Foxit APIs

Overview

Foxit offers a comprehensive suite of REST APIs powered by Foxit's industry-leading document technology and scalable cloud infrastructure. These APIs give developers unmatched flexibility and control to design, automate, and optimize end-to-end document workflows across modern applications.

Foxit's API ecosystem includes **PDF Services APIs**, **PDF Embed APIs**, **Document Generation APIs**, and **eSign APIs**, each designed to address different stages of the document lifecycle from viewing and editing to creating, converting, signing, and managing documents at scale

---

Foxit PDF Services APIs

Foxit PDF Services APIs provides a set of REST APIs that allow you to perform powerful PDF operations, such as conversion, editing, merging, and more by simply making HTTP requests. These APIs are platform-independent and can be integrated into any application with ease.

Available Features

📄 Format Conversions

- Convert to/from:
    - PDF ⇄ Word
    - PDF ⇄ Excel
    - PDF ⇄ PowerPoint
    - PDF ⇄ HTML
    - PDF ⇄ Text
    - PDF ⇄ RTF
    - PDF ⇄ Images

🛠️ PDF Operations

- Merge multiple PDFs
- Split PDFs into parts
- Compare document versions
- Compress file size
- Add or remove password protection
- Reorganize pages
- Flatten form fields and annotations
- Optimize for web with linearization

Read more about PDF APIs

Foxit Document Generation APIs

The Document Generation APIs are built to simplify and automate the creation, formatting, and manipulation of documents in a scalable and reliable manner. Designed as a standalone solution, they are ideal for organizations that need to dynamically generate documents from templates, merge data into PDFs, or automate workflows across various departments.

These APIs can be seamlessly integrated into applications or enterprise platforms, enabling:

- On-demand generation of standardized documents
- Data-driven document population from structured sources (e.g., JSON)
- Output in multiple formats such as PDF and DOCX.

By leveraging these APIs, businesses can reduce manual effort, ensure document consistency, and accelerate time-to-delivery for customer-facing or internal documents.

⭐ Key Features

- **Document Template Management** 📝⚙️
  - Use predefined template documents with tags for dynamic data insertion.
  - Analyze a template document for text tags

identification.
    - Embed text tags directly in document templates to automate content placement.
- **Input Data** 📄➕
    - Include your template document as a Base64 encoded file string.
    - Text tags are replaced with actual values from input data. It maps values directly to text tags within your template document.
- **Document Generation** 🔧📄
    - Combines structured data and template documents to produce polished, ready-to-use documents.
    - Choose your desired output format, such as PDF or DOCX.

Read more about Document Generation APIs

---

eSign APIs

Foxit eSign is the #1 easiest and most collaborative contract management and eSignature solution, offering a wide range of robust features for seamless agreement workflows.

✅ Key Features

- Collaborative contract and agreement building
- Template-based contract creation
- eSignature workflows supporting single or multiple users
- Advanced bulk eSignature with dashboard and notification controls
- API integration for seamless connectivity with other software or websites
- Developer tools to embed Foxit eSign in custom applications or workflows

---

🛠️ Ways to Build with eSign

Foxit eSign provides flexible integration options to suit your development environment:

1. 🔗 REST API

- Allows data exchange over HTTPS
- Easily integrates with web, server-side, or native applications
- Commonly used for:
  - Generating documents server-side
  - Automating eSignature workflows

2. 💻 SDKs (Software Development Kits)

- Available for .NET, Java, PHP, Python, Ruby, and TypeScript
- Wraps API calls for frequently used development scenarios
- Simplifies embedding Foxit eSign in custom web apps

3. 🌐 Embed Foxit eSign

- Use an to embed Foxit eSign templates in your web views
- Customize and complete document signing directly within your application
- Ideal for allowing users to:
  - Review
  - Fill
  - Sign
  - Save document copies seamlessly

4. 🔌 Platform Integrations

- Native integration with popular CRM, financial, helpdesk, and productivity tools
- Lets you access Foxit eSign features directly from your existing platforms

---

🔔 Webhooks – Real-Time System

Updates

Keep your internal systems in sync with automatic updates via Webhooks.

- Webhooks notify your system about document events (e.g., signer completed, document executed)
- Foxit eSign sends an HTTP POST with a JSON payload containing:
  - `event_name`
  - `event_date`
  - `data` (event-specific details)

Read more about eSign APIs

Getting Started

To get started with the Document Generation APIs and PDF Services APIs, first create a Foxit Developer account. Once registered, you will receive the API credentials required to begin integration.

Follow the steps below to quickly set up your environment and start using the APIs:
👉 **Start here**: Foxit Developer Portal

Important: Foxit eSign APIs use a separate account and dashboard from other Foxit API services.
To integrate the eSign APIs, you must create an account in the Foxit eSign Portal to obtain your eSign API credentials.

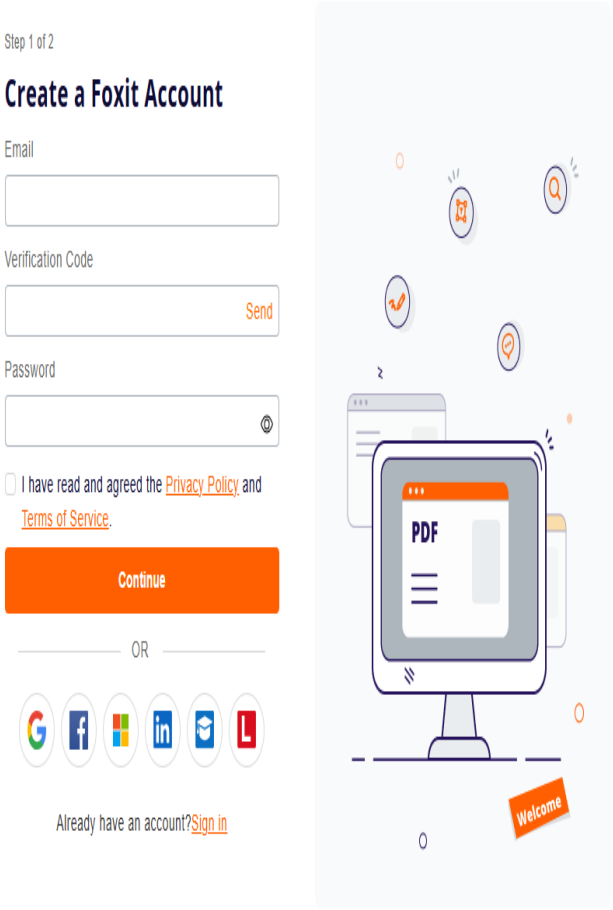
👉 **Foxit eSign Login**: Foxit eSign Portal

Step 1: Create Account

- Visit the Foxit Developer Portal.
- Click **Register** and fill in the required details.
- Verify your email address to activate your account.
- After login, you'll be taken to the Foxit APIs Dashboard.

> The account created provides access only to the

> PDF Services APIs and Document Generation APIs.

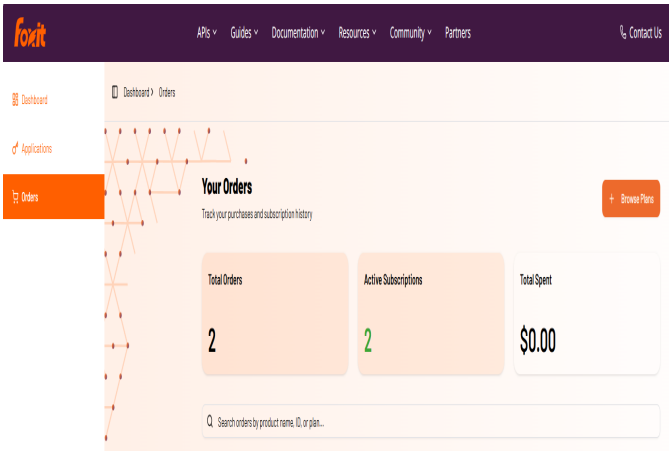


Foxit Developer Portal

Step 2: Request API Access Credentials

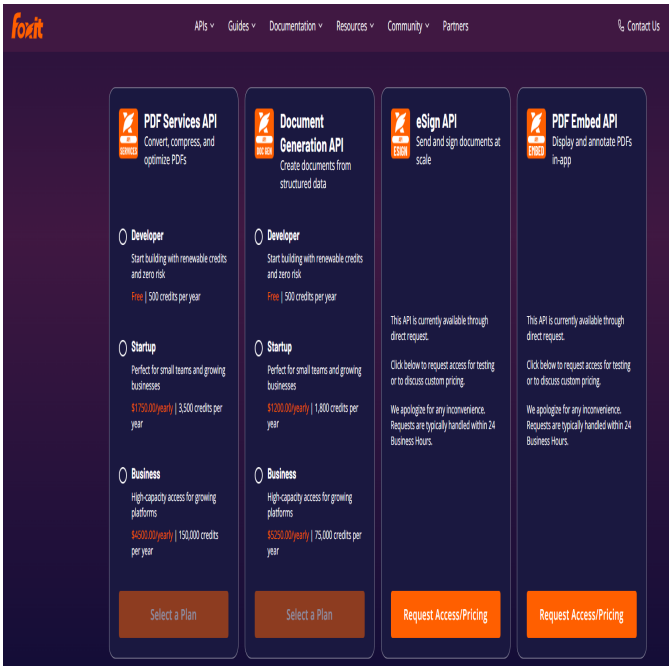
Once logged in:

- Navigate to the Foxit APIs Dashboard.
- On the left panel, click `Orders`.
- It will show all the plans along for their respective set of APIs.



Orders Section

- Choose the APIs you want to use:
  - PDF Services API
  - Document Generation API
  - PDF Embed API - Contact Sales
  - eSign API - Contact Sales



API Services

- Select your plan type:
  - 365-Day API Free Trial Access – Start testing instantly.
  - Request API Access – Submit a request for production use by clicking on **"Select a Plan"**.

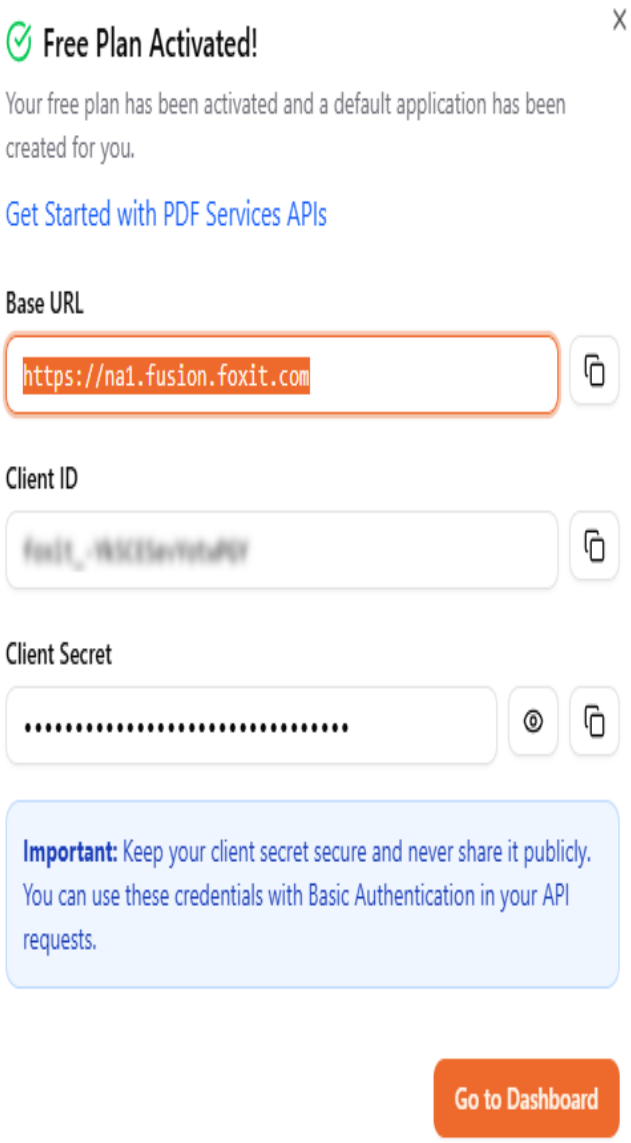
> 💡 Choosing Request API Access will prompt the Foxit team to review your application. Approval

> typically takes 24 hours.

Step 3: Get Your API Sandbox Credentials

If you selected the Free Trial Access option:

- Your temporary Client ID and Client Secret will be generated instantly after project creation.
- These API credentials are valid only for the PDF Services APIs and Document Generation APIs. To access the eSign APIs, you must **create an account on the** Foxit eSign Portal and coordinate with the Sales team to obtain the required credentials.
- Copy your credentials and store them securely.
- Use the credentials in your integration to begin making API calls.

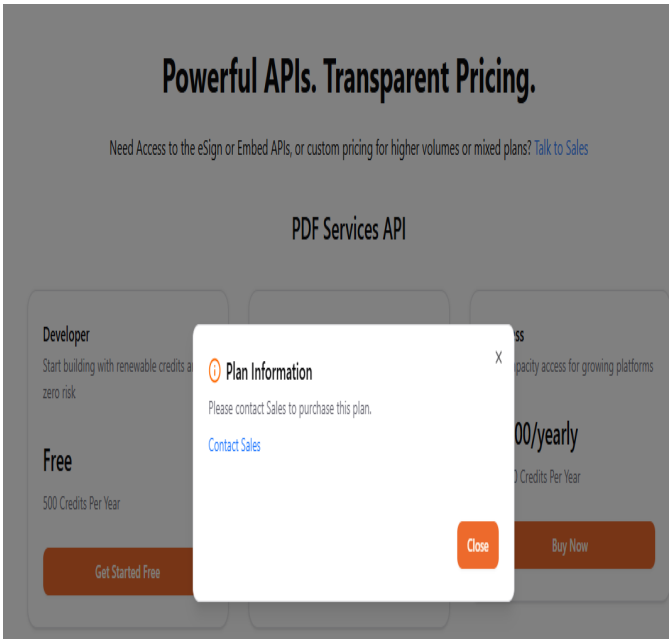


> 🔒 Note: Trial credentials are valid for 365 days and grant access to all supported API features.

Step 4: Complete Purchase (For Production Access)

If you selected Buy Now for API Access:

- You will be prompted to **Contact Sales**.



Contact Sales

- Click on **"contact sales"** to send the email. Provide your purchase details.
- The Foxit team will review your request and grant access within **24 hours**.
- You will receive an email confirmation along with your **API credentials**.

API Environments and Base URLs

Our APIs use environment-specific URLs for seamless integration across PDF, Document Generation, and eSign services.
Ensure you use the correct server and authentication credentials for reliable and secure access.

PDF APIs

US Server

Use this URL for US region:

Server	Base URI
default	`https://na1.fusion.foxit.com`

Document Generation APIs

US Server

Use this URL for US region:

Server	Base URI
default	`https://na1.fusion.foxit.com`

eSign APIs

US Server

Use this URL for US region:

Server	Base URI
default	`https://na1.foxitesign.foxit.com/api`

EU Server

Use this URL for the European region:

Server	Base URI
default	`https://eu1.foxitesign.foxit.com/api`

AU Server

Use this URL for the Australia region:

Server	Base URI
default	`https://au1.foxitesign.foxit.com/api`

CA Server

Use this URL for the Canada region:

Server	Base URI
default	`https://na2.foxitesign.foxit.com/api`

🔐 Authorization - PDF Services APIs

Foxit APIs use header-based authentication. To authorize your API requests, you'll need to obtain a Client ID and Client Secret.

> To access the PDF Services and Document Generation APIs, create an account in the Developer Portal.
> To access the eSign API, create a separate account in the eSign Portal.

PDF APIs

Foxit PDF Services APIs use Basic Authentication via custom headers. To authenticate your API requests, you must include your Client ID and Client Secret in the request headers.

Required Headers:

- `client_id` : Your unique Client ID
- `client_secret` : Your associated Client Secret

> ⚠️ Important: Keep your credentials secure. Never expose them in public repositories or client-side code.

Header Example

curl

```
curl {BASEURI}/{endpoint} \
--header "client_id: YOUR_CLIENT_ID" \
--header "client_secret: YOUR_CLIENT_SECR
```

> Replace YOUR_CLIENT_ID and YOUR_CLIENT_SECRET with your actual

> credentials. Replace {endpoint} with the specific API endpoint you're calling.

PDF Services APIs

Foxit PDF APIs offer a comprehensive suite of tools designed to streamline document processing, conversion, and management. Whether you're uploading source files, manipulating PDFs, or tracking document tasks, these APIs provide robust and flexible capabilities for developers.

API Categories

📄 Document Upload

- Upload source files for processing
- Supports multiple input formats
- Returns a unique `documentId` for subsequent operations

📥 Document Download

- Retrieve processed documents
- Supports file streaming with appropriate headers
- Optionally specify custom filenames

🛠️ PDF Management

- **Security**: Apply or remove password protection
- **Analysis**: Compare documents side-by-side
- **Modification**: Split, extract pages, flatten forms
- **Enhancement**: Add watermark, merge documents
- **Optimization**: Compress files, linearize for web

📄 PDF Creation

- Convert the following formats to PDF:
  - Microsoft Office files
  - Images (PNG, JPEG, etc.)
  - HTML pages
  - Text and RTF documents

🔄 PDF Conversion

- Convert PDFs into other formats:
  - Word (.docx), Excel (.xlsx), PowerPoint (.pptx)
  - Images (JPEG, PNG)
  - HTML, Plain Text, RTF

📊 Task Status

- Track the progress of operations
- View real-time completion percentages
- Access final results or handle any errors gracefully

🚀Foxit PDF Services - Quick Start Guide

Getting started with Foxit PDF APIs is fast and simple. Whether you're working on a web or backend platform, you can integrate document workflows in just a few steps.

Follow the steps below:

🔐 API Keys for Authorization

Before making any API calls, you'll need your developer credentials. Follow these steps:

- **Create Your Account**
  Sign up on the Foxit API Developer Portal. This gives you access to API tools and dashboard, see detailed instructions at Create Account.
- **Get API Trial Credentials**
  After account setup, retrieve your **Client ID** and **Client Secret** used for authenticating API requests. See detailed instructions at Get Foxit API Trial Credentials.

⚙️ Document Operation Workflow

1. 📤 Uploading a Document

- Use the `POST /pdf-`

`services/api/documents/upload` endpoint to upload your source file.

- Submit your file using `multipart/form-data`.
- Include your Client ID and Client Secret in the request headers for authentication.
- Receive a unique `documentId` that will be used for all subsequent API calls.

