Other Accounts, Share My Account; Users; Groups; Account Settings: Global Settings, Account Setup, Signature Preferences, Digital Signatures

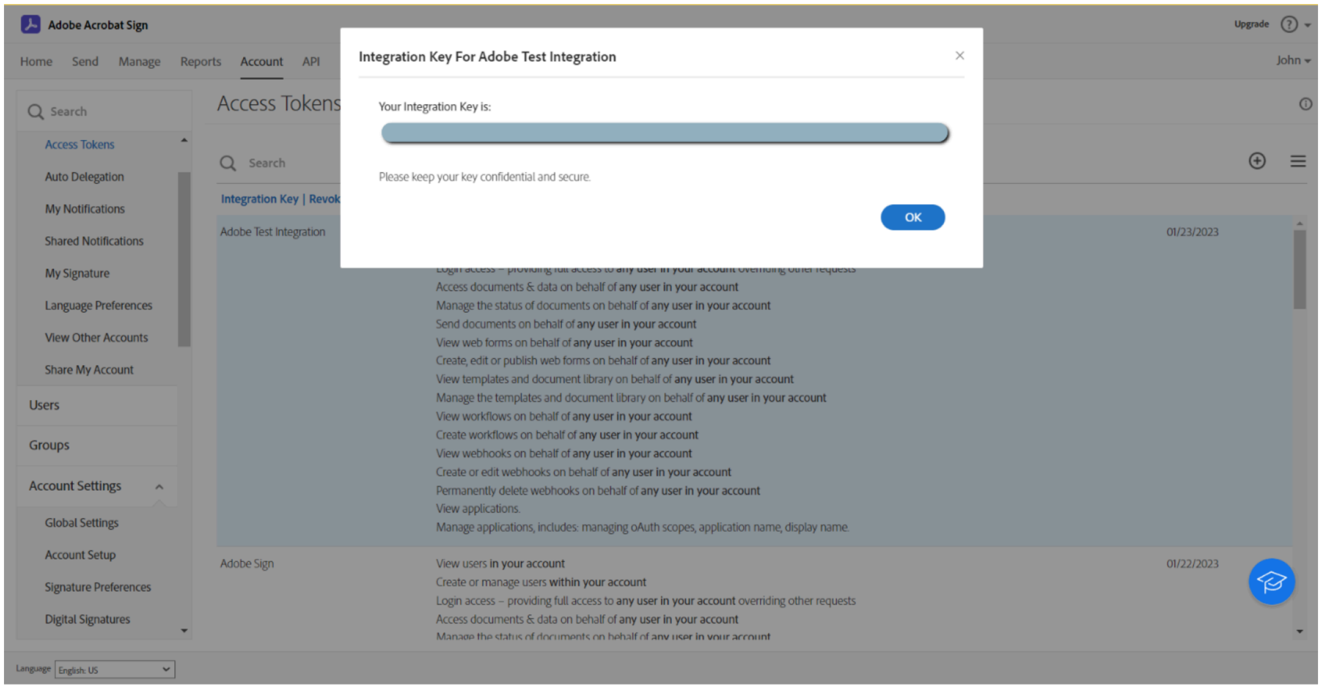
Access Tokens

Integration Key | Revoke

Adobe Test Integration	View users in your account Create or manage users within your account Login access – providing full access to any user in your account overriding other requests Access documents & data on behalf of any user in your account Manage the status of documents on behalf of any user in your account Send documents on behalf of any user in your account View web forms on behalf of any user in your account Create, edit or publish web forms on behalf of any user in your account View templates and document library on behalf of any user in your account Manage the templates and document library on behalf of any user in your account View workflows on behalf of any user in your account Create workflows on behalf of any user in your account View webhooks on behalf of any user in your account Create or edit webhooks on behalf of any user in your account Permanently delete webhooks on behalf of any user in your account View applications. Manage applications, includes: managing oAuth scopes, application name, display name.	01/23/2023
Adobe Sign	View users in your account Create or manage users within your account Login access – providing full access to any user in your account overriding other requests Access documents & data on behalf of any user in your account Manage the status of documents on behalf of any user in your account	01/22/2023

Language: English: US

6. You will see the option to get the Integration Key. Click on Integration Key and you can get the integration Key for your application.



Access Tokens

Integration Key For Adobe Test Integration

Your Integration Key is:

[redacted]

Please keep your key confidential and secure.

OK

Search

Integration Key | Revok

Adobe Test Integration — 01/23/2023

Login access – providing full access to any user in your account overriding other requests
Access documents & data on behalf of any user in your account
Manage the status of documents on behalf of any user in your account
Send documents on behalf of any user in your account
View web forms on behalf of any user in your account
Create, edit or publish web forms on behalf of any user in your account
View templates and document library on behalf of any user in your account
Manage the templates and document library on behalf of any user in your account
View workflows on behalf of any user in your account
Create workflows on behalf of any user in your account
View webhooks on behalf of any user in your account
Create or edit webhooks on behalf of any user in your account
Permanently delete webhooks on behalf of any user in your account
View applications.
Manage applications, includes: managing oAuth scopes, application name, display name.