Sample cURL Request

```bash
curl --location 'https://na1.fusion.foxit
--header 'Content-Type: multipart/form-da
--header 'client_id: YOUR_CLIENT_ID' \
```

- **Supported formats:** PDF, Microsoft Office files, images, and plain text. extensions include: `.jfif`, `.pjpeg`, `.jpeg`, `.jpe`, `.pjp`, `.jpg`, `.jpx`, `.bmp`, `.dib`, `.png`, `.gif`, `.tiff`, `.tif`, `.pdf`, `.doc`, `.docx`, `.xls`, `.xlsx`, `.ppt`, `.pptx`, `.txt`, `.html`, `.htm`, `.shtml`, `.zip`, `.gd`, `.rtf`, `.dot`, `.dotx`, `.docm`, `.dotm`, `.wpd`, `.xlt`, `.xltx`, `.xlsm`, `.xlsb`, `.xltm`, `.csv`, `.pot`, `.potx`, `.pptm`, `.ppsx`, `.ppsm`, `.potm`, `.vsd`, `.vsdx`, `.wps`, `.hwp`, `.ofd`

2. 🛠️ Perform Operation

- Create, Convert, Flatten, Split or Compress with their respective operation endpoints. Please check out API reference for endpoint details.
- Submit a request with `documentId` and operation-specific parameters.
- All operations are **asynchronous**.
- Receive a `taskId` to monitor the operation's progress.

3. ✅ Fetch Operation Status

- Use the `GET /pdf-services/api/tasks/:task-id` endpoint with the provided `taskId` to track the status

of your operation.

- Monitor the current status, which could be one of the following:
  PENDING, PROCESSING, COMPLETED, or FAILED.
- Progress is tracked in real-time as a percentage (0–100%).
- Once the task is successfully completed, a `resultDocumentId` will be returned, which you'll use to download the final file.

Sample cURL Request

```bash
curl --location 'https://na1.fusion.foxit
--header 'client_id: YOUR_CLIENT_ID' \
```

Sample Response

```json
{
    "taskId": "6838c9cxxb1cc44abxxx40c5",
    "status": "COMPLETED",
    "progress": 100,
    "resultDocumentId": "683xx9cf6b1ccxxxb
}
```

4. 📩 Download Result

- Use the `GET /api/documents/:document-id/download` endpoint with the `resultDocumentId` to retrieve the final processed document.
- Download supports content streaming with appropriate content-type headers.
- Optionally specify a custom filename.

```bash
curl --location 'https://na1.fusion.foxit
```

⚠️ Error Handling

The Foxit PDF API implements standard HTTP status codes to indicate the outcome of API requests. In the event of an error, a structured response is returned, making it easier for developers to diagnose and resolve issues.

Status Code	Meaning	Description
`400`	Bad Request	The request parameters are missing or malformed.
`401`	Unauthorized	The API key is missing or invalid.
`404`	Not Found	The requested resource could not be located.
`413`	Payload Too Large	The file exceeds the maximum allowed size.
`500`	Internal Server Error	An unexpected error

📄 Error Response Format

Each error response includes the following fields:

- **Error Code**: Identifies the type of error.
- **Message**: A human-readable explanation of what went wrong.
- **Details (optional)**: Additional technical information or context to assist with debugging.

✅ Best Practices

Asynchronous Operations:

- Remember that all operations are asynchronous.
- Always confirm task status before attempting to download results.
- Integrate thoughtful retry logic for reliable task completion.

File Size Limits

- The maximum file size supported for PDF Services APIs is **100 MB**.
- For Word to PDF conversion, the file size is limited to **50 MB**.

Support & Assistance

- **Foxit Support Center:** Your first stop for comprehensive help resources, tutorials, and FAQs. For more details check out Support Center.
- **API Status Updates:** Keep an eye on the health and performance of our APIs.
- **Direct Technical Support:** For personalized troubleshooting and issue resolution, connect with our technical experts at **support@foxitsoftware.com**.

Next Steps

To get the most out of your integration, here's what we recommend:

- **Dive into the API Reference:** For comprehensive details on every endpoint, request, and response, explore our full API documentation.
- **Try the Quick Start Example:** Get up and running fast by working through our quick start guide. It's the best way to see the API in action.
- **Contact Sales for Production Keys:** When you're ready to go live, reach out to our sales team to acquire your production API keys.
  Contact Sales

🔐 Authorization - Document Generation APIs

Document Generation APIs

Foxit Document Generation APIs use Basic Authentication via custom headers. To authenticate your API requests, you must include your Client ID and Client Secret in the request headers.

Required Headers:

- `client_id`: Your unique Client ID
- `client_secret`: Your associated Client Secret

> ⚠️ Important: Keep your credentials secure. Never expose them in public repositories or client-side code.

Header Example

curl

```
curl {BASEURI}/{endpoint} \
--header "client_id: YOUR_CLIENT_ID" \
--header "client_secret: YOUR_CLIENT_SECRE
```

> Replace YOUR_CLIENT_ID and YOUR_CLIENT_SECRET with your actual credentials. Replace {endpoint} with the specific API endpoint you're calling.

Document Generation APIs

The Document Generation API enables dynamic creation of professional, data-driven documents using pre-defined templates. By embedding smart text tags within your documents and supplying structured data, you can automate the generation of PDF or DOCX files with precision and efficiency.

To automate the creation of documents with dynamic data, Foxit offers two powerful APIs designed to work together seamlessly:

1. 🔍 **Analyze Document API**
   This API scans your uploaded template document to detect all embedded text tags (placeholders). It returns a detailed list of identified tags, enabling you to understand which variables need to be mapped and how to structure your input data.
2. 📄 **Generate Document API**
   This API takes your Word template (as a Base64-encoded string) and structured input data (JSON), and dynamically injects values into the corresponding text tags. The result is a fully formatted, ready-to-use document output in PDF or DOCX format.

These APIs streamline the entire document automation process from understanding your template structure to generating polished, data-populated documents.

📝 How to Generate Documents Dynamically

We offer RESTful APIs for document generation, enabling users to send requests and receive dynamic document outputs. These endpoints support key operations such as template creation, data submission, and document generation. All communications are secured, with built-in mechanisms to ensure authentication and authorization of each request.
Our APIs empower you to generate documents efficiently through the following key steps:

✔️ Prepare Word template with text tags

✔️ Prepare JSON with matching keys

✔️ Convert document to Base64

✔️ Make API call to generate final document

Let's dive into the step by step process

🧩 Step 1: Create Your Template

To generate a document, the first step is to create a template where values can be dynamically inserted. This template could be any document such as a contract, agreement, or form.

Add Text Tags on Document

Text tags act as placeholders that will be replaced with actual values when data is sent through the API. You can add text tags directly within your Word document using double curly braces, like `{{Foxit}}`.

The invoice template below showcases the use of dynamic elements like styled tables, repeating data, calculations, and conditional logic. These features help generate fully customized documents during document generation.

Word Template Screenshot

Reference Number	Description
1	A page header. Each

	page can have its own header or you can repeat headers across pages.
2	A reserved keyword, `today`, formatted in a specific data format. (reference here)
3	A simple string `AccountName` which will pre-populated with a string value using the Document Generation API
4	A Date Field

When merged, the placeholder merge fields are replaced with the data provided from the `documentValues` JSON object from Generate Document API Request. This merged template appears as below:

How to Add Text Tags

Text tags in the document must match the variable names in your JSON payload. Here's how you can use them with different data formats:

Simple Strings

You can directly insert simple string variables in your template using curly braces.
Example:

```
Plain Text

My name is {{name}}; I am {{age}} years o
```

JSON Example:

```
Plain Text

{
    "name":"Peter",
    "age":30
}
```

Output: `My name is Peter; I am 30 years old.`

Dates

You can insert the current date or format specific date fields using text tags in your document.
To insert today's date use the following text tag:

```
Plain Text

{{today \@ MM/dd/yyyy}}
```

> Note: You do not need to pass this value in the JSON. The tag will automatically insert the current system date in the specified format.

For Date Field formatting: `{{FIELD_NAME \\@ DATE_FORMAT}}` text tag may be applied.
To format a date from your JSON see the below example:

```
Plain Text

Payment Due Date is {{PaymentDueDate \@ MM
```

JSON Example:

```
Plain Text

{
    "PaymentDueDate": "06/02/2025"
}
```

Output: `Payment Due Date is 06/02/2025`

The following field codes are supported for Date Format:

Date Format Supported	Field Codes
Month/Day/Year	`MM/dd/yyyy`
Non-padded month/day	`M/d/yyyy`
Day/Month/Year (EU format)	`dd/MM/yyyy`
ISO 8601 format	`yyyy-MM-dd`
Full month name	`MMMM d, yyyy`
Abbreviated month	`MMM d, yyyy`
European text format	`d MMM yyyy`
Full date + month names	`dddd, MMMM d, yyyy`
Abbreviated weekday/month	`ddd, MMM d`

Tables

You can dynamically populate tables with rows and columns using text tags. To begin a table, use the `{{TableStart:field_name}}` tag. To close the table, add the `{{TableEnd:field_name}}` tag in the last column of the final row in your Word document.

Example usage in a Word document:

Sl	Qty	Total
`{{TableStart:OpportunityLineItems}}`	`{{OpportunityLineItems.quantity}}`	`{{OpportunityLineItems.totalprice}}` `{{TableEnd:OpportunityLineItems}}`

The tags like `{{OpportunityLineItems.quantity}}` and `{{OpportunityLineItems.totalprice}}` will be replaced with actual values from your JSON input during document generation. Each entry in the

`OpportunityLineItems` array will generate a new row in the table.

Output:

#	Qty	Unit Price	Total
1	1	3500.00	3500.00
2	2	3500.00	7000.00
3	1	2100.00	2100.00

Output of Table Values

Formula for SUM

The text tag `=SUM(ABOVE)` calculation fields are supported in tables. If a `=SUM(ABOVE)` field is added as its own row in a table, it will take the values of the last column in the table and dynamically calculate the total sum.

Example usage of SUM formula:

Sl	Qty	Total
`{{TableStart:OpportunityLineItems}}`	`{{OpportunityLineItems.quantity}}`	`{{OpportunityLineItems.totalprice}}` `{{TableEnd:OpportunityLineItems}}`
		Total Purchase Price: `{{=SUM(ABOVE)}}`

Output:

#	Qty	Unit Price	Total
1	1	3500.00	3500.00
2	2	3500.00	7000.00
3	1	2100.00	2100.00
			Total Purchase Price: $12,600.00

Output of SUM formula

eSignature Fields

An eSignature text tag for dynamically creating an eSignature field. The signature field will be automatically placed when uploaded into Foxit eSign.

Example of esignature field text tag:

For Signer 1

```
Plain Text

${signfield:1:y:____}
```

For Signer 2

```
Plain Text

${signfield:2:y:____}
```

Check out more examples from our eSign API documentation here(link)

Adding Conditional Blocks

To add **conditional logic** in your Word template, you can use mail merge fields to render specific content based on the data you provide.

Enable Field Code View

First, enable the visibility of field codes in your document:

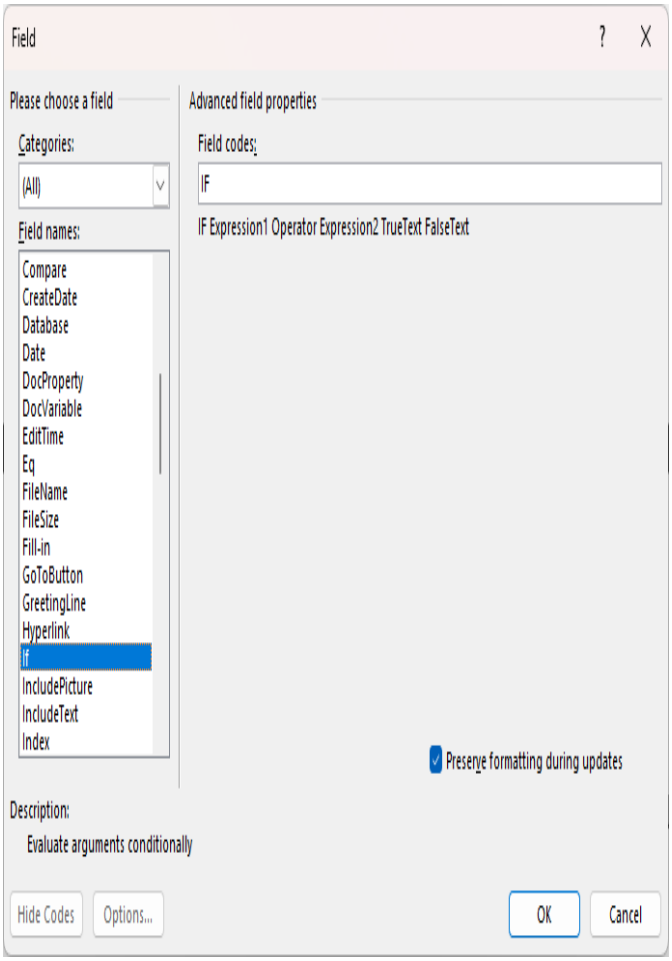
- Press `ALT + F9` to toggle field code view.

Insert a Conditional Field

Navigate to the part of the document where you want to add a condition. Then:

1. Go to the **Insert** tab.
2. Click on **Quick Parts**.
3. Select **Field**.

This opens the **Field Editor**. In the left-hand list, scroll down and select If:



This will insert an `IF` field into your document (make sure you're still viewing field codes). Right click and select toggle to view field:

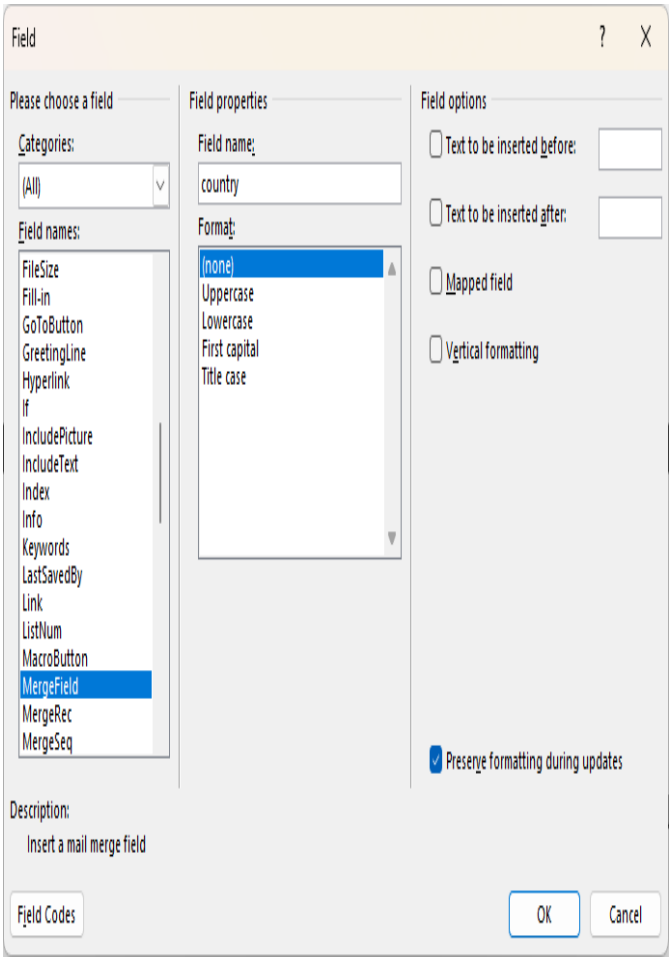
IF Field Syntax

```
Plain Text
```

You can also use other operations like `<`, `<=`, `>`, and `>=`.

Insert the Field You Want to Evaluate

1. Place your cursor after `IF` inside the brackets.
2. Go to **Insert > Quick Parts > Field**, and choose **MergeField**.
3. Enter the name of the field you want to check (e.g., a field from your JSON data).



Click OK, and it will be added to your IF condition.

```
{IF{MERGEFIELD country \* MERGEFORMAT}\* MERGEFORMAT}
```

Example:

```
Plain Text

{ IF "Country" }
```

Complete the IF Statement

You're almost there. The last step is to add the condition and define the output text for both the `true` and `false` cases. Technically both are optional and can be empty strings. Here we've added a check for the value of `country`:

```
{IF{MERGEFIELD country \* MERGEFORMAT}="USA" "Domestic Shipping" "Intl Shipping"\* MERGEFORMAT}
```

Example:

Plain Text

📦 Step 2: Map Data with JSON

The next step is to prepare the JSON data that will be used to inject values into your Word document template.

This JSON should be included in the request body of the Document Generation API (`/document-generation/api/GenerateDocumentBase64`) under the `documentValues` parameter as key-value pairs.

Each key in the JSON should match the corresponding text tag used in your template document.

Lets use example of real text tags and map their values in JSON.

- **Simple String** - `{{AccountName}}` , `{{Name}}`
- **Table** - `{{TableStart:OpportunityLineItems}}` , `{{OpportunityLineItems.quantity}}` , `{{OpportunityLineItems.unitprice \# Currency }}` ,

  `{{TableEnd:OpportunityLineItems}}`
- **Date** - `{{today \@ MM/dd/yyyy}}` , `{{PaymentDueDate \@ MM/dd/yyyy}}`
- **SUM formula** - `{{=SUM(ABOVE) \# Currency}}`
- **Conditional Formula** - `ShowAgreement`

JSON body request:

Plain Text

```
{
  "outputFormat": "pdf",
  "currencyCulture": "en-US",
  "documentValues": {
    "ShowAgreement": "true",
    "PaymentDueDate": "06/02/2025",
    "AccountName": "Jordan O'Connor",
    "Name": "Jordan"
```

🚀 Step 3: Upload and Generate Document

The final step is to **convert your Word document template to a Base64-encoded string** and include it in the request body of the **Document Generation API** `/document-generation/api/GenerateDocumentBase64`.

To specify the output format:

- Set the parameter **`outputFormat`** to `"DOCX"` to receive a **Word document** as output.
- Set it to `"PDF"` to receive a **PDF file** instead. (Default is PDF)

The API will inject the values from your JSON into the template and return the generated document in the specified format.

Account Name: Jordan O'Connor
Payment Due Date: 06/02/2025

#	Qty	Unit Price	Total
1	1	$3,500.00	$3,500.00
2	2	$3,500.00	$7,000.00
3	1	$2,100.00	$2,100.00
			Total Purchase Price: $12,600.00

Quote Accepted by:			
Printed Name:	Jordan	Date:	06/20/2025
Signature:	${signfield:1:y:___}		

🚀Document Generation - Quick Start Guide

Getting started with Foxit Document Generation APIs is designed to be **fast and simple**. Whether you're developing for a web or backend platform, you can seamlessly integrate our powerful document generation capabilities into your workflows in just a few steps.

Begin by following these steps:

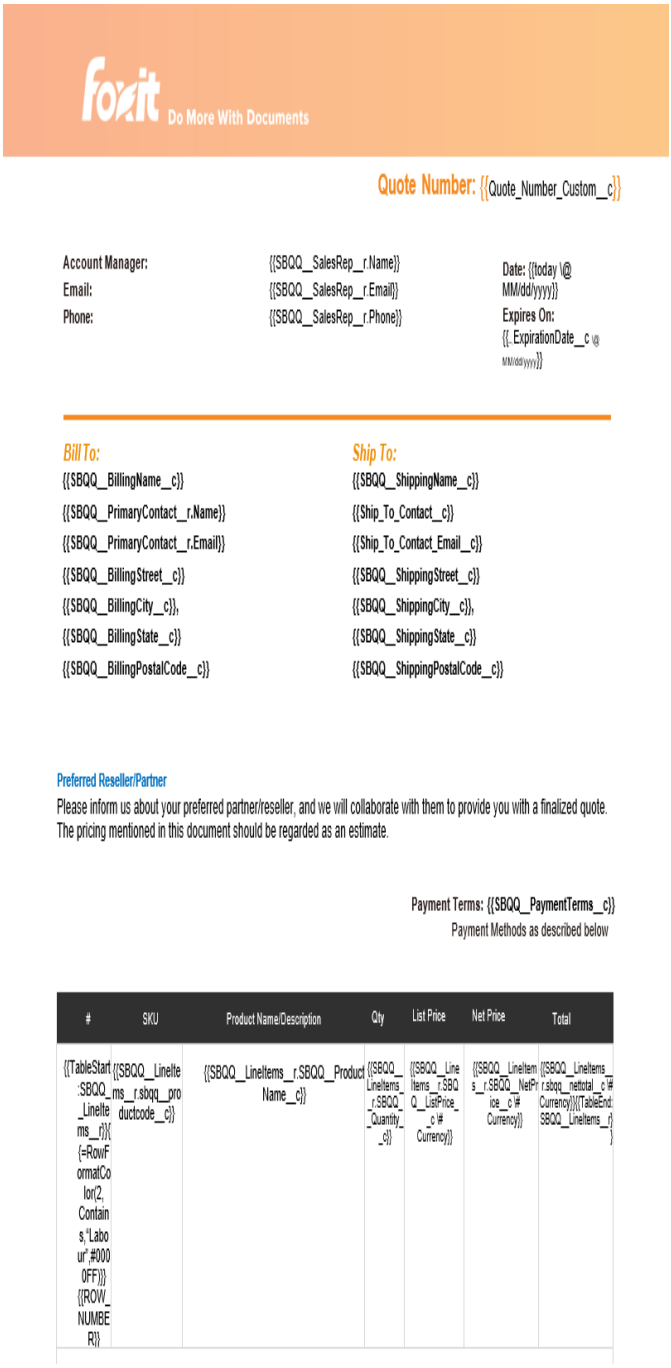
🔐 API Keys for Authorization

Before we proceed, make sure you have finished the account creation step to get your trial or production credentials.
If you haven't yet, you can do so by following the below:

- **Create Your Account**
  Sign up on the Foxit Cloud API Developer Portal. This gives you access to API tools and dashboard, see detailed instructions at Create Account.
- **Get API Trial Credentials**
  After account setup, retrieve your **Client ID** and **Client Secret** used for authenticating API requests. See detailed instructions at Get Foxit Cloud API Trial Credentials.

🔄 Add Text Tags on Document

For this example, we'll be using a DOCX file. You can create it using any preferred online or offline document editor. Once your DOCX is ready, insert the required text tags into the file. Refer to the screenshots and sample tags below for guidance.



🛠️ Prepare Documents with Text Tags

Text Tags allow you to dynamically populate documents by linking placeholder text with values from your JSON input data using the Document Generation API.

A **Text Tag** is a plain text placeholder embedded in your document. It is wrapped with double curly braces ({{ }}), where the content inside represents a variable or field name from your JSON data.

When a document is generated, these tags are automatically replaced with actual values.
Example Tags:

- `Account Name: {{Account.Name}}`
- `Payment Due Date: {{CloseDate}}`

📊 Analyze the Documents for Text Tags

Do you have existing documents embedded with text tags and need to programmatically identify them? The **Analyze Document API** is your solution.

This API allows you to extract all defined text tags from your template documents. This capability is crucial for programmatically mapping field values, ensuring you have all the necessary variables to construct your JSON payload when generating new documents using the Document Generation API.

How it Works

Simply send your document (e.g., as a Base64 string) to the Analyze Document API. It will parse the document and return a structured list of all detected tags, categorized for easy use.

Example cURL Request:

Plain Text

```
curl --location 'https://na1.fusion.foxit
--header 'client_id: 7bxxx0xx0dxx4xxdbxx5a
--header 'client_secret: 9a8xxx4d0xx0xx2xx
--header 'Content-Type: application/json'
--data '{
```

Example of Text Tags Analysis:

Plain Text

```
{
    "singleTagsString": "AccountName Payme
```

In this example:

- `singleTagsString` lists all individual text tags found. These correspond to fields that expect a single value.
- `doubleTagsString` identifies tags that typically represent tables or repeating sections (e.g., `OpportunityineItems` suggests a list of line items).

This output provides a ready-to-use list of all dynamic fields within your template, streamlining your integration process.

Generate Documents

This is the final step in the process: generating your documents by dynamically injecting data using a JSON input. This allows you to programmatically create fully populated documents from your templates.

The JSON payload you send will contain variables that directly match the text tags within your template document. The values you provide for each variable will then be mapped onto the document, enabling comprehensive content generation.

Example cURL Request:

Use the following cURL request to generate a document. Remember to replace `"xx5exx5xxfdxxbxx9xxxxxba8xx98xx"` with your actual Bearer Token and `"YOUR_DOCUMENT_BASE64_STRING"` with your Base64 encoded template file string.

Plain Text

```
curl --location 'https://na1.fusion.foxit
--header 'client_id: 7bxxx0xx0dxx4xxdbxx5a
--header 'client_secret: 9a8xxx4d0xx0xx2xx
--header 'Content-Type: application/json'
--data '{
  "outputFormat": "pdf",
  "currencyCulture": "en-US",
  "documentValues": {
```

This process allows you to effectively automate the creation of documents by programmatically populating their content.

⚠️ Error Handling

The Foxit Document Generation API implements standard HTTP status codes to indicate the outcome of API requests. In the event of an error, a structured response is returned, making it easier for developers to diagnose and resolve issues.

Status Code	Meaning	Description
`400`	Bad Request	The request parameters are missing or malformed.
`401`	Unauthorized	The API key is missing or invalid.
`404`	Not Found	The requested resource could not be located.
`413`	Payload Too Large	The file exceeds the maximum allowed size.
`500`	Internal Server Error	An unexpected error

🗒️ Error Response Format

Each error response includes the following fields:

- **Error Code**: Identifies the type of error.
- **Message**: A human-readable explanation of what went wrong.
- **Details (optional)**: Additional technical information or context to assist with debugging.

File Size Limits

The initial template file must be less than **4 MB** in size to ensure optimal processing and performance.

Next Steps

To get the most out of your integration, here's what we recommend:

- **Dive into the API Reference:** For comprehensive details on every endpoint, request, and response, explore our full API documentation.
- **Try the Quick Start Example:** Get up and running fast by working through our Quick Start guide. It's the best way to see the API in action.
- **Contact Sales for Production Keys:** When you're ready to go live, reach out to our sales team to acquire your production API keys.
  Contact Sales

🔐 Authorization - Foxit eSign APIs

eSign APIs

Foxit eSign APIs have OAuth2.0 based authentication. Generate an access token using your `client_id` and `client_secret` to access eSign APIs.

The client_id and client_secret for eSign APIs will be available on eSign Portal.
👉 **Foxit eSign Login**: Foxit eSign Portal

Generate your access token using below cURL request

Plain Text

```
curl --location '{BASEURI}/api/oauth2/acce
--header 'Content-Type: application/x-www-
--data-urlencode 'grant_type=client_creden
--data-urlencode 'client_id=YOUR_CLIENT_ID
--data-urlencode 'client_secret=YOUR_CLIEN
--data-urlencode 'scope=read-write'
```

> Replace YOUR_CLIENT_ID and YOUR_CLIENT_SECRET with your actual credentials. Replace {endpoint} with the specific API endpoint you're calling.

The bearer token is sent in the API request like this:

Plain Text

```
curl {BASEURI} -H 'Authorization: Bearer -
```

eSign APIs

Foxit eSign is the #1 easiest and most collaborative contract management and eSignature software with the most robust functionality amongst its peers. This page will help you configure or install the Foxit eSign APIs and SDKs.

Prepare Documents

📧 Creating an Envelope from PDF Files

With Foxit eSign API you can create a folder of documents and send them for signature right from your own application.

To create a folder of documents from PDF files you can either provide publicly accessible URLs to PDF documents or pass PDF documents as multipart form data and other parameters about the recipient parties, etc.

While creating the documents from PDF files you can define various fields like text field or signature field within the document using simple Text Tags. You can assign various properties for these fields like party responsible for filling the data etc. These Text Tags will then be converted to Foxit eSign document fields.

Or if you don't want to use Text Fields within the document, then you can also request to create an embedded document preparing session where you can manually drag and drop each field on the document itself without leaving your application. PDF documents with text tags.

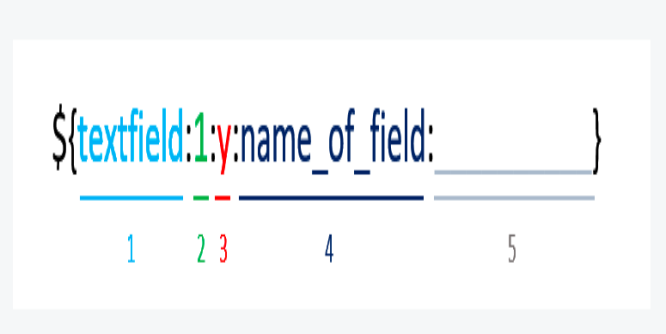
🔄 Preparing PDF documents with text tags

Here is a PDF file with Text Tag samples: Sample Document.

To send a PDF document from your application for signature, you need to define the signature and other Foxit eSign fields in the documents, and who is allowed to fill in the data. Foxit eSign supports simple Text Tags.

A Text Tag is a simple text block within your PDF document which defines the field type, the party who is responsible for it, and other field specific options.

Quick Tip: When you apply Text Tags to the document, Foxit eSign does not remove or replace the Text Tags text. You can hide codified Text Tags by using the same font color as the background of the document. So the Text Tags can be used by Foxit eSign processes and it will not be visible on the document.



1. Field Type: One of 'textfield', 'formulafield', 'signfield', 'initialfield', 'datefield', 'checkboxfield', 'attachmentfield', 'imagefield', 'accept', 'decline' or short notations.
2. Party Number (optional): the sequence number of the recipient party from the parties list which will be responsible for filling this field.
3. Required/Optional Field: 'y' makes the field mandatory to be filled before signing while 'n' makes it optional.
4. Field Name: Optional. Assign a name to this field. Assigning names to fields is a good practice for better productivity as named fields are downloaded in the document form data report and all same named fields in a document will automatically take up the value typed once in any one of those fields. Please note not to include any space character for the name of field (or anywhere else in the Text Tag) as it will then not recognize that word as a proper Text Tag syntax, instead you can use underscore which will then be replaced with space character in the final Foxit eSign field.
5. Underscores: Optional. A way to increase the field's width.

There are other options that you can add before underscores for various other properties as follows:

Sizing

By default, the new field has the same width and height as the original text tag, however it can be overridden by adding custom width as 90 and height as 20 (in pixels):

`${textfield:1:y:field_name:90:20}`

Validation

You also can validation information for text fields inside the tags. For example, the max number of characters required are 12 and only Numbers:

`${textfield:1:y:field_name:90:20:12:Numbers}`

And if you want to stick to default height and width for the new field (which is the height and width of the text tag)

`${textfield:1:y:field_name:::12:Numbers}`

NOTE: validation inputs are currently only available for textfields

Font Style

You can also control the font style for each new field via their text tags. For example, the following tag creates a textfield with 14 font size and gray as font color

`${textfield:1:y:field_name:90:20:12:Numbers:14:gray}`

You can pass pre-filled value to the textfields via its text tags as shown below. Please make to replace any space character with '_' (underscore) else the system may not recognize that text tag.

`${textfield:1:y:field_name:90:20:12:Numbers:14:gray:default_value}`

NOTE: pre-filled values are currently only available for textfields

Mark as Dependent Field

You can make a field dependent on any other field value. To set a field dependent on another field, select the values of the independent field that will enable the signer to see the dependent field.

`${t:1:y:field_name:90:20:12:Numbers:14:gray:default_value:parent_field_name:value_of_parent_field:options}`

Column	Description
parent_field_name	Name of the parent field. NOTE: Only the following type of fields are allowed as the parent or independent field: `textfields`, `textbox`, `checkbox`, `radiobutton`, `dropdown`
value_of_parent_field	Value of the parent field. NOTE: In the case of radio button and checkbox, the value can be either checked or unchecked
options (optional)	In the case of

The full list of Field Types is:

TextField - `textfield` or `t` (short notation)
TextBox - `textboxfield` or `tb` (short notation)
Signature - `signfield` or `s`
FormulaField - `formulafield` or `ff`
Initial - `initialfield` or `i`
Date - `datefield` or `d`
Checkbox - `checkboxfield` or `c` (recommended)
Radio Button - `radiobuttonfield` or `rb` (recommended)
Secured Field - `securedfield` or `sc`
Attachment Field - `attachmentfield` or `a`
Image Field - `imagefield` or `img`

Signer Name Field - `textfield` or `t`
Date Signed - `datefield` or `d`
Accept Button - `accept` or `ab`
Decline Button - `decline` or `db`
PayField - `payfield` or `pf`
Examples

- `${textfield:1:y:client_name:________}` - A mandatory textfield which is assigned to party number 1 with field name as 'client name'.
- `${tb:1:n:_________________}` - An optional textbox assinged to party 1.
- `${signfield:1:y:____}` - A mandatory signature field assigned to party number 1.
- `${i:2:n}` - An optional initial field assigned to party number 2.
- `${datefield:2:n::____}` - An optional date field assigned to party number 2 with empty field name.
- `${c:2:y:male:gender}` - A checkbox assigned to party number 2 with initial value as checked with name 'male' and group 'gender'.
- `${c:1:y:yes:group-y}` - A checkbox assigned to party number 1 with initial value as checked with name 'yes' and group 'group'. 'group-y' means group mandatory.
- `${rb:1:n:yes:grp1}` - An un-selected radio button assigned to party 1 with name yes and grp 'grp1'.
- `${rb:1:y:no:group2-y}` - A radio button assigned to party number 1 with initial value as checked with name 'no' and group 'group2'. 'group2-y' means group mandatory.
- `${sc:2:n:Credit_Card_Number:4:____}` - A secured field assigned to party 2 which is optional with name 'Credit Card Number' and only last 4 characters to remain unmasked.
- `${attachmentfield:1:y:____}` - A mandatory attachment field assigned to party number 1.
- `${img:1:n:stamp_image:120:50:__}` - An optional image field is field name as 'stamp image' with 120-pixel width and 50-pixels height which is assigned to party number 1.
- `${textfield:1:y:Signer_Name:________}` - A mandatory signer name field which is

assigned to party number 1 with field name as 'Signer_Name'. This field is auto-populated once the document is opened by the signer.

- `${datefield:1:y:Date_Signed:_______}` - A mandatory date signed field assigned to party number 1 with field name as 'Date_Signed'. This field will auto populate once the document is opened by the signer.
- `${accept:1:90:20}` - A accept button field with 90-pixel width and 20-pixel height assigned to party number 1.
- `${decline:1:90:20}` - A decline button field with 90-pixel width and 20-pixel height assigned to party number 1.
- `${payfield:1:paymentType:payeeOptions:productAndService:paymentDescription:paymentAmount}` - Tag consolidates Payment Type, Product/Service, Payment Description, and Payment Amount to utilize the payment field.

Personalized Fields:

We can use the personalized field tag while sending it via API in place of process tags. Personalized field creation instructions can be seen in the help center.

Examples:

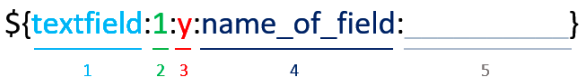
- `${field_7:1}` - Using this format, you can simply use the existing personalized field name and properties assigned to party #1.
- `${field_15:1:y:field_name:90:20}` - Using this format, you can simply use the existing personalized field name.

👬 Preparing PDF documents with Recipient Party Tags

To send a PDF document from your application for signature, you can either define the recipient parties in the documents itself or send the recipients information via API call. Foxit eSign supports simple Text Tags for recipient party information. In case both API call and Tags on the PDF are provided with Recipient Party information, Foxit eSign will use API call values.

A **Recipient Party Tag** is a simple text block within your PDF document that defines the party, who is responsible for filling the fields.

Quick Tip: When you apply Recipient Party tags to the document, Foxit eSign does not remove or replace the Recipient Party Tags text. You can hide codified Recipient Party Tags by using the same font color as the background of the document, so the Recipient Party tags will be used by Foxit eSign to derive signers. Still, the tags will not be visible on the document.



alt text

1. **Tag Type:** Accepts only value as rp, which is an identifier of recipient party tags.
2. **Party First Name:** First name of the recipient party.
3. **Party Last Name:** Last name of the recipient party.
4. **Party email:** Email of the recipient party.
5. **Party permission:** Use this field to assign folder permissions to a recipient. Can be any one of the following values: FILL_FIELDS_AND_SIGN, FILL_FIELDS_ONLY, SIGN_ONLY, VIEW_ONLY
6. **Party workflow sequence:** If the document is signed in sequence, the recipient party will receive the notification as per the workflow sequence number. It should be starting with 1 like 1,2,3,4, etc.
7. **Party sequence:** Assign a sequence number to a recipient in the list of recipient parties. Use unique sequence numbers or party numbers for each party, starting with 1 like 1,2,3,4, etc.

💿 Adding Fields in Documents via API

Fields can also be added on the documents via an array of JSON objects in the input data for the API request. This way of adding fields gives you more control over the size of fields, their font color and font

size, etc.

While adding the fields on each document page, the position of the fields is relative to the top left corner of that page.

Each field object must contain the following values:

1. type - (string) the type of field to be added at this place. Can be one of these values: text|signature|initial|textbox|date|secure|checkbox|radiobutton|dropdown|attachment|image|accept|decline.
2. x - (int) the x location coordinate of the top left corner of the field in pixels
3. y - (int) the y location coordinate of the top left corner of the field in pixels
4. width - (int) the width of field in pixel
5. height - (int) the height of field in pixel
6. documentNumber - (int) the document index in the document files array starting from 1
7. pageNumber - (int) page in the document where field needs to be added

There are other parameters which can be optional and different for different field types.