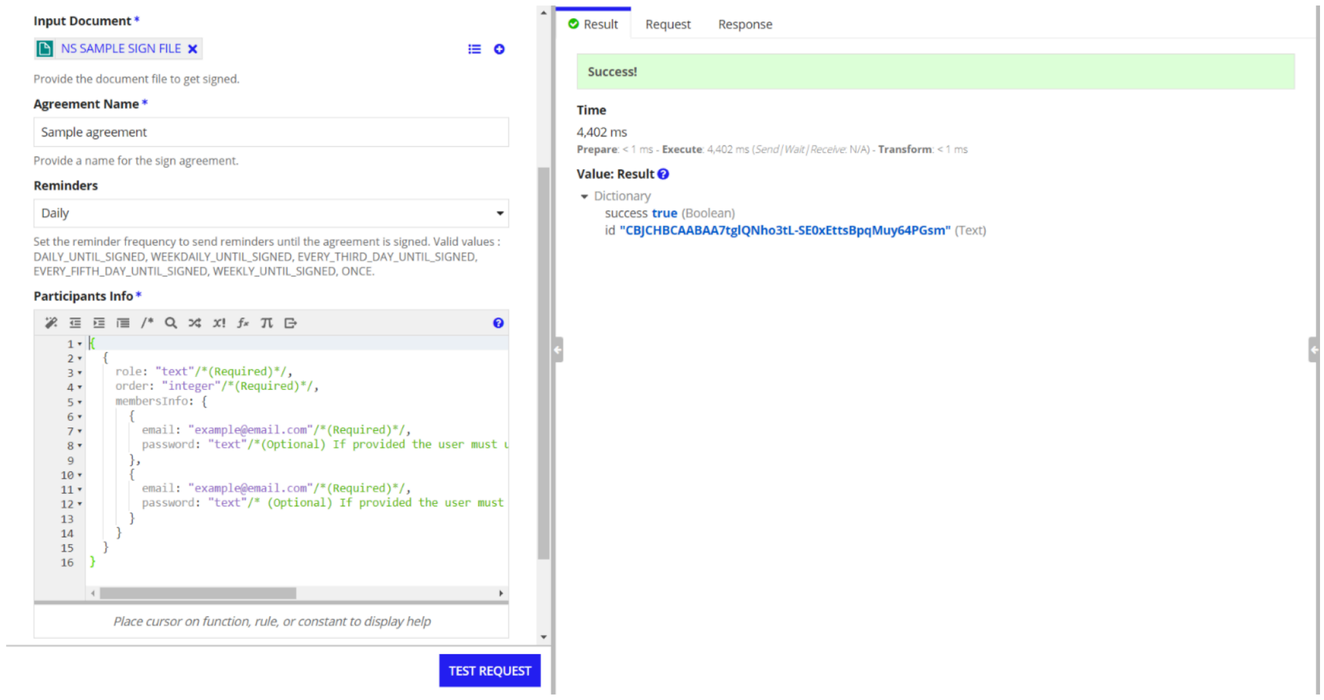
Adobe Sign — 01/22/2023

View users in your account
Create or manage users within your account
Login access – providing full access to any user in your account overriding other requests
Access documents & data on behalf of any user in your account
Manage the status of documents on behalf of any user in your account

7. Provide the Integration Key in the Connected System configuration.

Create Agreement

Creates an agreement. Sends it out for signatures, and returns the agreementID.



Input Document *

NS SAMPLE SIGN FILE

Provide the document file to get signed.

Agreement Name *

Sample agreement

Provide a name for the sign agreement.

Reminders

Daily

Set the reminder frequency to send reminders until the agreement is signed. Valid values : DAILY_UNTIL_SIGNED, WEEKDAILY_UNTIL_SIGNED, EVERY_THIRD_DAY_UNTIL_SIGNED, EVERY_FIFTH_DAY_UNTIL_SIGNED, WEEKLY_UNTIL_SIGNED, ONCE.

Participants Info *

```
{
  {
    role: "text"/*(Required)*/,
    order: "integer"/*(Required)*/,
    membersInfo: {
      {
        email: "example@email.com"/*(Required)*/,
        password: "text"/*(Optional) If provided the user must u
      },
      {
        email: "example@email.com"/*(Required)*/,
        password: "text"/* (Optional) If provided the user must
      }
    }
  }
}
```

Place cursor on function, rule, or constant to display help

Success!

Time

4,402 ms

Prepare: < 1 ms - **Execute**: 4,402 ms (*Send / Wait / Receive*: N/A) - **Transform**: < 1 ms

Value: Result

```
Dictionary
    success true (Boolean)
    id "CBJCHBCAABAA7tglQNho3tL-SE0xEttsBpqMuy64PGsm" (Text)
```

Inputs

Agreement Name (Text) - Required
Description: Provide a name for the Sign Agreement.

Reminders (Text) - Optional
Description: Set the reminder frequency to send reminders until the agreement is signed.
Default: null.
Valid values : DAILY_UNTIL_SIGNED, WEEKDAILY_UNTIL_SIGNED, EVERY_THIRD_DAY_UNTIL_SIGNED, EVERY_FIFTH_DAY_UNTIL_SIGNED, WEEKLY_UNTIL_SIGNED, ONCE.

Participants Info (List of Dictionary) - Required
Description: Provide the participants information for the agreement.

Role (Text) Required: Determines the role of the members in the participant set. Valid values : ‵SIGNER', 'APPROVER' , 'ACCEPTOR' , 'CERTIFIED_RECIPIENT' , 'FORM_FILLER' , 'DELEGATE_TO_SIGNER' , 'DELEGATE_TO_APPROVER' , 'DELEGATE_TO_ACCEPTOR' , 'DELEGATE_TO_CERTIFIED_RECIPIENT' , 'DELEGATE_TO_FORM_FILLER' .

Approver: Recipients marked as approvers review and approve the document but they are not required to sign it. They may be required to enter data into fields.

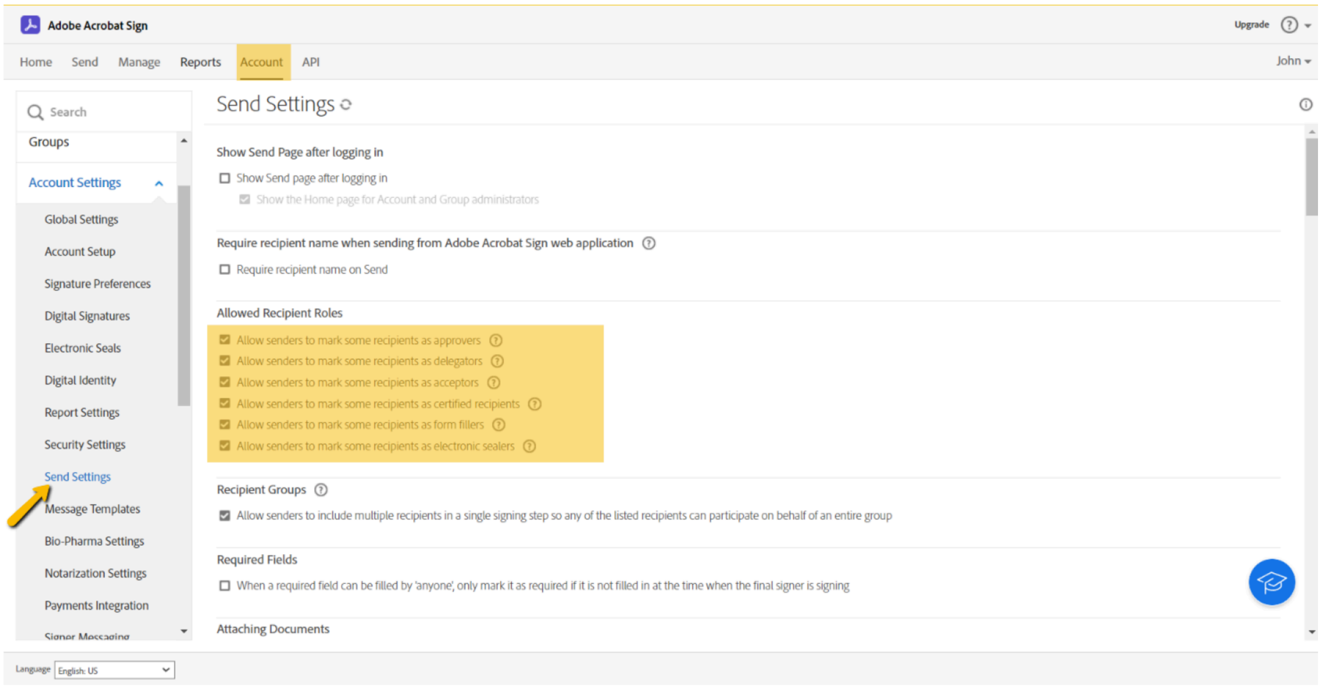
Delegator: Recipients marked as delegators may review the document but can't sign, approve or accept the document or acknowledge its receipt. They need to forward the document to another user who may take the appropriate action.

Acceptor: Recipients marked as acceptors are required to accept the document. They may be required to enter data into fields.

Certified Recipient: Recipients marked as certified recipients are required to view and acknowledge the receipt of the document.

Form Filler: Recipients marked as form fillers are required to enter data into the form fields and submit the document.

Please make sure to enable the roles in the Adobe Sign Account settings. By default, only SIGNER and APPROVER roles will be available.



Send Settings

Show Send Page after logging in

- [ ] Show Send page after logging in
  - [x] Show the Home page for Account and Group administrators

Require recipient name when sending from Adobe Acrobat Sign web application

- [ ] Require recipient name on Send

Allowed Recipient Roles

- [x] Allow senders to mark some recipients as approvers
- [x] Allow senders to mark some recipients as delegators
- [x] Allow senders to mark some recipients as acceptors
- [x] Allow senders to mark some recipients as certified recipients
- [x] Allow senders to mark some recipients as form fillers
- [x] Allow senders to mark some recipients as electronic sealers

Recipient Groups

- [x] Allow senders to include multiple recipients in a single signing step so any of the listed recipients can participate on behalf of an entire group

Required Fields

- [ ] When a required field can be filled by 'anyone', only mark it as required if it is not filled in at the time when the final signer is signing

Attaching Documents

Order (Integer) Required: Determines the order of signing for the participant set. Order is indexed from 1.

Members Info (List of Dictionary) Required : Each dictionary confirms to the following

```
Unset
{
email: (Text) - Required
password: (Text) - Optional - If a value is given, the member has to provide this password
before signing the document.
 }
```

Example:

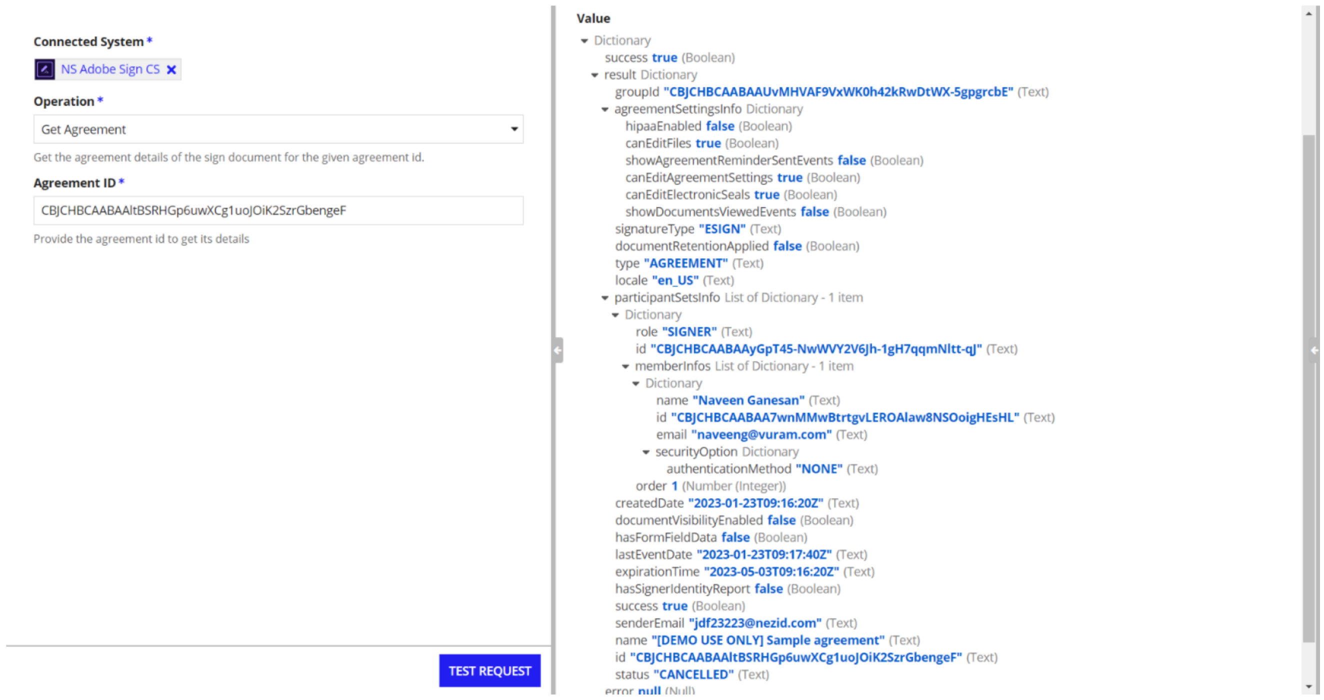
```
Unset
{
{
role: "text"/*(Required)*/,
order: "integer"/*(Required)*/,
membersInfo: {
{
email: "example@email.com"/*(Required)*/,
password: "text"/*(Optional) If provided the user must use this password to authenticate
before signing the document.*/ },
{
email: "example@email.com"/*(Required)*/,
password: "text"/* (Optional) If provided the user must use this password to
authenticate before signing the document.*/
}
}
}
}
```