`text` or `textbox` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- value - (string)
- required - (boolean) value: either true or false
- textAlignment - (string) value: left, center, or right
- characterLimit - (int)
- fontSize - (int)
- fontColor (string) example: #0000FF for blue
- validation (string) value: any one from None(default) or Numbers or Letters or RegexValidation(in case of text type) or CanadianSin(Canadian SIN Number)
- hideFieldNameForRecipients - (boolean) value: either true or false

`formulafield` type

- party - (int) party index from the parties submitted in this request starting from 1
- formulafieldName - (string)
- formula - (string)
- fontFamily - (string)
- documentNumber - (int)
- fontSize - (int)
- fontColor (string) example: #0000FF for blue
- x - (int) the x location coordinate of the top left corner of the field in pixels
- y - (int) the y location coordinate of the top left corner of the field in pixels
- width - (int) the width of the field in pixel
- height - (int) the height of the field in pixel
- pageNumber - (int) page in the document where field needs to be added
- tabOrder - (int)
- decimalPlaces - (int)

`payfield` type

- payeeOptions - (string) Specify payment methods for the payee to choose from any of the following options:["CardPayment"] or ["AchBankPayment"] or ["CardPayment", "AchBankPayment"]
- paymentType - (string) Determines payment amount type: "Fixed" or "Payee decides". Note: "Payee_decides" in case of text tag.
- paymentAmount - (string) Required for 'Fixed' payment type, must be an integer or have up to two decimal places
- paymentDescription - (string) Description of the payment (Character limit: 250)
- productAndService (string) Name of the product or service (Character limit: 75)
- documentNumber - (int)
- pageNumber - (int) page in the document where field needs to be added
- x - (int) the x location coordinate of the top left corner of the field in pixels

- y - (int) the y location coordinate of the top left corner of the field in pixels
- tabOrder - (int)
- width - (int) the width of the field in pixel
- height - (int) the height of the field in pixel
- party - (int) party index from the parties submitted in this request starting from 1
  Note:
  1. The Payment field is available exclusively in the US region and initially supports payments only in US Dollars.
  2. The Super Admin must create a merchant account to use the Payment field.

`signature` or `initial` type

- party - (int)
- required - (boolean) value: either true or false

`date` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- value - (string)
- required - (boolean) value: either true or false
- textAlignment - (string) value: left, center, or right
- fontSize - (int)
- fontColor (string) example: #0000FF for blue
- dateFormat (string) example: MM-DD-YYYY for 04-30-2019

`secure` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- value - (string)
- required - (boolean) value: either true or false
- fontSize - (int)
- fontColor (string) example: #0000FF for blue

- charToDisplay (int)
- hideFieldNameForRecipients - (boolean) value: either true or false

`checkbox` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- group - (string)
- required - (boolean) value: either true or false
- multicheck - (boolean) value: either true or false
- hideCheckboxBorder - (boolean) value: either true or false

`dropdown` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- value - (string)
- required - (boolean) value: either true or false
- fontSize - (int)
- fontColor (string) example: #0000FF for blue
- options (array of strings)

`attachment` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- required - (boolean) value: either true or false

`image` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- required - (boolean) value: either true or false
- inputType - (string) value: either url or base64

- imageName - (string) image name with extension(png, jpg or jpeg)
- source - (string) value: either url or base64 value

`signer name` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- required - (boolean) value: either true or false
- fontSize - (int)
- fontColor (string) example: #0000FF for blue
- readOnly - (boolean) value: either true or false
- systemField - (boolean) value: either true or false

`date signed` type

- party - (int) party index from the parties submitted in this request starting from 1
- name - (string)
- tooltip - (string)
- required - (boolean) value: either true or false
- textAlignment - (string) value: left, center, or right
- fontSize - (int)
- fontColor (string) example: #0000FF for blue
- dateFormat (string) example: MM-DD-YYYY for 04-30-2019
- readOnly - (boolean) value: either true or false
- systemField - (boolean) value: either true or false

`accept` or `decline` type

- party - (int)

Apply a Template while uploading the documents

- You can copy template fields to the document via API while using the call: /folders/createfolder
- To copy template fields while uploading the

document, add the following to your document folder creation API call:

Parameter	Description
applyTemplate (default: `false`)	Value can be either `true` or `false`. If `true`, It will process the copy template fields.
templateIds	An array of template IDs you want to use to copy template fields into the documents for this folder. You can determine the template ID from the template URL. (https://{{HOST_NAME}}/templates/prepareimmutabletemplate?template.templateId={TEMPLATE_ID})

Fields JSON Object Example

Plain Text

```
"fields":
[
    {
        "type":"text",
        "x":100,
        "y":50,
        "width":60,
        "height":20,
```

Embed Foxit eSign within your application

There are two different ways to embed a Foxit eSign view into an application or website.

- Embedded Signing View
- Embedded Sending View

Note: In both Embedded Signing View and Embedded Sending view, please include the following script within your HTML for a better user experience:

```
Plain Text
```

Please also set the iFrame ID as: `esignIframe`

Create embedded signing view

When embedded signing view is enabled, you can show a document within your application.

The embedded signing view URL obtained via embeddedSessionURL will look like the following: `https://{{HOST_NAME}}/embedded/embeddedsign?eetid={URL-ENCODED EMBEDDED-TOKEN}`

You can embed Foxit eSign in your application using iFrame like:

```
html

<divstyle="height: 100%; width: 100%; ove
```

You can style the outer div element as per your application look and feel.

Foxit eSign adds the following two more parameters to the original success url:

Parameter	Description
folderId	This is the id of the folder that was sent by the party in the embedded session. You can use this id to further query our API to get the document status, etc.
event	Its value is `signing_success`, if the user successfully signs the document. Or `signing_declined`, if the user declines to sign the document.

Create embedded sending view

When embedded sending view is enabled, you can prepare a document within your application where you can drag and drop various fields on your document.

The embedded sending view URL obtained via embeddedSessionURL will look like the following: `https://{{HOST_NAME}}/embedded/embeddedsend?eetid={URL-ENCODED-EMBEDDED-TOKEN}`

You can embed Foxit eSign in your application using iFrame like:

```
html

<divstyle="height: 100%; width: 100%; ove
```

You can style the outer div element as per your application look and feel.

In the iframe, you will see the document which the User needs to sign.

Once the user successfully sends the folder, he/she

will be redirected to application success URL which you submitted in the request.

Foxit eSign adds the following two more parameters to the original success url:

Parameter	Description
folderId	This is the id of the folder that was sent by the party in the embedded session. You can use this id to further query our API to get the document status, etc.
event	If the user successfully sends the document, Its value is sending_success, if the user successfully sends the document.

Reusable Templates

What is a Template?

A template is a reusable form that may be used multiple times to send a document to recipients. The only changes are the signatures and data collected. Templates will only need to be set up once. Next time the form is used, it will be located under the Templates tab and contain all fields that have already been placed and saved.

Foxit eSign supports multiple formats including DOC, DOCX, PDF, XLSX, XLS, PPT, PPTX, CSV, TXT, RTF, and PNG. Select a file from a device, or from cloud storage systems such as Google Drive, Box, Dropbox or OneDrive.

Preparing a Template for Reuse

Adding Parties/Roles

When creating a template, party roles may be added

under the Recipient Parties tab on the right side of the application. With a template, there are two types of parties and a few ways to add party roles.

Dynamic and Static Party Roles

A dynamic party role can be assigned if the recipient's information changes each time a document is sent out. A party can be left blank displaying Role (optional), or it can be identified with a label. For example, a common label for a party role would be 'Client' or 'Patient'.

A static party includes mandatory recipient information such as first name, last name, and email when added. This information does not change and is used each time a template is used to send out a document. A common use case for a static party could be a coworker that will always counter-sign or receive a copy each time a document is completed.

Add Recipient

By typing underneath , a static party can be added to the template using the name or email address of a recipient from the account's address book.

Party/Role

Select Party/Role to add an additional dynamic party role. Dynamic party roles will be labeled as PN1, PN2, etc.

Optional: Name the type of role the recipient has. For example; Patient, student, legal guardian, employee, client.

Address Book

Search for a recipients name or e-mail address under the Add Recipient section allowing you to quickly add a recipient you have sent an envelope to previously.

Others

You can click Add Others to add a new recipient.

Add Yourself

You can click Add Me to add yourself as a recipient.

Permission Levels

Choose different permission levels for each party.

‘Fill out fields and sign’
‘Fill out fields only’
‘CC/View only’
‘Edit and sign’ the document
‘Fill out fields and sign’ is the default for all parties.

Fields

The fields will automatically be assigned to the party that is highlighted and will have a color-coded tag next to it and the color of the field corresponds with the party who is responsible for it. You can easily reassign a field to another party by clicking on the field, go to ‘Field properties’ and reassign the party.

On the left-hand side you will see your toolbox.

- ‘Signature fields’ are pre-populated fields
- ‘Data entry fields’ require input from a signer
- ‘Advanced fields’ includes secured fields to use for sensitive data (SSN or bank account information for example). - Use Attachment and Image Fields for uploading files and photos

Resize Field

Drag the bottom corner arrow of a field to adjust the size.

Field Properties

Under Field Properties, there are various ways to customize a field. Fields can be made mandatory, checkboxes can be grouped together, text alignment can be changed, add conditional logic, set a format for input values, change font size and color.

Create an Envelope

Templates will be stored as documents inside an envelope. More templates may be added to the envelope before sending. Click the add template

button and select another document.

Assign Recipients

Recipients information will be added inside each party role box.

Address Book

You can search for a name or e-mail address of a recipient you have sent a document to previously and add them as a party.

Others

You can click ‘Add Others’ to add a new recipient.

Add Yourself

You can click ‘Add Me’ to add yourself as a recipient.

Authentication Levels

Additional security may be added to verify a recipients identity by using different authentication levels.

No Authentication
SMS Document Link
Email Two-Factor Authentication
Mobile Two-Factor Authentication
Phone Two-Factor Authentication
User-defined Access Code
Knowledge Based Authentication
Workflows
Signing sequence forces the recipients to sign in a specific order the sender set. By default all recipients can sign in parallels with each other.

In-person signing is recommended when recipients are signing physically in-person and the recipients do not have access to an email address.

🔌 Webhooks

Our webhooks enable you to monitor changes in your document folders without polling the Foxit eSign platform.

Webhook is a URL handled by your application which will be called by Foxit eSign, based on certain events. As Foxit eSign is calling your application, your application should be publicly accessible on the internet.

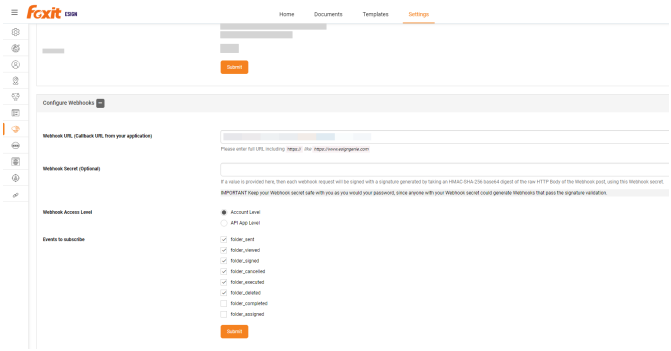
You can add your webhook in Foxit eSign API Settings page. Also you can define the events that will cause Foxit eSign to call your webhook. Eg, you can decide whether the webhook should be called when any party signs your folder, or only when the folder is finally executed.

Foxit eSign posts JSON data to your webhook. Every JSON payload posted contains event name (event_name), event date (event_date) and data related to that particular event (data).

Configuration Page

Webhooks can be configured by accessing:

1. Settings tab
2. API from the left panel
3. Configure Webhooks section at the bottom of API page



alt text

Webhook Access Level

Event Name	Description
Account Level	Foxit eSign calls your webhook events whenever an envelope is sent out from your account via application or API
App Level	Foxit eSign calls your webhook events whenever an envelope is sent out from your account via API only

Webhook Events

Event Name	Description
folder_sent	Foxit eSign calls your Webhook for this event whenever any document folder is sent out from your account for signature. The `data` field in this case consists of the folder (`folder`) field, which contains the data for the folder which was sent. This webhook call is in realtime
folder_viewed	Foxit eSign calls your Webhook for this event whenever any

📝 Sample Webhook Request Body

Plain Text

```
{
```

```
    "event_name":"folder_signed",
    "event_date":1464237988093,
    "data":{
      "folder":{
        "folderId":649,
        "folderName":"NDA",
        "folderAuthorId":1,
```


Webhook Security

It is recommended to use HTTPS for the webhook urls at your application end to avoid any kind of tampering with your webhook request data.

Signature verification

You can enable extra security around your Webhooks by providing a Webhook Secret (in API settings page, under section of webhooks) which will enable Foxit eSign to sign each of the webhook post request with a signature using your Webhook Secret.

The signature is generated using an HMAC-SHA-256 base64 digest of the raw HTTP Body of the Webhook post using this Webhook secret.

> Using your Webhook Secret a `signature` is calculated and sent with each Webhook (as a query string parameter signature). You can thus verify the contents of Webhook as being authentic and un-tampered.

You can generate the signature in php as follows:

```php
$request_body = file_get_contents('php://
$s = hash_hmac('sha256', $request_body, 'r
echo base64_encode($s);
```

This Signature is sent with each Webhook post as a query parameter signature in your URL.
For example, for the following URL you registered as a Webhook:

https://www.test.com/WebhookHandler

Foxit eSign will post to the following webhook URL:

https://www.test.com/WebhookHandler?signature=XXXXXXXXXXXXXXXXXXXX

You can thus verify the contents of the POST by taking the same hash yourself and comparing with a signature you got in request.

IMPORTANT: Keep your Webhook secret safe with you as you would your password, since anyone with your Webhook secret could generate Webhooks that pass the signature validation.

Creating Multiple Webhook Channels

Choose which notification types are sent to each channel. Plus, monitor individual channels through our developer portal.

With Foxit eSign's Various Advanced APIs for Managing your Webhooks:

- Create a Webhook Channel
- Update a Webhook Channel
- Deactivate a Webhook Channel

Next Steps

To get the most out of your integration, here's what we recommend:

- **Dive into the API Reference:** For comprehensive details on every endpoint, request, and response, explore our full API documentation.
- **Contact Sales for Production Keys:** When you're ready to go live, reach out to our sales team to acquire your production API keys.
  Contact Sales

PDF Embed API

Foxit PDF Embed API Overview

The **Foxit PDF Embed API** offers a streamlined way to

integrate a powerful PDF viewer into your web applications with minimal coding. This cloud-based solution enables fast deployment without requiring complex server configurations or reliance on third-party PDF tools. It eliminates the need for download prompts or additional plugins, allowing users to view and edit PDFs directly within the browser.

Customizable and Developer-Friendly

The viewer is fully customizable, giving developers the flexibility to adjust its appearance and behavior to match specific design and branding needs, no design expertise required.

Powered by Foxit's Trusted PDF Engine

At its core, the Embed API uses **Foxit's industry-leading PDF rendering engine**, trusted by major enterprises worldwide for its speed and reliability. This robust foundation ensures smooth and consistent document rendering across platforms. Additionally, Foxit provides dedicated technical support and routine performance and security updates, available through optional support and maintenance plans.

Copy, Paste, and Run

To launch the PDF viewer on your local web server (e.g., `localhost`), you'll need a valid **Client ID** and **Client Secret** from the Foxit Developer Console. Simply replace `'REPLACE_YOUR_CLIENT_ID'` and `'REPLACE_YOUR_CLIENT_SECRET'` in the provided code snippet with your actual Client ID and Client Secret. Then, run the webpage to initialize the viewer.

You can also modify the `previewFile` function to load a PDF from your server.

> 💡 Note: Make sure your local web server is properly configured. If you're using Node.js, learn more about setup here.

```javascript
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=dev
  <title>Foxit Embed - Full Window</title>
  <style>
    html, body, #foxit-embed-view { height
  </style>
```

Supported Browsers

Foxit Embed API is supported on the latest versions of the following browsers:

- **Windows**: Microsoft Edge, Google Chrome, Mozilla Firefox.
- **macOS**: Safari, Google Chrome, Microsoft Edge, Mozilla Firefox.
- **Android**: Google Chrome.
- **iOS**: Safari, Google Chrome.

Read more about PDF APIs

Next Steps

To get the most out of your integration, here's what we recommend:

- **Dive into the API Reference:** For comprehensive details on every endpoint, request, and response, explore our full API documentation.
- **Contact Sales for Production Keys:** When you're ready to go live, reach out to our sales team to acquire your production API keys.
  Contact Sales

PDF Services APIs

PDF Operations

Foxit PDF Services API provides modern cloud-based capabilities for PDF manipulation.

- Upload the document and start operations
- Add Security to PDFs by adding Passwords
- Optimize with PDF linearization
- Modify the PDF by splitting, manipulating, flattening and compressing
- Enhance the PDFs by combining two or more files
- Create PDFs from different file formats
- Convert PDFs to different formats
- Analyze PDFs
- Download document after operations

Security

Add or Remove Security on PDFs

- Remove PDF Password
- Protect PDF with Password

POST pdf-remove-password

https://na1.fusion.foxit.com/pdf-services/api/documents/security/pdf-remove-password

Remove password protection from a PDF document. Requires the current password to remove protection. The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- `documentId` *(required)* — The ID of the

document generated by the Upload Document API (`/api/documents/upload`).

- **`password`** (*required*)- The document password. The password parameter will remove the user password if it matches, and will also remove the owner password if it matches as well.

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "{{documentId}}",
  "password": "hesoyam"
}
```

POST pdf-protect

https://na1.fusion.foxit.com/pdf-services/api/documents/security/pdf-protect

Apply password protection to a PDF document.
The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- **`documentId`** *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

- **`config`** *(object)* — Configuration options for setting passwords and permissions:
  - **`userPassword`** — Password required to open the PDF (user-level access).
  - **`ownerPassword`** — Password granting full control over the PDF (owner-level access).
  - **`userPermissions`** *(array)* — Define the allowed operations for users:
    - `"PRINT_HIGH_QUALITY"` — Allow high-quality printing of the PDF.
    - `"PRINT_LOW_QUALITY"` — Allow standard/low-quality printing.
    - `"EDIT_CONTENT"` — Allow modification of PDF content. Grants permission to edit the document contents.
    - `"EDIT_FILL_AND_SIGN_FORM_FIELDS"` — Allow users to fill interactive form fields, including signature fields.
    - `"EDIT_ANNOTATION"` — Allow editing text annotations and form fields. If `"EDIT_CONTENT"` is also set, users can create or modify interactive form fields.
    - `"EDIT_DOCUMENT_ASSEMBLY"` — Allow document assembly actions such as inserting, rotating, or deleting pages, and creating bookmarks or thumbnails. This permission is granted regardless of whether `"EDIT_CONTENT"` is set.
    - `"COPY_CONTENT"` — Allow extraction of text and graphics for accessibility or other purposes.
  - **`cipher`** — The encryption algorithm to use. Options: `"AES_128"`, `"AES_256"`, `"RC4"`.
  - **`encryptMetadata`** — Boolean (`true` or `false`). Indicates whether to encrypt the document's metadata.

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

json

```
{
  "documentId": "{{documentId}}",
  "config": {
    "userPassword": "user123",
    "ownerPassword": "admin456",
    "userPermissions": [
      "PRINT_HIGH_QUALITY",
      "EDIT_FILL_AND_SIGN_FORM_FIELDS"
    ],
    "cipher": "AES_256",
    "encryptMetadata": true
```

Optimize

Linearize the documents

Optimize PDFs for quick viewing on the web.

POST pdf-linearize

https://na1.fusion.foxit.com/pdf-services/api/documents/optimize/pdf-linearize

Optimize PDF document for fast web viewing by linearizing the document structure

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "{{documentId}}"
}
```

Modify

Modify the documents

- Split PDF
- Manipulate PDF
- Flatten PDF
- Extract PDF
- Compress PDF

POST pdf-combine

https://na1.fusion.foxit.com/pdf-services/api/documents/enhance/pdf-combine

Combine multiple PDF documents into a single PDF file.
Allows merging an array of PDF files in a specified order to create one consolidated document.
The operation is asynchronous - use the returned taskId to track the operation status.

Key features:

- Merge multiple PDF files into a single document
- Maintain bookmarks and other document properties

Parameter Details

- `documentInfos` *(array)* — An array of document IDs you want to merge into a single PDF.
  - `documentId` — The ID of the document generated by the Upload Document API (`/api/documents/upload`).
  - `password` — Password for the document, if it is protected. Leave empty for open (unsecured) documents.
- `config` *(object)* — Configuration options for merging the PDF documents.
  - `addBookmark` *(boolean)* — Set to `true` to add bookmarks for each merged document.
  - `continueMergeOnError` *(boolean)* — Set to `true` to continue merging even if an error occurs with one or more documents.
  - `retainPageNumbers` *(boolean)* — Set to `true` to retain the original page numbering logic of each document.
  - `addToc` *(boolean)* — Set to `true` to include a table of contents in the merged PDF.
  - `tocBookmarkLevels` *(integer)* — Specifies the number of bookmark levels to include in the table of contents. Accepts values from `1` to the deepest level

available.

- `tocTitle` *(string)* — The title to be displayed for the table of contents.

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentInfos": [
    {
      "documentId": "doc_123"
    },
    {
      "documentId": "doc_456",
      "password": "secret123"
    }
  ],
  "config": {
```

POST pdf-compress

https://na1.fusion.foxit.com/pdf-services/api/documents/modify/pdf-compress

Optimize PDF file size by reducing image resolution, applying compression algorithms, and removing unnecessary elements.
The operation is asynchronous - use the returned taskId to track the operation status.

Compression levels:

- HIGH: Maximum compression with potential quality trade-offs
- MEDIUM: Balanced compression maintaining good quality
- LOW: Light compression preserving maximum quality

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).
- `compressionLevel` *(optional)* — Defines the level of compression to apply to the PDF. The default value is `"LOW"` .
  Supported values are: `"HIGH"` , `"MEDIUM"` , and `"LOW"` .

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc_abc123",
  "compressionLevel": "MEDIUM"
}
```

POST pdf-extract

https://na1.fusion.foxit.com/pdf-services/api/documents/modify/pdf-extract

Extract text, image, or specific pages from a PDF document.
The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).
- `pageRange` *(optional)* — Specifies the page range to extract content from.
  If not provided, the entire document will be processed.
  Usage examples:
  - For specific pages: `"1,8,9"`
  - For a range of pages: `"1-30"`
  - To extract all pages: `"all"`
- `extractType` *(required)* — Defines the type of content to extract from the PDF.
  Supported values:
  - `"TEXT"` — Extracts all text content from the specified pages and the response will be plain text.
  - `"IMAGE"` — Extracts images from the PDF pages. The output will be delivered as a `.zip` file containing the extracted images.
  - `"PAGE"` — Extracts specific pages from a PDF and generates a new PDF containing only the selected pages.

AUTHORIZATION

Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID

client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```
json

{
  "documentId": "doc_abc123",
  "pageRange": "1-5,8,10-12",
  "extractType": "TEXT"
}
```

POST pdf-flatten

https://na1.fusion.foxit.com/pdf-services/api/documents/modify/pdf-flatten

Flatten form fields and annotations in a PDF document, making them part of the page content.
The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- **`documentId`** *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc_abc123"
}
```

POST pdf-manipulate

https://na1.fusion.foxit.com/pdf-services/api/documents/modify/pdf-manipulate

Reorganize, rotate, or delete pages in a PDF document.
The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- **`documentId`** *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).
- **`config`** *(required)* — Configuration object used to apply various manipulation operations on the PDF:
  - **`operations`** *(array)* — List of operations to perform on the document. Each operation supports the following fields:
    - **`type`** — The type of operation to apply. Supported values include: `"MOVE_PAGES"`, `"ROTATE_PAGES"`, `"DELETE_PAGES"`, and `"ADD_PAGES"`.
    - **`pages`** *(array)* — An array of specific page numbers where the operation should be applied.
    - **`targetPosition`** *(integer)* — Used with `"MOVE_PAGES"` to specify the page number where the selected pages should be moved.

- `rotation` — Specifies the rotation angle for the `"ROTATE_PAGES"` operation. Supported values are: `"ROTATE_0"`, `"ROTATE_CLOCKWISE_90"`, `"ROTATE_180"` and `"ROTATE_COUNTERCLOCKWISE_90"`
- `pageCount` — Indicates the number of pages to be added when using the `"ADD_PAGES"` operation.

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc_abc123",
  "config": {
    "operations": [
      {
        "type": "MOVE_PAGES",
        "pages": [
          1,
          2,
          3
        ],
```

POST pdf-split

https://na1.fusion.foxit.com/pdf-services/api/documents/modify/pdf-split

documents/modify/pdf-split

Split a PDF document into multiple files based on specified criteria.
The API returns multiple PDF files in a zip file.
The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).
- `pageCount` (*required, integer*): Number of pages in each split file.

Response Details

The output will be a zip containing multiple PDFs.

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "{{documentId}}",
  "pageCount": 10
}
```

Create

Create PDFs from other documents format

- PDF from Word
- PDF from URL
- PDF from Text
- PDF from PPT
- PDF from Image
- PDF from HTML
- PDF from Excel

POST pdf-from-excel

https://na1.fusion.foxit.com/pdf-services/api/documents/create/pdf-from-excel

Convert a Microsoft Excel spreadsheet to a PDF file. Supports various Excel formats: .xls, .xlsx, .xlt, .xltx, .xlsm, .xlsb, .xltm, .csv.
The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION

Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

Body raw (json)

```json
{
  "documentId": "doc_abc123"
}
```

POST pdf-from-html

https://na1.fusion.foxit.com/pdf-services/api/documents/create/pdf-from-html

Convert HTML content to a PDF document with customizable page settings.

Features:

- Configurable page dimensions (default: A4)
- Page orientation control through rotation
- Flexible content layout modes (single or multiple pages)
- Content scaling options

Common Use Cases:

1. Web page archiving
2. Report generation from HTML templates
3. Creating printable documents from web content
4. HTML newsletter to PDF conversion

Notes:

- The operation is asynchronous
- Use the returned taskId with Task Status API to track progress
- Maximum input HTML size: 100MB
- Supports embedded CSS, images, and web fonts
- External resources must be accessible to the service

Parameter Details

- **`documentId`** *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).
- **`config`** *(object)* — Configuration options for the PDF output.
  - **`dimension`** *(object)* — Specify the output dimensions in pixels.
    - **`width`** — Width of the PDF in pixels.
    - **`height`** — Height of the PDF in pixels.
  - **`rotation`** *(string)* — Rotation angle for the output. Use `"NONE"` to keep the original orientation.
  - **`pageMode`** *(string)* — Set the page display mode:
    - `"SINGLE_PAGE"` — Renders the document on a single page.
    - `"MULTIPLE_PAGE"` — Allows the content to span multiple pages.
  - **`scalingMode`** *(string)* — Set to `"SCALE"` to enable scaling of content.

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc123",
```

```
  "config": {
    "dimension": {
      "width": 612,
      "height": 792
    },
    "rotation": "NONE",
    "pageMode": "MULTIPLE_PAGE",
    "scalingMode": "SCALE"
}
```

POST pdf-from-image

https://na1.fusion.foxit.com/pdf-services/api/documents/create/pdf-from-image

Convert one or more images to a PDF document.
Supported formats:

- JPEG/JPG
- PNG
- TIFF
- BMP
- GIF
  The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc_abc123"
}
```

POST pdf-from-word

https://na1.fusion.foxit.com/pdf-services/api/documents/create/pdf-from-word

Convert a Microsoft Word document to a PDF file.
Supports various formats: .doc, .docx, .rtf, .dot, .dotx, .docm, .dotm, .wpd.
The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- **`documentId`** *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
```

```
{
  "documentId": "doc_abc123"
}
```

POST pdf-from-ppt

https://na1.fusion.foxit.com/pdf-services/api/documents/create/pdf-from-ppt

Convert a Microsoft PowerPoint presentation to a PDF file.

Parameter Details

- **`documentId`** *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

json

```
{
  "documentId": "doc_abc123"
}
```

POST pdf-from-text

https://na1.fusion.foxit.com/pdf-services/api/documents/create/pdf-from-text

Convert a plain text file to a PDF document.

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc_abc123"
}
```

POST pdf-from-url

https://na1.fusion.foxit.com/pdf-services/api/documents/create/pdf-from-url

Convert a web page (URL) to a PDF document. Same as pdf-from-html but with a URL as input.
The operation is asynchronous - use the returned

taskId to track the operation status.

Parameter Details

- `url` *(required)* — A publicly accessible URL of the web page to be converted.
- `config` *(object)* — Configuration settings for the output PDF.
  - `dimension` *(object)* — Specify the output dimensions in pixels.
    - `width` — The width of the PDF in pixels.
    - `height` — The height of the PDF in pixels.
  - `rotation` *(string)* — Set the rotation angle for the output. Use `"NONE"` to retain the original orientation.
  - `pageMode` *(string)* — Choose the page layout mode:
    - `"SINGLE_PAGE"` — Render content on a single page.
    - `"MULTIPLE_PAGE"` — Allow pagination across multiple pages.
  - `scalingMode` *(string)* — Specify scaling behavior. Use `"SCALE"` to enable scaling.

AUTHORIZATION

Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body

raw (json)

```json
{
  "url": "https://www.example.com/page-to
  "config": {
```

```
  "dimension": {
    "width": 595,
    "height": 842
  },
  "rotation": "NONE",
  "pageMode": "MULTIPLE_PAGE",
  "scalingMode": "SCALE"
}
```

Convert

Convert PDFs to other document formats

- PDF to Word
- PDF to Text
- PDF to PPT
- PDF to Image
- PDF to HTML
- PDF to Excel

POST pdf-to-excel

https://na1.fusion.foxit.com/pdf-services/api/documents/convert/pdf-to-excel

Convert PDF tables to Microsoft Excel spreadsheet (.xlsx).

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION

Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc_abc123"
}
```

POST pdf-to-html

https://na1.fusion.foxit.com/pdf-services/api/documents/convert/pdf-to-html

Convert PDF to web-friendly HTML format.

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "documentId": "doc_abc123"
}
```

POST pdf-to-image

https://na1.fusion.foxit.com/pdf-services/api/documents/convert/pdf-to-image

Convert PDF pages to images.

Parameter Details

- `documentId` (*required*) – The ID of the document, generated by the Upload Document API (`/api/documents/upload`).
- `pageRange` – Specifies the range of pages in the PDF to convert to images.
  Use `"1-5"` for a range, or `"1,3,7"` for specific pages. Leave empty to convert all pages.
- `config` – Configuration settings for image conversion.
  - `dpi` – DPI range 1-1000.Default value: 96. Higher values yield better image quality but larger file sizes.

AUTHORIZATION

Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```
json

{
  "documentId": "doc_abc123",
  "pageRange": "1-5",
  "config": {
    "dpi": 96
  }
}
```

POST pdf-to-ppt

https://na1.fusion.foxit.com/pdf-services/api/documents/convert/pdf-to-ppt

Convert PDF to PowerPoint presentation (.pptx).

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```
json
```

```
{
  "documentId": "doc_abc123"
}
```

POST pdf-to-text

https://na1.fusion.foxit.com/pdf-services/api/documents/convert/pdf-to-text

Convert PDF text content to plain text.

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION

Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body

raw (json)

```
json

{
  "documentId": "doc_abc123"
}
```

POST pdf-to-word

https://na1.fusion.foxit.com/pdf-services/api/documents/convert/pdf-to-word

Convert PDF to Microsoft Word document (.docx).

Parameter Details

- `documentId` *(required)* — The ID of the document generated by the Upload Document API (`/api/documents/upload`).

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
    "documentId": "6838b8ea6b1cc44ab40036
}
```

Analyze

Compare PDFs

Compare one PDF file (as 'base document') with another PDF file (as 'compared document') page by page.

POST pdf-compare

POST pdf-compare

https://na1.fusion.foxit.com/pdf-services/api/documents/analyze/pdf-compare

Compare two PDF documents and generate a comparison report highlighting the differences. The operation is asynchronous - use the returned taskId to track the operation status.

Parameter Details

- `baseDocument` (*object*) – The primary reference document to compare against.
  - `documentId` – The ID of the base document, generated by the Upload Document API.
  - `password` – The password for the base document, if it's protected (optional).
- `compareDocument` (*object*) – The document to be compared with the base document.
  - `documentId` – The ID of the document to compare.
  - `password` – The password for the compare document, if it's protected (optional).
- `config` (*object*) – Settings to configure the comparison output.
  - `compareType` – Defines the type of comparison:
    - `"ALL"` – Compare both text and visual differences.
    - `"TEXT"` – Compare text content only.
  - `resultType` – Defines the format of the comparison result:
    - `"PDF"` – Output a PDF highlighting differences.
    - `"JSON"` – Output structured JSON detailing the differences.

AUTHORIZATION Basic Auth

HEADERS

Content-Type	application/json
client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "baseDocument": {
    "documentId": "doc_abc123",
    "password": "optional_password"
  },
  "compareDocument": {
    "documentId": "doc_xyz789",
    "password": "optional_password"
  },
  "config": {
    "compareType": "ALL",
```

Download Document

Download a document by its ID.

GET download-document

https://na1.fusion.foxit.com/pdf-services/api/documents/:document-id/download?filename=financial-report-2023

Download a document by its ID.
The response is streamed as binary content and includes:

- Content-Type header matching the document type (e.g., 'application/pdf', 'image/jpeg')

- Content-Disposition header for browser download handling (e.g., 'attachment; filename="document.pdf"')
- Document content as a binary stream

The optional filename parameter can be used to override the original filename.

File Size & Retention Policy

- Maximum supported file size is **100 MB**
- Uploaded files are **retained for 24 hours** before automatic deletion

Parameter Details

- `documentId` (*path variable*, *required*) — The ID of the document generated by the Upload Document API (/api/documents/upload).
- `filename` (*query*, *optional*) — Custom filename for the downloaded document, file extension is not needed.

HEADERS

client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

PARAMS

filename	financial-report-2023 Optional custom filename for the downloaded document, file extension is not needed.

PATH VARIABLES

document-id	6838c9cf6b1cc44ab40040cb ID of the document to download

Upload Document

Upload a document to the Foxit and generate a documentId for further PDF operations.

POST upload-document

https://na1.fusion.foxit.com/pdf-services/api/documents/upload

Upload a document for processing. Returns a document ID that can be used in other operations. Supports various file formats including:

- PDF documents
- Microsoft Office documents (Word, Excel, PowerPoint)
- Images (PNG, JPEG, TIFF)
- Text files

File Size & Retention Policy

- Maximum supported file size is **100 MB**
- Uploaded files are **retained for 24 hours** before automatic deletion

Parameter Details

- `file` (*required*) - Upload a document to the Foxit and generate a **documentId** for further PDF operations.

AUTHORIZATION

Basic Auth

Username	<username>
Password	<password>

HEADERS

client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body formdata

file

Get Status

Get the current status of an asynchronous operation. The response includes:

- Task state (PENDING, PROCESSING, COMPLETED, FAILED)
- Progress percentage (0-100)
- Result document ID (when completed successfully)
- Error details (if failed)

GET get-task-status

https://na1.fusion.foxit.com/pdf-services/api/tasks/:task-id

Get the current status of an asynchronous operation. The response includes:

- Task state (PENDING, PROCESSING, COMPLETED, FAILED)
- Progress percentage (0-100)
- Result document ID (when completed successfully)
- Error details (if failed)

AUTHORIZATION Basic Auth

HEADERS

client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

PATH VARIABLES

task-id	6838c9c66b1cc44ab40040c5 ID of the task to check status for

Document Generation APIs

POST Analyze Document

https://na1.fusion.foxit.com/document-generation/api/AnalyzeDocumentBase64

Analyze Document

Document Analyze API is designed to extract all predefined text tags from your template documents. It plays a key role in identifying the dynamic fields required for document generation, enabling developers to map values accurately. This API ensures that you capture every necessary variable, allowing you to construct the correct JSON payload for seamless integration with the **Document Generation API**.

Parameter Details

Utilize the API to analyze template documents and automatically extract their text tags, simplifying the programmatic mapping of data fields.

Request Parameters

1. `base64FileString` (*required*) - The Base64-

encoded string of the `.docx` document you want to analyze. The API will process the file, extract all text tags present in the document, and return their details in the response.

Response Parameters

The response will be in JSON format and include the following parameters:

- **`singleTagsString`** – Lists all single-value text tags detected in the Word document. These tags are intended to be mapped to individual data values.
- **`doubleTagsString`** – Lists multi-value (repeating) text tags identified in the document. These are used to dynamically populate multiple entries, such as rows in a table.