Output: (Dictionary)

```
Unset
{
success: true,
id: agreementID
}
```

Get Agreement

Gets an agreement with its details in the form of Dictionary for the given AgreementID



Connected System *
NS Adobe Sign CS

Operation *
Get Agreement

Get the agreement details of the sign document for the given agreement id.

Agreement ID *
CBJCHBCAABAAltBSRHGp6uwXCg1uoJOiK2SzrGbengeF

Provide the agreement id to get its details

```
Value
▾ Dictionary
    success true (Boolean)
  ▾ result Dictionary
      groupId "CBJCHBCAABAAUvMHVAF9VxWK0h42kRwDtWX-5gpgrcbE" (Text)
    ▾ agreementSettingsInfo Dictionary
        hipaaEnabled false (Boolean)
        canEditFiles true (Boolean)
        showAgreementReminderSentEvents false (Boolean)
        canEditAgreementSettings true (Boolean)
        canEditElectronicSeals true (Boolean)
        showDocumentsViewedEvents false (Boolean)
      signatureType "ESIGN" (Text)
      documentRetentionApplied false (Boolean)
      type "AGREEMENT" (Text)
      locale "en_US" (Text)
    ▾ participantSetsInfo List of Dictionary - 1 item
      ▾ Dictionary
          role "SIGNER" (Text)
          id "CBJCHBCAABAAyGpT45-NwWVY2V6Jh-1gH7qqmNltt-qJ" (Text)
        ▾ memberInfos List of Dictionary - 1 item
          ▾ Dictionary
              name "Naveen Ganesan" (Text)
              id "CBJCHBCAABAA7wnMMwBtrtgvLEROAlaw8NSOoigHEsHL" (Text)
              email "naveeng@vuram.com" (Text)
            ▾ securityOption Dictionary
                authenticationMethod "NONE" (Text)
          order 1 (Number (Integer))
      createdDate "2023-01-23T09:16:20Z" (Text)
      documentVisibilityEnabled false (Boolean)
      hasFormFieldData false (Boolean)
      lastEventDate "2023-01-23T09:17:40Z" (Text)
      expirationTime "2023-05-03T09:16:20Z" (Text)
      hasSignerIdentityReport false (Boolean)
      success true (Boolean)
      senderEmail "jdf23223@nezid.com" (Text)
      name "[DEMO USE ONLY] Sample agreement" (Text)
      id "CBJCHBCAABAAltBSRHGp6uwXCg1uoJOiK2SzrGbengeF" (Text)
      status "CANCELLED" (Text)
    error null (Null)
```

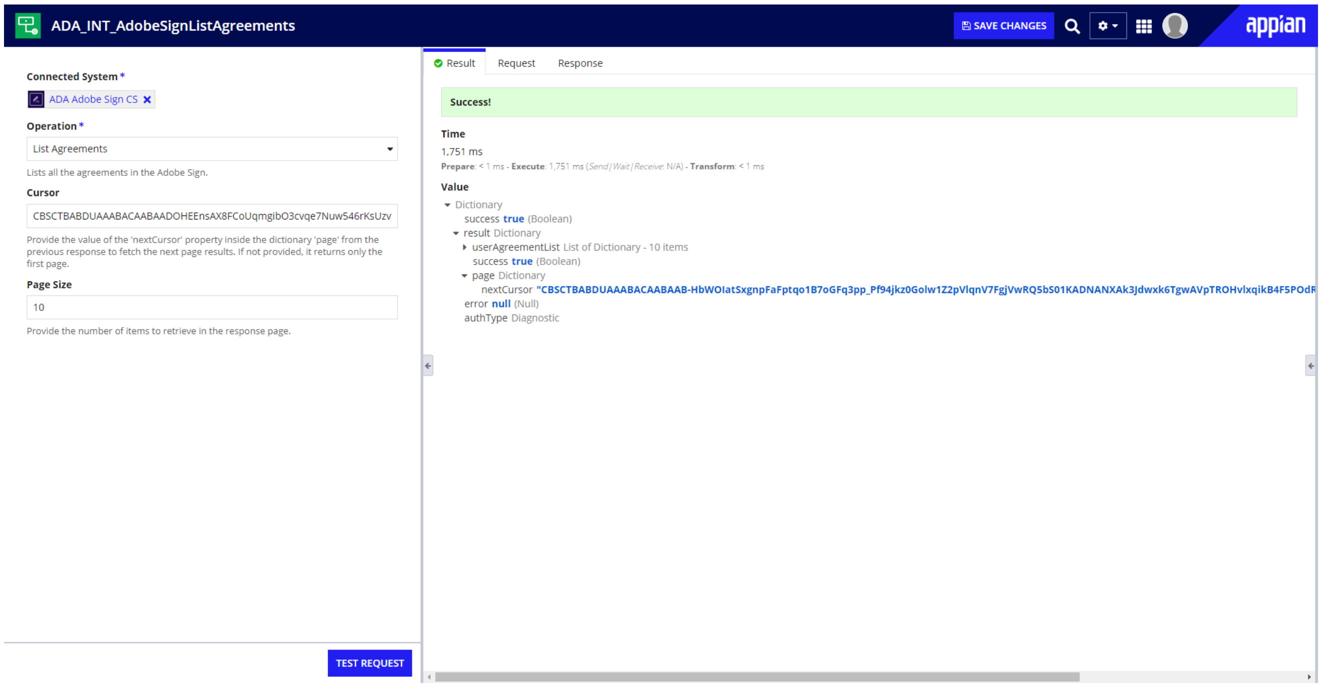
Input:
<u>Agreement ID:</u> (Text) Required
Provide the ID of the agreement to get its details.

Output: (Dictionary)

```
Unset
{
success: true,
result: Dictionary of agreement details.
}
```

List Agreements

Lists all the agreements along with its details.



ADA_INT_AdobeSignListAgreements

Connected System *

ADA Adobe Sign CS

Operation *

List Agreements

Lists all the agreements in the Adobe Sign.

Cursor

CBSCTBABDUAAABACAABAADOHEEnsAX8FCoUqmgibO3cvqe7Nuw546rKsUzv

Provide the value of the 'nextCursor' property inside the dictionary 'page' from the previous response to fetch the next page results. If not provided, it returns only the first page.

Page Size

10

Provide the number of items to retrieve in the response page.

Success!

Time

1,751 ms

Prepare: < 1 ms - **Execute**: 1,751 ms (*Send | Wait | Receive*: N/A) - **Transform**: < 1 ms

Value

```
▾ Dictionary
    success true (Boolean)
  ▾ result Dictionary
    ▸ userAgreementList List of Dictionary - 10 items
      success true (Boolean)
    ▾ page Dictionary
        nextCursor "CBSCTBABDUAAABACAABAAB-HbWOIatSxgnpFaFptqo1B7oGFq3pp_Pf94jkz0Golw1Z2pVlqnV7FgjVwRQ5bS01KADNANXAk3Jdwxk6TgwAVpTROHvlxqikB4F5POdR
    error null (Null)
    authType Diagnostic
```

Input:

<u>Cursor:</u> (Text) Optional

Provide the value of the 'nextCursor' property inside the dictionary 'page' from the previous response to fetch the next page results. If not provided, it returns only the first page.

<u>Page Size:</u> (Integer) Optional

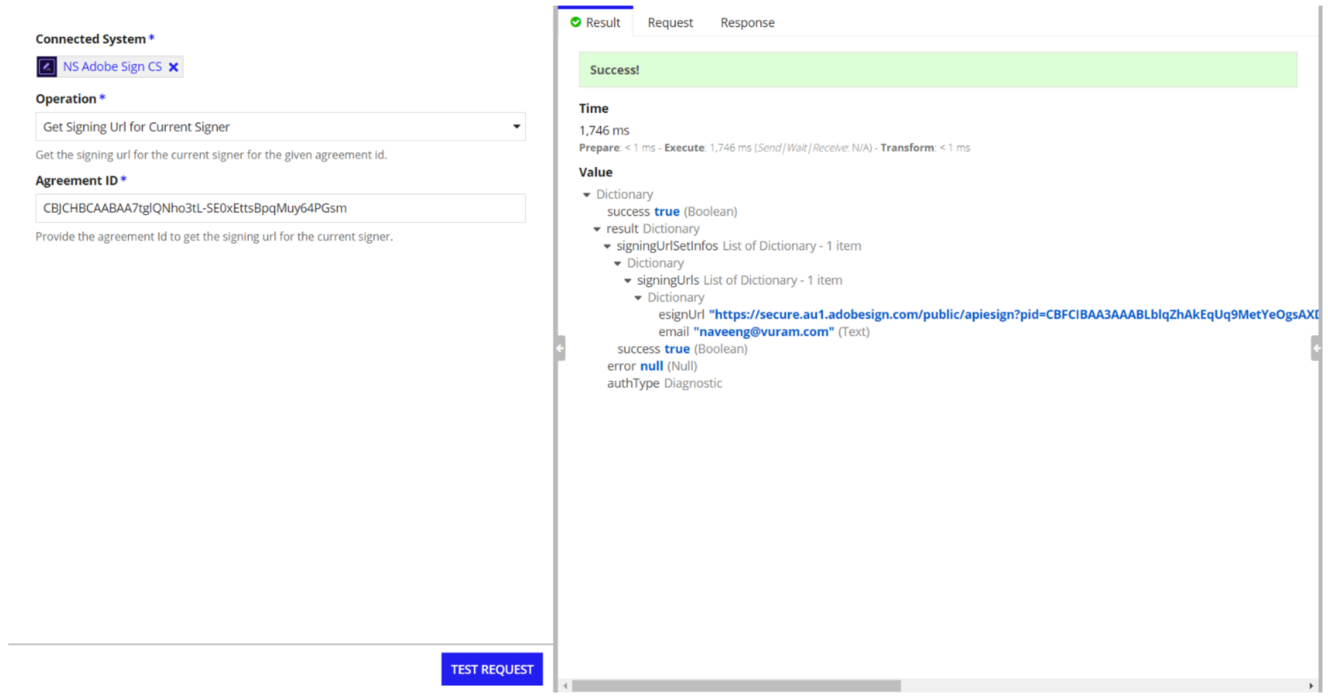
Provide the number of items to retrieve in the response page.

Output: (Dictionary)

```
Unset
{
success: true,
result: Dictionary of userAgreementList
}
```

Get Signing URL of Current Signer

Retrieves the URL for the e-sign page for the current signer(s) of an agreement.



Connected System *

NS Adobe Sign CS

Operation *

Get Signing Url for Current Signer

Get the signing url for the current signer for the given agreement id.

Agreement ID *

CBJCHBCAABAA7tglQNho3tL-SE0xEttsBpqMuy64PGsm

Provide the agreement Id to get the signing url for the current signer.

Success!