AUTHORIZATION Basic Auth

HEADERS

client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
    "base64FileString": "UEsDBBQABgAIAAAAI
}
```

POST Generate Document

https://na1.fusion.foxit.com/document-generation/api/GenerateDocumentBase64

Document Generation

Document Generation API is used to create dynamic, data-driven documents by injecting JSON input into predefined templates. Each key in the JSON payload corresponds to a text tag within the document, allowing for seamless replacement of placeholders with actual values. This enables automated, programmatic generation of complete documents tailored to specific data inputs, streamlining workflows and reducing manual effort.

Parameter Details

The API has request and response parameters to successfully generate documents.

Request Parameters

1. `documentValues` (object, required) - A set of key-value pairs used to populate the document dynamically. Each key should match a text tag in the document template (e.g., `{{name}}`), and the corresponding value will be inserted in place of that tag.
2. `base64FileString` (string, required)- The Base64-encoded string of the `.docx` document where you have added text tags. The API will process the file, map the tags on the document with JSON data.
3. `outputFormat` - The value could be `"PDF"` or `"DOCX"` to get the pdf or word output. The default value is "PDF".
4. `currencyCulture` (optional) - Specifies the culture code used to format currency values (e.g., for symbols, decimal separators, and thousands separators). The following currency codes are supported:

Culture Code	Country / Region	Symbol
en-US	United States	$
en-GB	United Kingdom	£

de-DE	Germany	€
fr-FR	France	€
hi-IN	India	₹
pt-BR	Brazil	R$
ru-RU	Russia	₽
es-ES	Spain	€
it-IT	Italy	€

Response Parameters

Response Parameters

- **`message`** – Success confirmation message upon document generation.
  - If the output is a PDF, the message will be: *"PDF created successfully."*
  - If the output is a Word document, the message will be: *"Word document created successfully."*
- **`fileExtension`** – The file extension of the generated document.
  - Default value: `"pdf"`
  - Possible value: `"docx"` (if a Word document is generated)
- **`base64FileString`** – The base64 string of generated document.

HEADERS

client_id	YOUR_CLIENT_ID
client_secret	YOUR_CLIENT_SECRET

Body raw (json)

```json
{
  "outputFormat": "pdf",
  "currencyCulture": "en-US",
  "documentValues": {
    "ShowAgreement": "true",
    "PaymentDueDate": "06/02/2025",
    "AccountName": "Jordan O'Connor",
    "Name": "Jordan",
    "OpportunityLineItems": [
    {
        "OpportunityLineItems.quantity":
```

eSign APIs

AUTHORIZATION OAuth 2.0

Envelopes

With Foxit's robust Envelope APIs, you can easily perform essential envelope operations such as:

- **Create, Update, and Delete Envelopes**
  Effortlessly manage the lifecycle of envelopes, create new ones, update details, or delete them when no longer needed.
- **Send Draft Envelopes**
  Finalize and send envelopes saved as drafts, streamlining your document workflow.
- **Modify Shared Envelopes**
  Collaborate efficiently by updating envelopes shared with team members or recipients.
- **Send Signature Reminders**
  Automatically trigger reminder emails to recipients who haven't signed yet, helping speed up turnaround time.
- **Update Fields in Envelopes**

  Dynamically update form fields such as text fields and dates on the fly.

These APIs are designed to provide flexibility and automation, empowering your application to handle digital signature workflows with ease and precision.

AUTHORIZATION OAuth 2.0

POST Create Envelope from URL

https://na1.foxitesign.foxit.com/api/folders/createfolder

Envelopes can be created and send with publicly accessible PDF files. Pass the array of URLs of PDFs in the body for documents to send.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

json

```
{
  "folderName": "eSignature Document",
  "fileUrls": [
    "https://developersguide.foxitesign.fo
  ],
  "fileNames": [
    "Example Service Contract.pdf"
  ],
  "parties": [
    {
```

```
"firstName": "john",
```


POST Send Draft Envelope

https://na1.foxitesign.foxit.com/api/folders/sendDraftFolder

Note: this endpoint will only send an envelope if it is currently in DRAFT mode. To create an envelope in DRAFT mode, please pass *sendNow* as `true` for any envelope creation endpoint (such as Create Envelope from URL)

AUTHORIZATION

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body

json


```
{
  "folderId": 2520579,
  "folderName": "Example Documents",
  "parties": [
    {
      "firstName": "John",
      "lastName": "Doe",
      "emailId": "john.doe@example.com",
      "permission": "FILL_FIELDS_AND_SIGN"
      "sequence": 1
    }
```


POST Update Envelope Fields

POST Update Envelope Fields

https://na1.foxitesign.foxit.com/api/folders/updateEnvelopeFields

You can update envelope and the custom fields `(custom_field1 and custom_field2)` anytime before execution using the 'Update Envelope Fields' API.

Note: Envelope fields for a party can only be updated if the document has not been signed by that party yet.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

```json
{
  "folderId": -50787227,
  "fields": {
    "FIELD1_NAME": "VALUE",
    "FIELD2_NAME": "VALUE"
  },
  "custom_field1": {
    "name": "NAME",
    "value": "VALUE"
  },
  "custom_field2": {
```

GET Get Envelope Details

https://na1.foxitesign.foxit.com/api/folders/myfolder?folderId=123456&access_token=<token>

You can pull our API to get the Envelope/Document data

AUTHORIZATION

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Content-Type	application/json
Accept	application/json

PARAMS

folderId	123456 (Required)
access_token	<token>

GET Get Envelope Ids

https://na1.foxitesign.foxit.com/api/folders/getAllFolderIdsByStatus?dateFrom=6789-11-04&dateTo=6789-11-04&status=EXECUTED

You can poll our API to get the Id(s) of multiple envelopes by providing a date range and (optionally) a status as filters.

AUTHORIZATION

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Accept	application/json
Authorization	Bearer <token>
Cache-Control	no-cache

PARAMS

dateFrom	6789-11-04 (Required) Start of the Envelope Creation Date range. Accepted format: YYYY-MM-DD
dateTo	6789-11-04 (Required) Start of the Envelope Creation Date range. Accepted format: YYYY-MM-DD. Note: dateTo value should be under the 6 months from dateFrom value.
status	EXECUTED Get FolderIds by the folder status. Status parameter can have any of the following values: EXECUTED,SHARED,DRAFT, PARTIALLY SIGNED, CANCELLED, EXPIRED and DELETED. Note: If you don't pass this parameter, then by default you will receive envelopes with any status.

POST Cancel Envelope

https://na1.foxitesign.foxit.com/api/folders/cancelFolder

With Foxit eSign API, you can cancel/decline to sign an Envelope by calling /folders/cancelFolder with the given parameters.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

```json
{
  "folderId": 51298042,
  "reason_for_cancellation": "fugiat ipsur
}
```

GET Download Envelope Files

https://na1.foxitesign.foxit.com/api/folders/download?folderId=12345678&access_token=<token>

You can download executed folder document(s) as well as signature certificate all at once in a zip file.

Returns ZIP file binary stream of all the documents in the folder with the signature certificate.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Accept	application/octet-stream

PARAMS

folderId	12345678 (Required) The ID of the Envelope for which you would like to obtain the binary zip file for.
access_token	<token>

POST Modify Shared Envelope

https://na1.foxitesign.foxit.com/api/folders/modifySharedFolder

Modify any shared envelope and send again to recipients for signing.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Accept	application/json
Content-type	application/json
Authorization	Bearer <token>

Body raw (json)

json

```
{
  "folderId": 31187216,
  "folderName": "eSignature Document",

  "parties": [
    {
```

```
"firstName": "John",
"lastName": "Doe",
"emailId": "john.doe@example.com",
"permission": "FILL_FIELDS_AND_SIGN"
"sequence": 1
```

POST Send Signature Reminder

https://na1.foxitesign.foxit.com/api/folders/signaturereminder

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Authorization	Bearer <token>
Content-Type	application/json
Accept	application/json

Body raw (json)

json

```
{
  "folderId": 58182357
}
```

GET Download Single Document PDF

https://na1.foxitesign.foxit.com/api/folders/document/download?folderId=12345678&docNumber=1&access_token=<token>

You can download individual document as a pdf file.

Returns PDF file binary stream of the document which you can save as PDF file.

AUTHORIZATION

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Accept	application/octet-stream

PARAMS

folderId	12345678 (Required) The ID of the Envelope for which you would like to obtain the binary zip file for.
docNumber	1 (Required) The index of the document starting from 1 whose PDF you want to download.
access_token	<token>

POST Move Envelopes to Recycle Bin

https://na1.foxitesign.foxit.com/api/folders/movetorecyclebin

Using Foxit eSign API you can move your Envelope(s) to the recycle bin. Documents in the recycle bin are permanently removed from Foxit eSign systems automatically after 14 days.

AUTHORIZATION

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Authorization	Bearer <token>
Accept	application/json
Content-type	application/json

Body raw (json)

```
json

{
  "folderIds": [
    12345678,
    87654321
  ]
}
```

POST Update Envelope Recipients

https://na1.foxitesign.foxit.com/api/folders/updateFolder

You can change the first name, last name, and email if the recipient party has not signed on a shared or a partially signed document using the `Update Envelope Recipients` API. Recipients can be changed only if the Envelope is in `DRAFT` status.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Accept	application/json
Content-type	application/json
Authorization	Bearer <token>

Body raw (json)

```
json

{
  "folderId": 23500405,
  "parties": [
    {
      "action": "update",
      "sequence": 1,
      "firstName": "John",
      "lastName": "Doe",
      "emailId": "john.doe@example.com",
      "dialingCode": "+1",
      "mobileNumber": "1234567890"
```

GET Get Envelope Activity History

https://na1.foxitesign.foxit.com/api/folders/viewActivityHistory?folderId=12345678&access_token=<token>

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Accept	application/json

PARAMS

folderId	12345678 (Required)
access_token	<token>

GET Get Deleted Envelope History

https://na1.foxitesign.foxit.com/api/folders/deletedLog?folderId=98765432&access_token=<token>

You can poll our API to get the deleted folder history.

AUTHORIZATION

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Accept	application/json

PARAMS

folderId	98765432 (Required)
access_token	<token>

POST Delete Envelopes

https://na1.foxitesign.foxit.com/api/folders/delete

Using Foxit eSign API you can remove your Envelope(s) permanently from Foxit eSign systems.

AUTHORIZATION

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Content-Type	application/json
Accept	application/json

Authorization	Bearer <token>

Body raw (json)

```
json

{
    "folderIds": [
        12345678
    ]
}
```

POST Download Report

https://na1.foxitesign.foxit.com/api/folders/getFolders/download?access_token=<token>

To download the author email, Folder Name, Folder Number, date sent, status etc via API of envelopes from a date range in an Excel file use this endpoint.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder Envelopes

HEADERS

Content-Type	application/json
Accept	application/vnd.ms-excel

PARAMS

access_token	<token> (Required) Oauth2.0 access token should be added here.

Body raw (json)

json

```
{
  "status": "EXECUTED",
  "creationDateFrom": "2022-01-01",
  "creationDateTo": "2022-04-01",
  "folderName": "example document",
  "includeFields": "false",
  "authorEmail": "user2@example.com",
  "signerEmail": "john.doe@example.com"
}
```

Templates

Streamline Your Workflow with Reusable Templates

Easily create reusable documents by saving them as **Templates**. Assign **party roles** (e.g., Sender, Signer, Approver) during setup, so you can quickly generate and send out documents for signature without starting from scratch each time.

AUTHORIZATION OAuth 2.0

This folder is using OAuth 2.0 from folder eSign APIs

POST Create Envelope from Template

https://na1.foxitesign.foxit.com/api/templates/createFolder

Create envelopes using existing templates.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Authorization	Bearer <token>
Content-Type	application/json
Accept	application/json

Body raw (json)

json

```
{
  "templateIds": [
    407369
  ],
  "parties": [
    {
      "firstName": "Peter",
      "lastName": "Parker",
      "emailId": "peter.parker@spiderman.c
      "permission": "FILL_FIELDS_AND_SIGN"
      "workflowSequence": 1,
```

GET Get a list of all Templates

https://na1.foxitesign.foxit.com/api/templates/list?templateId=407369&access_token=<token>

You can get a list of all the templates from your Foxit eSign account into your application using the following call.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

PARAMS

PARAMS

templateId	407369
access_token	<token>

POST Get Templates by Template Ids

https://na1.foxitesign.foxit.com/api/templates/templateDetails

You can get a list of all the templates from your Foxit eSign account into your application using the following call.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

json

```
{
  "templateIds": [
    226
  ]
}
```

POST Create Template from URI

https://na1.foxitesign.foxit.com/api/templates/createtemplate

Template can be created by uploading a PDF document. To create a template from a PDF file, provide publicly accessible URLs to PDF documents or pass PDF documents as multipart form data with the number of recipient parties, etc.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body

```json
{
  "templateName": "onboardingmulti_2.pdf",
  "templateUrl": "https://developersguide
  "processTextTags": true,
  "processAcroFields": true,
  "shareAll": false,
  "numberOfParties": 2,
  "parties": [
    {
      "permission": "FILL_FIELDS_AND_SIGN"
      "sequence": 1,
```

GET Get Template Details

https://na1.foxitesign.foxit.com/api/templates/mytemplate?folderId=407369&access_token=

<token>

You can poll our API to get the details of a template in JSON format.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

PARAMS

folderId	407369 (Required) The ID of the Template for which you would like to obtain parseable details about
access_token	<token>

Webhook

AUTHORIZATION

This folder is using OAuth 2.0 from folder eSign APIs

GET Get Webhook Channel Details

https://na1.foxitesign.foxit.com/api/webhook/mychannel?channelId=3959&access_token=<token>

You can poll our API to get information about a specific Webhook channel.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json
access_token	<token>

PARAMS

channelId	3959 (Required)
access_token	<token>

GET List all Webhook Channels

https://na1.foxitesign.foxit.com/api/webhook/channellist?access_token=<token>

To get a list of all the webhook channels in the account, use the following call.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

PARAMS

access_token	<token>

POST Update Webhook Channel

https://na1.foxitesign.foxit.com/api/webhook/updatewebhookchannel

To create a new channel via API, please make a call to /webhook/updatewebhookchannel with the following parameters.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json

Body raw (json)

```
json
{
  "channelId": "1",
  "channelName": "updateName",
  "webhookUrl": "https://abc.com/xyz",
  "webhookSecret": "updatesecret",
  "webhookLevel": "Account",
  "events": {
    "folder_sent": true,
    "folder_viewed": true,
    "folder_signed": true,
    "folder_cancelled": true,
```

GET Reactivate Webhook Channel

https://na1.foxitesign.foxit.com/api/webhook/channelreactivate?channelId=3959&access_token=<token>

To create a new channel via API, please make a call to

/webhook/updatewebhookchannel with the following parameters.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

PARAMS

channelId	3959 (Required) Your Channel ID received when creating the webhook. Can also be found in the Foxit eSIgn Settings page.
access_token	<token>

GET Deactivate Webhook Channel

https://na1.foxitesign.foxit.com/api/webhook/channeldeactivate?channelId=3959&access_token=<token>

This endpoint will deactivate your Webhook Channel.

Note: To reactivate an channel via API, please make a call to /webhook/channelreactivate

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	text/plain

PARAMS

channelId	3959 (Required) Your Channel ID received when creating the webhook. Can also be found in the Foxit eSIgn Settings page.
access_token	<token>

POST Delete Webhook Channel

https://na1.foxitesign.foxit.com/api/webhook/deletechannels

To delete a webhook channel via API, please make a call to this endpoint with the following parameters.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

json

```
{
    "channelIds": [
        4405
    ]
}
```

POST Create Webhook Channel

https://na1.foxitesign.foxit.com/api/webhook/createwebhookchannel

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

json

```
{
  "channelName": "abc",
  "webhookUrl": "https://abc.com/xyz",
  "webhookLevel": "Account",
  "events": {
    "folder_sent": true,
    "folder_viewed": true,
    "folder_signed": false,
    "folder_cancelled": true,
    "folder_executed": true,
    "folder_deleted": true
```

Parties

AUTHORIZATION OAuth 2.0

This folder is using OAuth 2.0 from folder eSign APIs

POST Create Email Group

https://na1.foxitesign.foxit.com/api/parties/createEmailGroup

To create a new email group via API, please make a call to parties/createEmailGroup with the following parameters.

You can only pass up to 20 party details.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	<token>

Body raw


```
{
  "emailGroupName": "Email_Group_Name",
  "emailGroupDescription": "Email_Group_
  "allowAdvancedEmailValidation": true,
  "parties": [
    {
      "firstName": "Sandra",
      "lastName": "Smith",
      "emailId": "san_smith@foxitesign.c
    },
    {
      "firstName": "Hannah",
      "lastName": "Pitt",
```


POST Get Email Group Details

https://na1.foxitesign.foxit.com/api/parties/get

EmailGroups

You can pool our API to get email group information.

You can pass this request with no parameters to get the details of all of the email groups in your account.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Breaer <token>

Body

```json
{
  "emailGroupNames": [
    "Email_Group_1",
    "Email_Group_2"
  ]
}
```

Users

AUTHORIZATION

This folder is using OAuth 2.0 from folder eSign APIs

POST Create User

https://na1.foxitesign.foxit.com/api/users/creat

e

This endpoint allows you to create a User in any account by using the API.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Breaer <token>

Body raw (json)

```json
{
  "user": {
    "firstName": "eSign",
    "lastName": "Demo",
    "emailId": "esigndemo@foxit.com",
    "allowAdvancedEmailValidation": true,
    "address": "Miami, Florida",
    "userRole": "admin",
    "department": "DEV",
    "title": "Tech Lead",
    "active": true,
```

POST Update User

https://na1.foxitesign.foxit.com/api/users/update

This endpoint allows you to update the User's profile parameters using the API.

AUTHORIZATION OAuth 2.0

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

```json
{
  "user": {
    "partyId": "54",
    "firstName": "eSign",
    "lastName": "Demo",
    "address": "Miami, Florida",
    "userRole": "super_admin",
    "department": "DEV",
    "title": "Tech Lead",
    "active": false
  }
```

GET Delete User

https://na1.foxitesign.foxit.com/api/users/delete?userId=54

This endpoint allows you to delete a user via API.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

PARAMS

userId	54 (Required) Party Id for that user

GET List all Users

https://na1.foxitesign.foxit.com/api/users/list

This endpoint allows you to list all of the users under a specific account via API.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

Accounts

AUTHORIZATION

This folder is using OAuth 2.0 from folder eSign APIs

POST Create New Account

https://na1.foxitesign.foxit.com/api/accounts/create

This endpoint allows you to create a new account as a partner by using the API.

Note: Instead of providing the `access_token` Authorization header from your partner portal, you will instead provide the `client_id` and `client_secret` as part of the body parameters of this call.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json

Body raw (json)

```
json

{
  "client_id": "12345678901234567890123456
  "client_secret": "1234567890123456789012
  "company": {
    "companyName": "Wayne Tech",
    "companyAddress": "LA, US"
  },
  "user": {
    "firstName": "Bruce",
    "lastName": "Wayne",
    "emailId": "esigndemo@foxit.com",
```

GET Cancel Account

https://na1.foxitesign.foxit.com/api/accounts/cancel

This endpoint allows you to cancel a specific account.

`Note:` please provide the `access_token` Authorization header from the account which you are looking to cancel.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

GET Reactivate Account

https://na1.foxitesign.foxit.com/api/accounts/reactivate

This endpoint allows you to reactivate a specific account.

`Note:` please provide the `access_token` Authorization header from the account which you are looking to reactivate.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

GET Change Licenses

https://na1.foxitesign.foxit.com/api/accounts/changenumberoflicences?newNumberOfLicences=20&partnerCode=125478sfg

This endpoint allows you to edit the number of licenses available in a a specific account.

`Note:` please provide the `access_token`

Authorization header from the account which you are looking to reactivate.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

PARAMS

newNumberOfLicences	20 (Required) The new required number of licences for the account
partnerCode	125478sfg (Required) Enter the unique partner code assigned to the partner to link this account with the specified partner

GET Change Plan

https://na1.foxitesign.foxit.com/api/accounts/changeplan?newNumberOfLicenses=20&newPlan=Professional

This endpoint allows you to change the plan type for a specific account.

Note: please provide the `access_token` Authorization header from the account which you are looking to change the plan type for.

AUTHORIZATION

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Accept	application/json

PARAMS

newNumberOfLicenses	20 (Required) The new required number of licences for the account in the new plan
newPlan	Professional (Required) Name of plan that you want to subscribe to

POST Generate OAuth2.0 Access Token

https://na1.foxitesign.foxit.com/api/oauth2/access_token

Foxit eSign is an oAuth2 based authorization provider. Security implementation is based on the final version of "The OAuth 2.0 Authorization Framework".

HEADERS

Content-Type	application/x-www-form-urlencoded

Body urlencoded

client_id	YOUR_ESIGN_CLIENT_ID (Required)
client_secret	YOUR_ESIGN_CLIENT_SECRET (Required)

grant_type	client_credentials (Required)
scope	read-write (Required)

POST Regenerate Embedded Signing Session

https://na1.foxitesign.foxit.com/api/embedded/regenerateEmbeddedSigningSession

You can regenerate the expired folder embedded signing token for accessing the signature from the signer.

AUTHORIZATION OAuth 2.0

This request is using OAuth 2.0 from folder eSign APIs

HEADERS

Content-Type	application/json
Accept	application/json
Authorization	Bearer <token>

Body raw (json)

json

```
{
  "folderId": 16501764,
  "emailId": "john.doe@example.com",
  "partyId": "2",
  "sessionExpire": "false",
  "expiry": 30000
}
```

PDF Embed API

Embed PDF Viewer

📄 PDF Embed Viewer

Kickstart your development with Foxit's easy-to-integrate PDF Embed Viewer. This guide walks you through embedding a fully customizable PDF viewer directly into your web page.

After obtaining your Client ID and Client Secret from the Foxit Developer Console, follow the steps below to render a PDF using the embedded viewer.

🧩 Integration Steps

1. Include the SDK Script

Add the script tag in your HTML to load the Foxit Embed Viewer:

```javascript
<script src="https://embed.developer-api
```

Replace `YOUR_CLIENT_ID` with the actual Client ID you received from the Foxit Developer Console.

2. Define the Viewer Container

Set up a container in your HTML where the viewer will be rendered by using a `div` tag with the ID foxit-embed-view.

3. Initialize the Viewer

Use the following JavaScript snippet to configure and launch the viewer:

```html
<script>
  new FoxitEmbed.View({
    clientId: 'YOUR_CLIENT_ID',
    divId: 'foxit-embed-view'
  }).previewFile(
    {
      content: 'https://yourdomain.com/s
      metaData: { fileName: 'sample.pdf'
    },
    {
      showToolControls: true,
```

📦 PDF File Object Structure

- `content`: File URL or binary content that will be rendered (can be a string or ArrayBuffer).
- `metaData`: Contains key metadata like the fil name.

🧪 Run the Viewer

Load your HTML file in a browser to see the embedded PDF viewer in action.

> 💡 Tip: Use a local server (e.g., with Node.js, Python, or VS Code Live Server) to avoid cross-origin issues.

Example Code

javascript

```
<html lang="en">
<head>
    <meta charset="UTF-8">
              content="width=device-widt
    <meta http-equiv=X-UA-Compatible cor
    <meta name="renderer" content="webki
    <title>Foxit Embed Viewer - In Line<
    <style>
        html, body{
            width: 100%;
            height: 100%;
```

🌐 Cross-Origin Resource Sharing (CORS)

When using the **Foxit PDF Embed API**, CORS issues can occur if you're loading PDF files via URL. The viewer attempts to fetch the file from the provided URL, and if the request violates browser CORS policies, it will fail.

⚠️ Why CORS Matters

Modern browsers enforce **CORS** to protect user data and application integrity. If your PDF file is hosted on different domain than your web page, the request ma be blocked unless CORS is explicitly allowed by the server hosting the PDF.

✅ Recommended Solutions

To avoid or resolve CORS-related issues, you can:

`Option 1`: Host on the Same Domain

Ensure both your **web page** and the **PDF file** are served from the same domain and protocol (e.g., `https://yourdomain.com`). This removes the cross-origin restriction entirely.

`Option 2`: Set Proper CORS Headers

Configure the server hosting the PDF to include appropriate **CORS headers**, such as:

```http
Access-Control-Allow-Origin: https://you
Access-Control-Allow-Methods: GET
Access-Control-Allow-Headers: Content-Ty
# Embed Modes
The Foxit Embedded Viewer provides four
By default, the viewer uses the `FULL_WI
``` javascript
var pdfUrl = 'https://embed.developer-ap
embedView.previewFile(
  {
    content: pdfUrl,
```

Embedded mode overview

EMBED MODE	DESCRIPTION	EXAMPLE
Full Window (FULL_WINDOW)	Default mode. The PDF reader is displayed in the full screen of the browser window, so that it is convenient for users to read and operate documents	Here

UI Customization

UI Customization

The Foxit Embedded Viewer API provides a flexible set of options for tailoring the PDF viewing experience to
```

of options for tailoring the PDF viewing experience to match your application's needs and branding. You can configure various interface elements and interactions, including:

- **Toolbar Controls** – Show or hide the top toolbar and its tools.
- **Navigation Panel** – Enable or disable the left-hand pane for page thumbnails or bookmarks.
- **Download & Print** – Allow or restrict PDF download and print capabilities.
- **Viewer Appearance** – Customize the layout and color scheme for optimal presentation.
- **Branding** – Add your company logo and adjust theme colors to reflect your brand identity.

Tool Options

You can fine-tune the behavior and appearance of the PDF viewer using the customization options provided by the Foxit PDF Embed API. These configurations allow you to deliver a tailored user experience while maintaining consistency with your application's design and brand identity. Whether it's modifying tool visibility, adjusting layout settings, or applying custom branding, the API provides the flexibility needed to match your specific requirements.

Variable	Default	Description
showToolControls	True	Show tool bar or not
showLeftHandPanel	True	Show the left panel or not
defaultViewMode	Null	`FIT_WIDTH`: Extend the page horizontally to the full width of the document pane. `FIT_PAGE`: Display the entire page in the current

		view pane.In addition, there are two other view

Callbacks and Workflows

The Foxit PDF Embed API supports advanced workflow customization through event callbacks. By registering specific callbacks, developers can respond to user

interactions and events such as document loading, viewer state changes, text selection, and more.

To register a callback, use the addListener method on the FoxitEmbedViewer object. Here's the syntax:

javascript

```
FoxitEmbedViewer.addListener(
  "<CallbackType>",   // Specify the event
  callbackFunction,   // Define the functi
  options             // (Optional) Pass a
);
```

This functionality enables seamless integration into custom workflows and improves interactivity within your application.

Callback Type	Callback function arguments	Description
File_Opened	True	Triggered after that the document is displayed
Viewer_Zoomed	newScale, oldScale	Triggered when the zoom page is done
Viewmode_Changed	NewViewModeType, OldViewModeType	Triggered after changing the view mode
Text_Selected	text content	Triggered after selecting tex
PageNum_Changed	newPageNumber	Triggered when the

Viewer APIs

Bookmark

getBookmarks

The API returns the list of the existing PDF bookmarks. Each bookmark item in this list is shown as a JSON containing important information like ID, title and the list of the nested bookmarks residing under this bookmark.

Input parameters

```json
{
  id: 'some_uniq_id' // Unique identifier
  title: 'some_title', // Title of the boo
  children: [CHILD_BOOKMARK_1, CHILD_BOOKM
}
```

API output

Returns a Promise

- Resolves with the list of bookmarks available in the PDF. `[ Bookmark_1, Bookmark_2, ...]`

openBookmark

The API accepts a bookmark ID as input and navigates to the particular PDF bookmark.

Input parameters

Parameters: .

API output

Returns a Promise

- Resolves to true on if API is successfully done. And the user navigates to that particular bookmark.
- Resolves to false when the bookmark does not exist in the PDF.

Search

search

These APIs can be used to search a term in the PDF programmatically. And they are supported in all the embed modes.

Input parameters

- keywords: The text will be searched
- startPageIndex , endPageIndex: Search range

from "startPageIndex" to "endPageIndex"

- matchRule: `WholeWordsOnly | CaseSensitive`

API output

Returns a Promise

- Resolves with a JSON object containing the following interfaces: `onResultsUpdate()`, `next()`, `previous()`, `clear()`

onResultsUpdate

Users can register a callback function which will be passed to the `onResultsUpdate()` function as an input.

This callback function will be triggered every time when a search operation is done by the search API and the search result is highlighted in the PDF. The callback function will receive the important information about the current search result in the form of a JSON.

The JSON will include information like the current page number, the current search result index, the total number of the search results and status.

JSON information like:

```json
{
  "currentResult": {
    "pageNumber": Integer,
    // Current page number in the view
    "index": Integer
    // Index of the current highlighted se
  },
  "totalResults": Integer,
  // Total number of search results found
  "status": String
  // Status of search result till the time
```

API output

Returns: Returns a Promise

- Resolves to true on the successful operation. When operation is successful, the callback function is executed and receives the

information about the search results in the form of a JSON.

- Resolves to false when `onResultsUpdate()` operation fails.

onResultsUpdate Example

```javascript
function callbackFunction(searchResult) {
  console.log('search result: ', searchRes
}
FoxitEmbedViewer.search(&#x27;<KEY_STRING>
  searchObj.onResultsUpdate(callbackFunct
  // call searchObj.next() or searchObj.pr
  searchObj.next();
});
```

next

This function will highlight and navigate to the next search result in the PDF.

API output

Returns a Promise

- Resolves to true on the successful operation and the next search result is highlighted in the PDF.
- Resolves to false when `next()` operation fails.

previous

This function will highlight and navigate to the previous search result in the PDF.

API output

Returns a Promise

- Resolves to true on the successful operation. And, the previous search result is highlighted in the PDF.
- Resolves to false when `previous()` operation fails.

clear

This function will cancel the ongoing search operation and clear the search results.

Zoom

zoomTo

Zoom the current view to the given scale. It can't be used in the inline mode.

Input parameters

Parameters: `number|string` scale - Greater than zero. If it's a string, only 'fitWidth' and 'fitHeight' are valid.

API output

Returns: `Promise`

getPageZoom

The API takes the PDF page number as input and returns the zoom level of that page.

Input parameters

Parameters :

API output

Returns a Promise

- Resolves with the zoom scale if it is successful.

Others

getSelectedContent

If a user selects any content in the viewer, then the selected content can be fetched by the API. The API currently only supports the text selection.

API output

Returns a Promise

- Resolves with a JSON object including the type of content and actual selected content if it is successful: `{ type: "", data: "" }`

getCurrentPage

The API returns the current page number of the current page.

API output

Returns a Promise

- Resolves with the current page number if it is successful.

gotoLocation

The API enables navigation to any PDF page. It accepts a page number as input. You can also pass the x and y coordinates on the page as the optional input parameters to navigate to a particular location on the page. If don't input any coordinates, the default coordinates is (0, 0).

Input parameters

parameters: (, ,)

User Preferences

Users can set their own preferences. The UpdatePreferences API allows users to update the embed viewer preferences.

Input parameters

preferences

```json
{
  "be_markup_with_selected_text": true
}
```

input parameters data description

PARAMETER	DESCRIPTIO N	REQUIRED

be_markup_with_selected_text	Whether to use the selected text as the markup content after adding a text markup annotation.	YES

API output

Returns a Promise:

- Resolves on the API operation success.
- Rejects with error object including an error code and message.