Time

1,746 ms

Prepare: < 1 ms - Execute: 1,746 ms (Send/Wait/Receive: N/A) - Transform: < 1 ms

Value

```
▾ Dictionary
    success true (Boolean)
  ▾ result Dictionary
    ▾ signingUrlSetInfos List of Dictionary - 1 item
      ▾ Dictionary
        ▾ signingUrls List of Dictionary - 1 item
          ▾ Dictionary
              esignUrl "https://secure.au1.adobesign.com/public/apiesign?pid=CBFCIBAA3AAABLblqZhAkEqUq9MetYeOgsAXD
              email "naveeng@vuram.com" (Text)
      success true (Boolean)
    error null (Null)
    authType Diagnostic
```

Input:

Agreement ID: (Text) Required

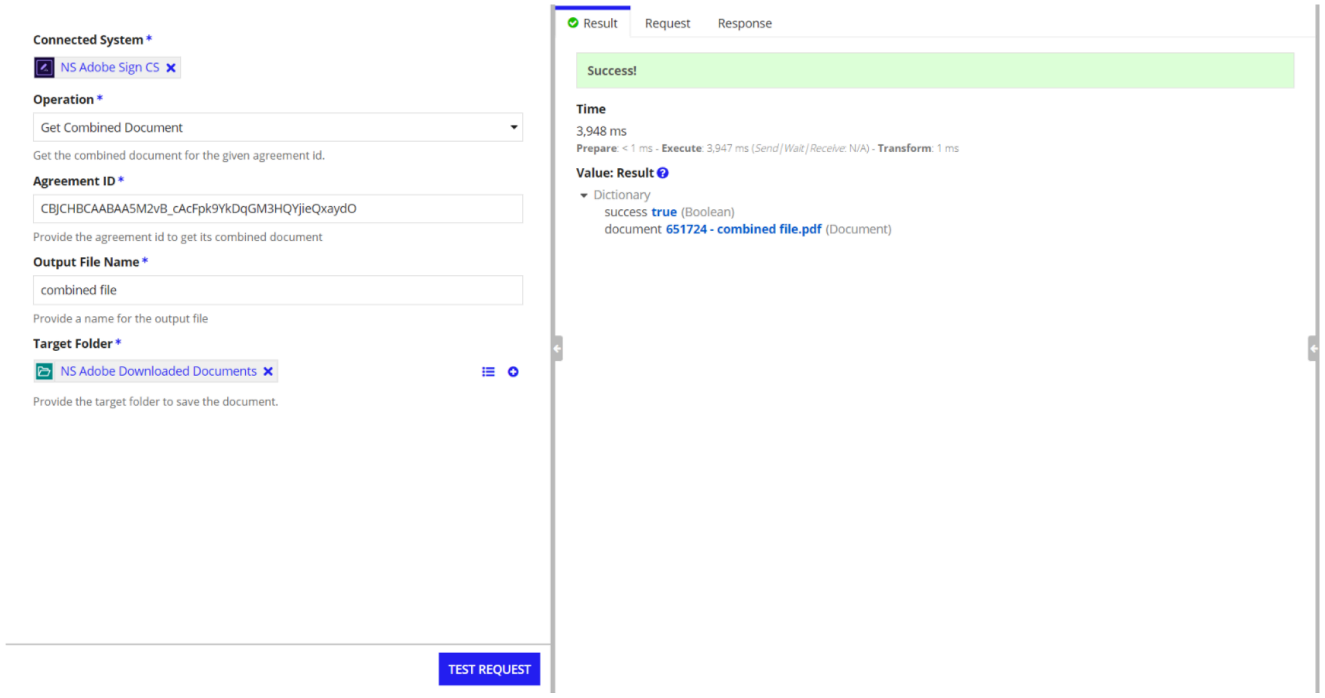
Provide the ID of the agreement to get the signing URL for the current signer(s).

Output: (Dictionary)

```
Unset
{
success : true,
result: Dictionary of signingURLSetInfos
}
```

Get Combined Document

Retrieves a single combined PDF document for the documents associated with an agreement and stores it in Appian in the given folder.



Connected System *
NS Adobe Sign CS
Operation *
Get Combined Document
Get the combined document for the given agreement id.
Agreement ID *
CBJCHBCAABAA5M2vB_cAcFpk9YkDqGM3HQYjieQxaydO
Provide the agreement id to get its combined document
Output File Name *
combined file
Provide a name for the output file
Target Folder *
NS Adobe Downloaded Documents
Provide the target folder to save the document.

Success!
Time
3,948 ms
Prepare: < 1 ms - Execute: 3,947 ms (Send/Wait/Receive: N/A) - Transform: 1 ms
Value: Result
Dictionary
success true (Boolean)
document 651724 - combined file.pdf (Document)

Input:

Agreement ID: (Text) Required

Provide the ID of the agreement to get its details.

Output File Name (Text) - Required

Description: Provide the name of the output file.

Target Folder(Folder) - Required

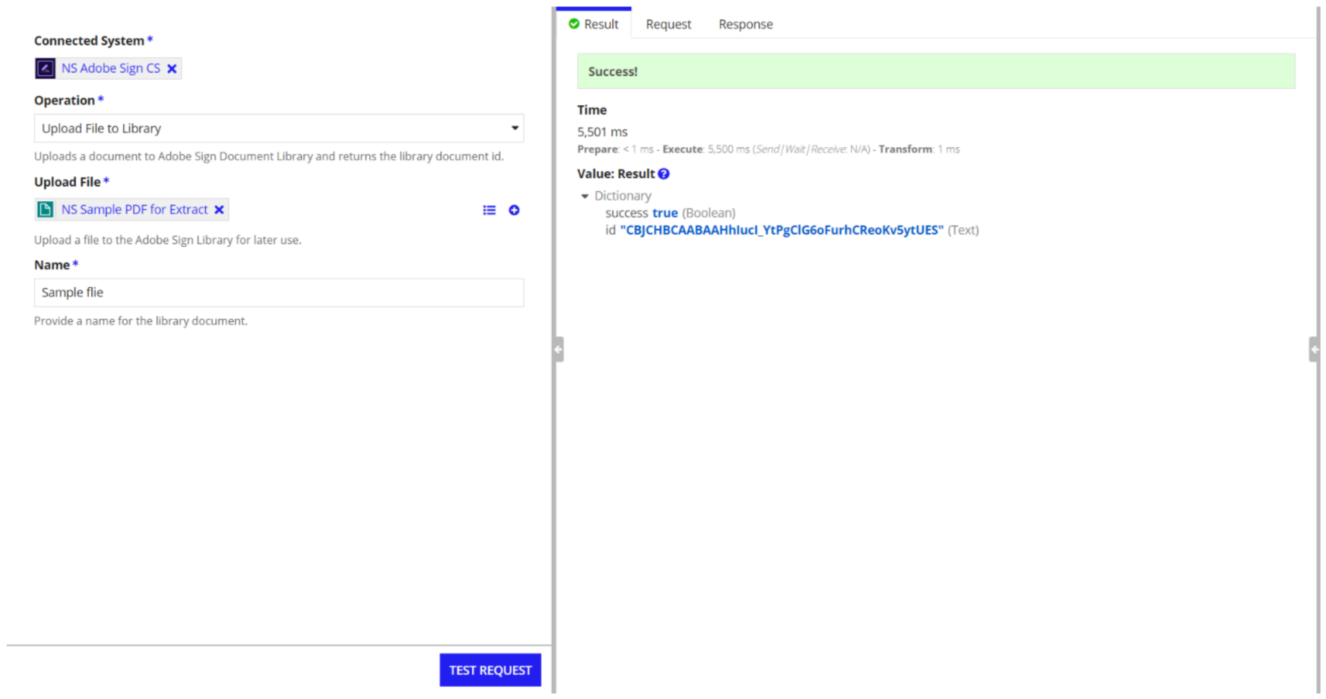
Description: Provide the target folder where the combined document needs to be saved.

Output - Dictionary

```
Unset
 {
 success: true,
 document: documentId - Output file name
 }
```

Upload File to Library

Uploads the file to Adobe Sign Library for later use.



Connected System *
NS Adobe Sign CS
Operation *
Upload File to Library
Uploads a document to Adobe Sign Document Library and returns the library document id.
Upload File *
NS Sample PDF for Extract
Upload a file to the Adobe Sign Library for later use.
Name *
Sample flie
Provide a name for the library document.

Success!
Time
5,501 ms
Prepare: < 1 ms - Execute: 5,500 ms (Send/Wait/Receive: N/A) - Transform: 1 ms
Value: Result

```
▾ Dictionary
    success true (Boolean)
    id "CBJCHBCAABAAHhlucl_YtPgClG6oFurhCReoKv5ytUES" (Text)
```

Inputs:
<u>Upload File</u> (Document) - Required
Description: Provide the document that needs to be uploaded to the Adobe Sign Library.

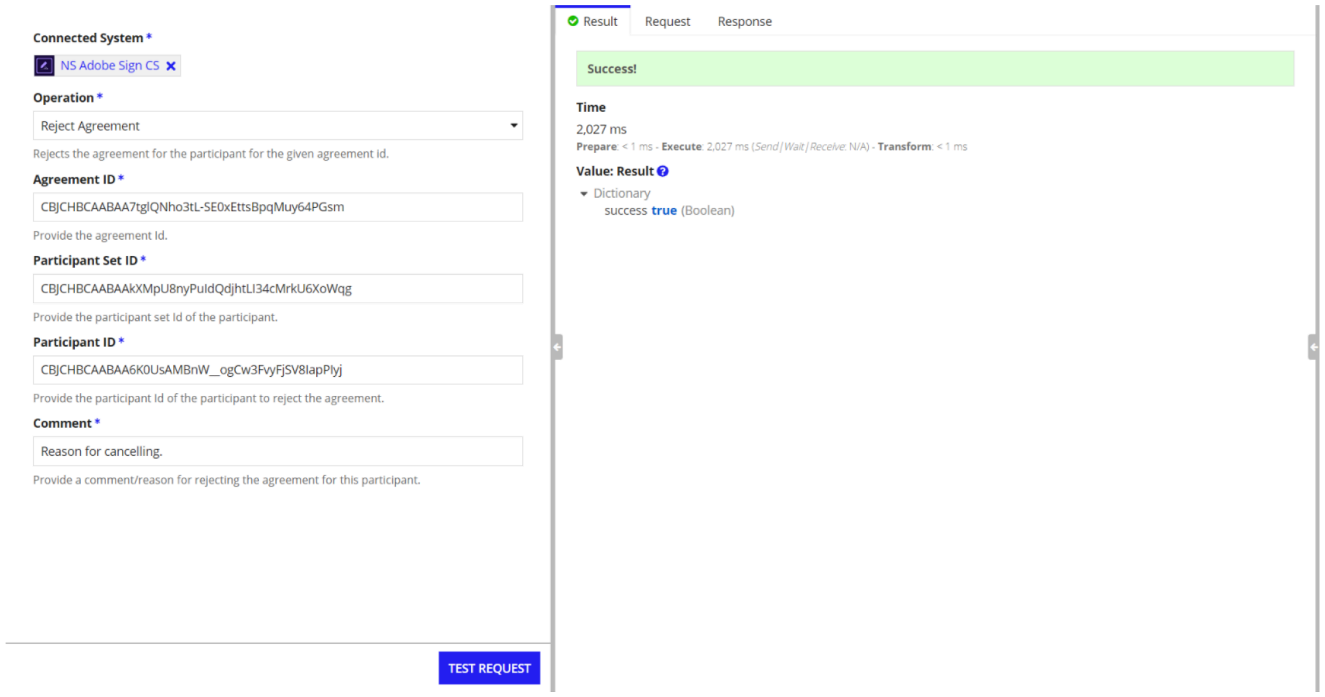
<u>Name</u> (Text) - Required
Description: Provide a name for the Library Document.

Output: (Dictionary)

```
Unset
{
success: true,
id: libraryDocumentID
}
```

Reject Agreement

Rejects the agreement for a participant.



Connected System *
NS Adobe Sign CS

Operation *
Reject Agreement
Rejects the agreement for the participant for the given agreement id.

Agreement ID *
CBJCHBCAABAA7tglQNho3tL-SE0xEttsBpqMuy64PGsm
Provide the agreement Id.

Participant Set ID *
CBJCHBCAABAAkXMpU8nyPuldQdjhtLI34cMrkU6XoWqg
Provide the participant set Id of the participant.

Participant ID *
CBJCHBCAABAA6K0UsAMBnW__ogCw3FvyFjSV8IapPlyj
Provide the participant Id of the participant to reject the agreement.

Comment *
Reason for cancelling.
Provide a comment/reason for rejecting the agreement for this participant.

Success!

Time
2,027 ms
Prepare: < 1 ms - Execute: 2,027 ms (*Send / Wait / Receive*: N/A) - Transform: < 1 ms

Value: Result
Dictionary
success true (Boolean)

Input:
Agreement ID: (Text) Required
Provide the ID of the agreement that needs to be rejected.

Participant Set ID (Text) - Required
Provide the ID of the member's participant set.
You can get the participant set ID from the response of Get Agreement.

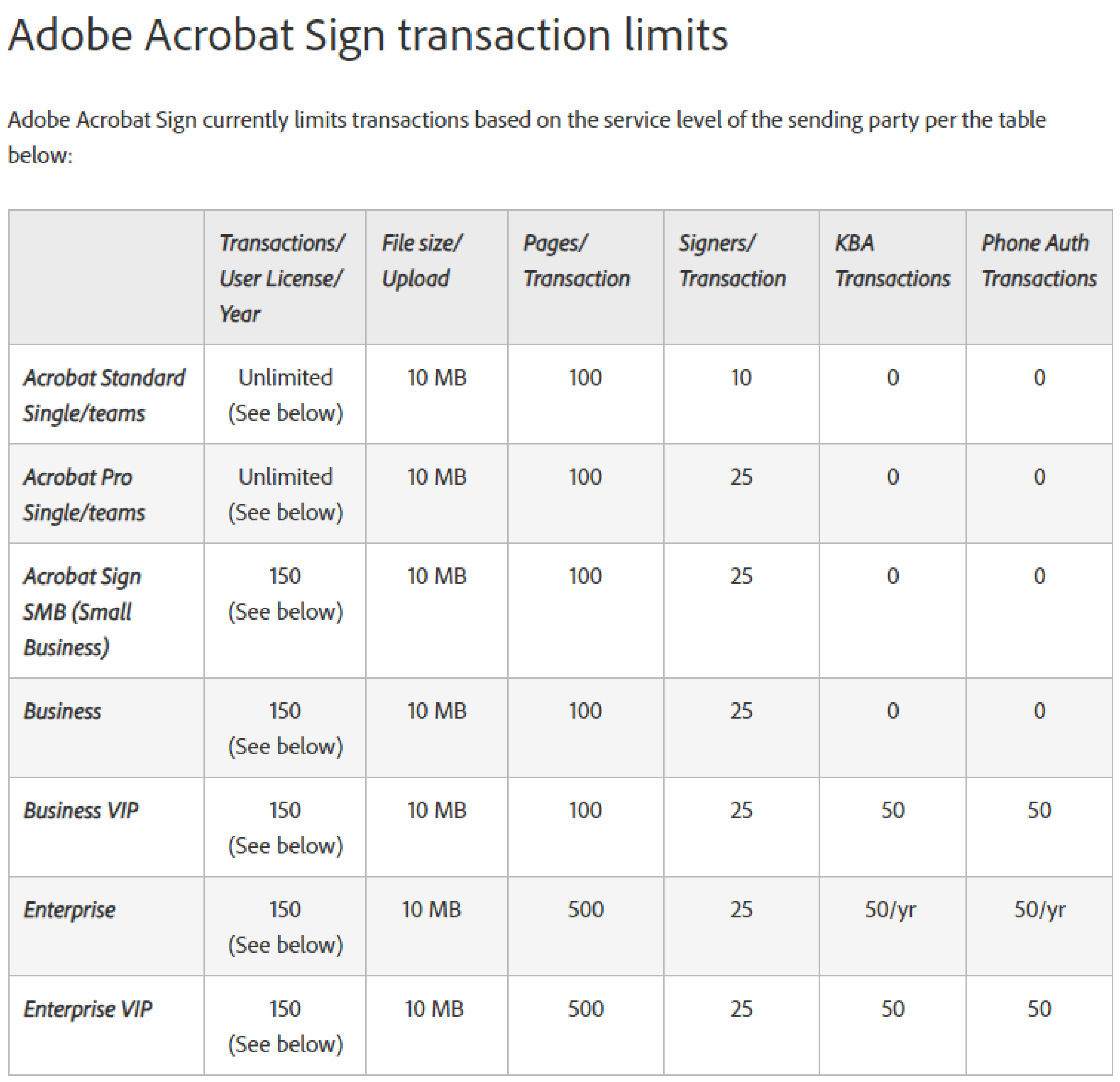
Participant ID (Text) - Required
Provide the ID of the participant.
You can get the participant ID from the response of Get Agreement.

Comment: (Text) - Required
Provide a comment/reason for rejecting the agreement for the participant.

Output: (Dictionary)

```
Unset
{ success: true }
```

Usage Limits



Adobe Acrobat Sign transaction limits

Adobe Acrobat Sign currently limits transactions based on the service level of the sending party per the table below:

	Transactions/ User License/ Year	*File size/ Upload*	*Pages/ Transaction*	*Signers/ Transaction*	*KBA Transactions*	*Phone Auth Transactions*
Acrobat Standard Single/teams	Unlimited (See below)	10 MB	100	10	0	0
Acrobat Pro Single/teams	Unlimited (See below)	10 MB	100	25	0	0
Acrobat Sign SMB (Small Business)	150 (See below)	10 MB	100	25	0	0
Business	150 (See below)	10 MB	100	25	0	0
Business VIP	150 (See below)	10 MB	100	25	50	50
Enterprise	150 (See below)	10 MB	500	25	50/yr	50/yr
Enterprise VIP	150 (See below)	10 MB	500	25	50	50

For more details please visit the Adobe Sign usage page here.