```javascript
preferences = {};
embedView
.UpdatePreference(preferences)
.then(() => console.log('Success'))
.catch((error) => console.log(error));
```

Comments and Markups

Comments and Markup

The Foxit PDF Embed API enables users to annotate PDF documents with a variety of commenting and markup tools. Supported actions include adding text comments, sticky notes, highlights, underlines,

strikethroughs, and freehand drawings. Users can also erase portions of drawing annotations. Undo and redo controls are conveniently located in the top toolbar.

Commenting Features Overview

The API supports the following commenting capabilities:

- Add, update, or delete annotations (e.g., text comments, highlights, drawings).
- Reply to annotations allowing collaborators to engage in threaded discussions.
- View all comments in the left-hand comments panel.
- Any annotation activity (add/update/reply) activates the Save button.
- Once saved, annotations become part of the PDF buffer, preserving them in the document.

These features provide an interactive, collaborative experience for PDF document handling directly within your web application.

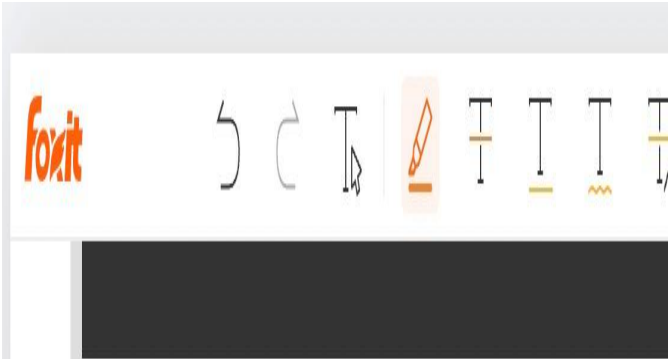
Text Markup Anotations

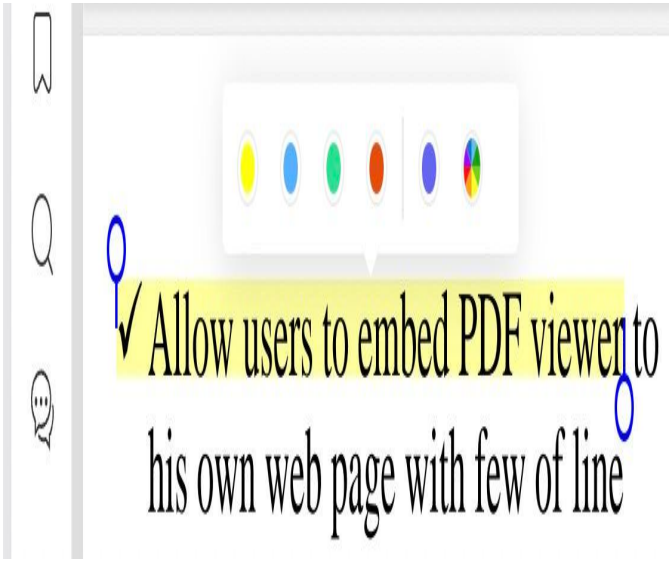
Text markup annotations include highlight text , strikeout text , underline text , squiggly text , replace text and caret text.

Adding a text markup:

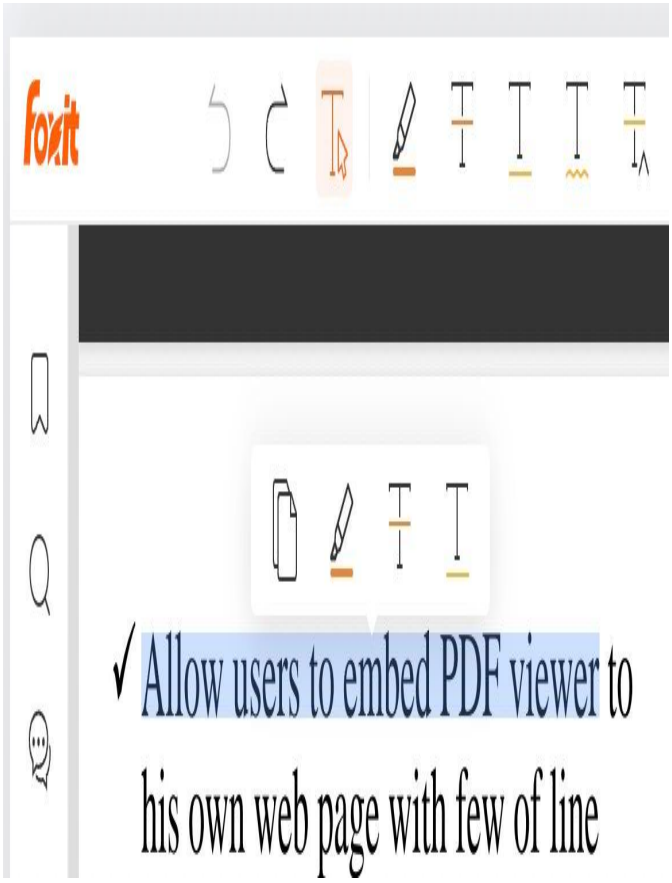
There are two ways to add a text markup:

- Click the highlight tool in the top bar and select some text. The text gets highlighted





- Using the select tool to select some text, and then choose either the highlight, strikethrough or underline options in the popover menu.



Updating a text markup

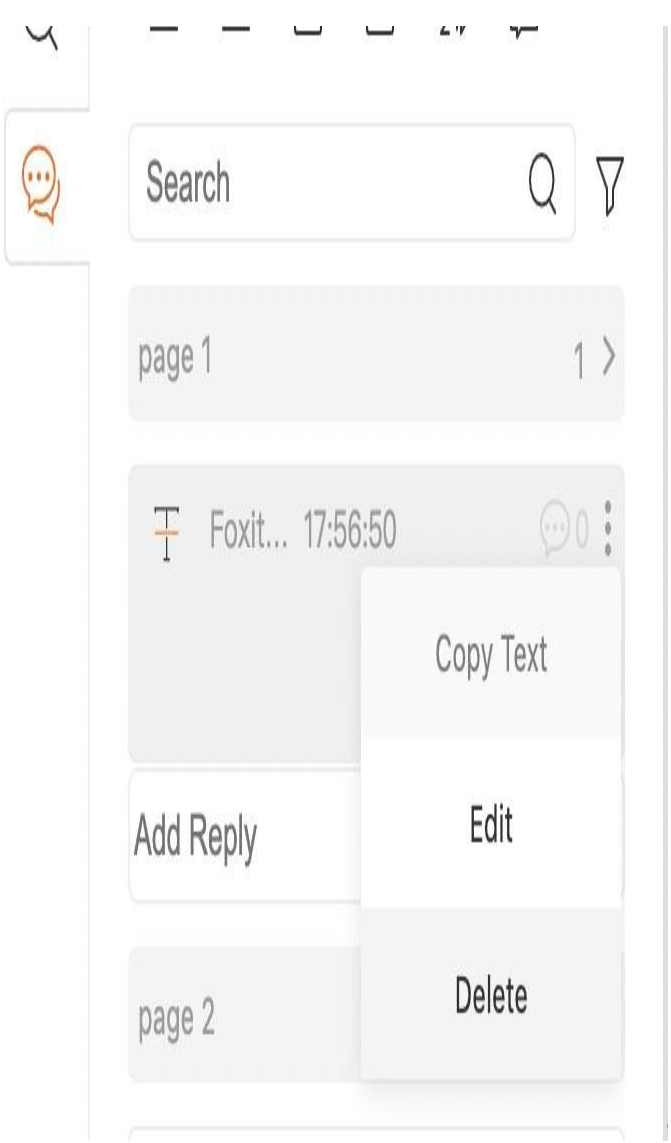
While focusing on the text markup , you can:

- Click on the existing text markup to open a popover menu to change the color.
- Moving the 'text boundary cursor' to select more text for commenting.

- Double click on the text markup will open the comments panel for editing comment.

Removing a text markup

- Click on the existing text markup to open the overflow menu, and choose **Delete**.
- Click on the overflow menu for the comment in the comments panel, and choose **Delete**.

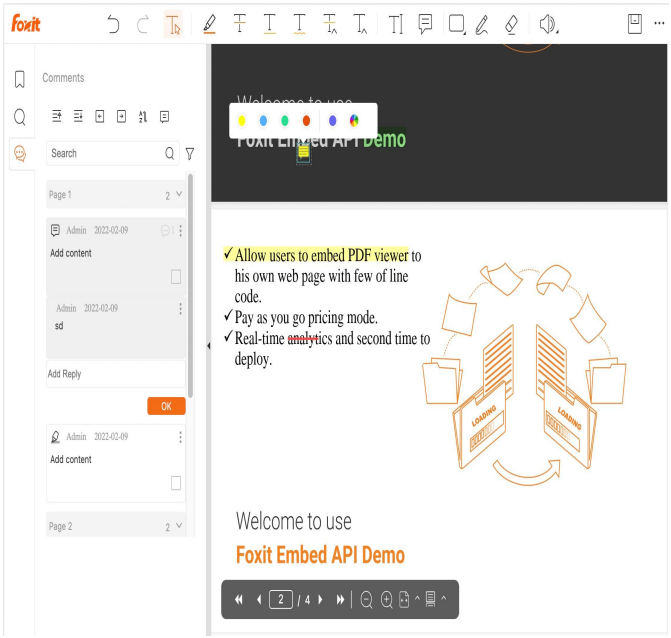


Note

Using Note to add text reply. And users can put it anywhere on the PDF document. The features supported are as following:

- Adding a note
  - Select the note tool in the top toolbar and click where you want to add the note. You can input the reply message into the left panel's comment box.
- Editing a note
  - Click on the existing note to open a popover menu to change the properties of the note.
- Delete a note
  - Click on the existing note to open the menu and click on the delete option to remove the note.

- Moving
    - Choosing the Selecting tool and placing the cursor over the existing note and dragging to move it to the other location.



Typewriter

Using typewriter tool to add text anywhere in the PDF document.

- Adding text
    - Select the Typewriter tool in the top toolbar, click on the page to place where you want for adding the typewrite and inputting the text. Click outside the text box to add the annotation.
- Updating text
    - Click on the existing Typewriter annotation to edit the text or change any of the text attributes in the properties box. You can change the attributes such as the color and font size of the text.
- Deleting text
    - Click the existing typewriter annotation to open the toolbar, and choose **Delete**.
- Moving
    - Choosing the selecting tool and placing the cursor over the existing typewriter annotation and dragging to move it to the other location.

Pencil & Eraser

Use Pencil tool to create any freehand drawing or shape, and using the eraser tool to erase the parts of the freehand drawing annotation.

- Adding a drawing
    - Select the Pencil tool in the top bar, click where you want to begin drawing and drag to create a shape. You can release the mouse button, move the pointer to a new location, and continue drawing.
- Updating a drawing
    - Click on the existing drawing. You can change the attributes such as the color and the stroke width.
    - You can edit the comment by clicking on the Edit option in the overflow menu.
- Removing a drawing
    - Click on the existing drawing to display the menu and click on the delete option.
    - Use the Eraser tool to erase the parts of the freehand drawing.
- Moving or resizing a drawing
    - Choose the selecting tool and place the cursor over a drawing and drag to move it to a different location.
    - Choose the selecting tool and place the cursor over a drawing and drag one of the corner circles to resize the annotation.

Undo or Redo changes

Use the Undo tool to reverse the previous actions, you can find it in the top toolbar. Each clicking the Undo command will reverse the action, repeatedly clicking

the Undo command will go back the previous actions.

Use the Redo tool to reverse the action, you can find it in the top toolbar next to the Undo button.

Annotations APIs

The annotations APIs support programmatic adding,

deleting, updating, importing and exporting both comments and other types of text markup annotations, such as highlight and underlines. The **enableAnnotations** will be used to enable and control PDF annotations:

- enableAnnotations: Default is true. The true option will enable the Annotation APIs and the viewer will display the existing annotations.The client's application passes the annotations and the PDF buffer to The Foxit PDF Embed API which renders it in the browser.

Annotation Schema

Annotation data sent from the API should be as the JSON format, which allows users to identify the properties of annotations.

The JSON format of The Foxit PDF Embed API's annotation is as following.

```json
{
"source": | ,
"type": "Annotation",
"subtype": "note" | "strikeout" | "highlig
"underline" | "squiggly" | "replace" | "caret"
"id": < ANNOTATION_ID > ,
```

```
"contentValue": String,
"motivation": String,
"index": < PAGE_INDEX >
"boundingBox": \[Xmin, Ymin, Xmax, Ymax\]
"quadPoints": \[...\],
```

Annotations Data Parameters

PARAMETER	DESCRIPTION	REQUIRED
type	The annotation type which must be *Annotation*.	YES
id	The annotation's unique identifier.	YES
contentValue	The string value of the plain text comment.	YES
motivation	The reason for its creation. The value is either **commenting** or **replying**.	YES

Annotation Data Examples

Add text annotation data

json

```
{
  "source": "77c6fa5d-6d74-4104-8349-657c8
  "type": "Annotation",
  "subtype": "typewrite",
  "id": "02dcf931-d1cb-49c1-a8bc-d047892a0
  "contentValue": "I added a text annotat
```

```
  "motivation": "commenting",
  "index": 0,
  "fontStyle": {
    "size": "24px",
    "color": "#0000FF"
```

Reply annotation data

```json
{
  "type": "Annotation",
  "id": "eb46d1a9-e9c3-4e81-a6f4-ce5ba7a9(
  "contentValue": "Reply to typewrite",
  "motivation": "replying",
  "source": "02dcf931-d1cb-49c1-a8bc-d047
  "creator": {
    "type": "Person",
    "name": "Samuel"
  },
  "created": "2022-02-02T14:45:37Z",
```

The Foxit PDF Embed API allows users to programmatically add, import, export, delete, and update annotations.

When users enabled the annotations, users can invoke all the annotations APIs with the **AnnotationManager.** The example is as following:

```html
<html lang="en">
  <head>
    <meta charset="UTF-8">
              content="width=device-width,
    <meta http-equiv= X-UA-Compatible cont
    <meta name="renderer" content="webkit"
```

```
    <title>Foxit Embed Viewer - Full Windo
    <style>
      html, body{
      width: 100%;
      height: 100%;
```

Basic APIs for Commenting

The PDF Embed API offers several APIs which allow you to programmatically add, import, export, delete, and update annotations.

Adding Annotations

The addAnnotations API allow users to add the supported annotations. Either a single annotation or multiple annotations can be added.The annotation data must be specified in the Annotation schema.

input parameters
A JSON array specified in the Annotation schema, contains the list of annotations :[Annotation_1, Annotation_2, ...]

API Output
Returns a Promise:

- Resolves on a successful API operation.
- Rejects with error object including code and message on the API failure.

javascript

```
const list_of_annotations = [Annotation_1
  embedView.getAnnotationManager().then(ar
  annotationManager.addAnnotations(list_o
  .then(() => console.log("Success"))
  .catch(error => console.log(error));
  });
```

Getting Annotations

The getAnnotations API is able to receive the exist PDF annotations from the PDF file. The data returned is in the form of a JSON array. And the data format is described in the Annotation schema.

Input parameters

A filter is provided by the client

- If set a list of annotation IDs in the filter, the annotations matched with this IDs will be returned.
- If set a page range of the PDF file, the annotations in these pages will be returned. The first page number is 0.
- If no filter, the data returned will include all the annotations in the PDF file.

```javascript
filter = {
  annotationIds:
  [Annotation_ID_1, Annotation_ID_2, ...],
  pageRange:{
  startPage: \<Page_Number>,
  endPage: \<Page_Number>
  },
  }
```

API output

Returns a Promise:

- Resolves with a list of annotations on success,
- Rejects with an error object which includes a code and message on the failure.

```javascript
const filter = {
  // annotationIds: [Annotation_ID_1, Anno
  // OR,
  // pageRange: {startPage: \<Page_Number>
  }
  embedView.getAnnotationManager().then(an
```

```
  annotationManager.getAnnotations(filter)
  .then(() => console.log("Success"))
  .catch(error => console.log(error));
  });
```

Deleting Annotations

The deleteAnnotations API is able to delete the PDF annotations in the PDF file.

Input parameters

A filter is provided by the client.

- If set a list of annotation IDs in the filter, the annotations matched with this IDs will be deleted.
- If set a page range of the PDF file, the annotations in these pages will be deleted. First page number is 0.
- If no filter, all the annotations in the PDF file will be deleted.

javascript

```
filter = {
  annotationIds:
  [Annotation_ID_1, Annotation_ID_2, ...]
  pageRange:{
    startPage: \<Page_Number>,
    endPage: \<Page_Number>
    },
  }
```

API output

Returns a Promise:

- Resolves on the delete operation success.
- Rejects with error object including code and message on the delete operation failure.

javascript

```
const filter = {
  // annotationIds: [Annotation_ID_1, Anno
  // OR,
  // pageRange: {startPage: \<Page_Number>
```

```
}
embedView.getAnnotationManager().then(a
  annotationManager.deleteAnnotations(f
  .then(() => console.log("Success"))
  .catch(error => console.log(error));
});
```

Updating Annotations

The updateAnnotation API updates a single existing annotations in the PDF. The API takes the updated annotations data ,finds the annotations with the ID in the input data, and applies the update to that annotation.

Input parameters

JSON object containing the annotation data, must be in the same format as the format described in the Annotation schema.

API output

Returns a Promise:

- Resolves on the update operation success.
- Rejects with error object including code and message on the update operation failure.

```javascript
const annotation_data = \<Annotation_Data
  embedView.getAnnotationManager().then(a
  annotationManager.updateAnnotation(anno
  .then(() => console.log("Success"))
  .catch(error => console.log(error));
  });
```

Selecting Annotation

The selectAnnotation API selects any existing annotation and give the focus to the selected annotation.

Input parameters

API output

Returns a Promise:

- Resolves on the select annotation operation success.
- Rejects with error object including an error code and message

```javascript
embedView.getAnnotationManager().then(anno
  annotationManager.selectAnnotations(anno
  .then(() => console.log("Success"))
  .catch(error => console.log(error));
  });
```

Unselecting Annotation

The unselectAnnotation API unselects the last selected annotation.

```javascript
embedView.getAnnotationManager().then(anno
    annotationManager.unselectAnnotation()
    });
```

Importing/Exporting Annotations by fdf

The exportAnnotationstoFdf API allows users to export all the supported annotations from the PDF file to a specific format file.

Input parameters

`fileType: 'xfdf' | 'fdf'` - Specify data file type. The default is 'fdf'. `annots: arrays` - Specify the particular annotation id to be exported. If it's null, all the annotations will be exported.

API output

Returns a Promise:

- Resolves on success with blob of annotations data.
- Rejects with error object including an error code and message.

```javascript
```

```javascript
embedView.getAnnotationManager().then(
  annotationManager => {
annotationManager.exportAnnotationstoFdf()
  .then(annotData => {
  //Save Date to the file storage;
  })
.catch(error => console.log(error));
```

The `importAnnotationsfromFdf` API allows users to import all the supported annotations from a fdf/xfdf format file.

Input parameters

Specify fdf file's stream.

```javascript
data: {File\|Blob\|ArrayBuffer\|TypeArray
```

API output

Returns a Promise:

- Resolves on the import operation success.
- Rejects with error object including an error code and message

```javascript
const fdfData //Get fdf file's stream;
embedView.getAnnotationManager().then(anno
  annotationManager.importAnnotationsfromF
  .then(() => console.log("Success"))
  .catch(error => console.log(error));
});
```

Annotation Events

Developers can receive the events when a user action interacts with an annotation. These events are generated for the annotation actions performed by the UI and the annotation APIs. Using addListener to register callback function to the Embedded Viewer.

Event Type	Description	Param
Annotation C	An existing	Annotation id

licked	annotation is clicked.	
Annotation_Updated	An existing annotation is updated.	Annotation id
Annotation_Added	A new annotation is added .	Annotation id
Annotation_Deleted	An existing annotation is deleted.	Annotation id
Annotation_Activated	Any existing annotation is selected.	Annotation id
Annotation_Unactivated	Any existing annotation is unselected.	Annotation id

Add Event

For more detail see Callbacks & Workflows.

```javascript
FoxitEmbedViewer.addListener(
  <Callback Type>, // Replace with the typ
  <Function>,   // Replace with the functi
  options     // Optional: Additional opti
);
```

Read Aloud

Read Aloud

The Foxit PDF Embed API provides the way to read aloud the PDF text contents.

Activate the "Read Aloud" module

The activate API allows users to activate the "read aloud" module. It will read aloud the selected text automatically if it is activated.

automatically if it is activated.

API output

Returns a Promise which:

- Resolves on the API operation success.
- Rejects with error object including an error code and message.

```javascript
embedView.GetReadAloud().then((readAloud)
  readAloud
    .activate()
    .then(() => console.log('Success'))
    .catch((error) => console.log(error));
});
```

Deactivate the "Read Aloud" module

The deactivate API allows users to deactivate the "read aloud" module.

API output

Returns a Promise :

- Resolves on the API operation success.
- Rejects with error object including an error code and message.

```javascript
embedView.GetReadAloud().then((readAloud)
  readAloud
    .deactivate()
    .then(() => console.log('Success'))
    .catch((error) => console.log(error));
});
```

Start to Read Aloud

The start API allows users to read aloud the pages as the page ranges if the "read aloud" module is activated.

input parameters

```json
{
  "start_page_index": startPageIndex,
  "end_page_index": endPageIndex,
}
```

Input parameters data description

PARAMETER	DESCRIPTION	REQUIRED
start_page_index	The start page index.	YES
end_page_index	The end page index.	YES

API output

Returns a Promise:

- Resolves on the API operation success.
- Rejects with error object including an error code and message.

```javascript
data = {};
embedView.GetReadAloud().then(readAloud=>
readAloud.readAloudPages(data).then(() =>
.catch(error => console.log(error));
})
```

Stop to Read Aloud pages

The stop API allows users to stop reading aloud.

API output

Returns a Promise:

- Resolves on the API operation success.
- Rejects with error object including an error code and message.

```javascript
embedView.GetReadAloud().then(readAloud=>
readAloud.stop().then(() => console.log("
.catch(error => console.log(error));
})
```

pause

The pause API allows users to pause reading aloud.

API output

Returns a Promise:

- Resolves on the API operation success.
- Rejects with error object including an error code and message.

```javascript
embedView.GetReadAloud().then(readAloud=>
readAloud.pause().then(() => console.log('
.catch(error => console.log(error));
})
```

resume

The resume API allows users to resume reading aloud.

API output

Returns a Promise:

- Resolves on the API operation success.
- Rejects with error object including an error code and message.

```javascript
embedView.GetReadAloud().then(readAloud=>
readAloud.resume().then(() => console.log
.catch(error => console.log(error));
})
```

update Read Aloud option

The updateOption API allows users to update the read

aloud option, includes rate and volume.

input parameters

```json
option
{
"rate": Float,
"volume": Float,
}
```

input parameters data description

PARAMETER	DESCRIPTION	REQUIRED
rate	The rate of reading aloud. It is float for the rate value. It can range between 0.1 (lowest) and 10 (highest), while 1 is a normal speaking rate.	YES
volume	A float for the volume value, between 0 (lowest) and 1 (highest.)	YES

API output

Returns a Promise:

- Resolves on the API operation success.
- Rejects with error object including an error code

and message.

```javascript
option = {};
embedView.GetReadAloud().then(readAloud=>{
readAloud.updateOption(option).then(() =>
.catch(error => console.log(error));
})
```

Form Handling

Form Handling

The Foxit PDF Embed API supports interactive form filling out-of-the-box. Users can seamlessly fill in text fields, checkboxes, radio buttons, dropdowns, and list selections within the embedded PDF viewer. Once any field is edited, the Save button in the top toolbar becomes active, allowing users to save their input directly into the PDF.

> ⚠️ Limitations
> The following form features are not supported:

- Creating, editing, or deleting form fields
- XFA-based forms
- Digital signature and barcode fields
- File picker text fields
- Rich Text Format (RTF) text fields
- JavaScript-based validation, calculation, or actions
- Special/custom formats in text fields or dropdowns
- Submit buttons with PDF actions (view-only supported)

Comments and Markup

The Foxit PDF Embed API includes commenting features. Users can add annotations using text comments, sticky notes, highlights, and drawing tools. An eraser tool is available for removing parts of drawings, and the top toolbar provides undo and redo capabilities.

Class View

Class: View

This is the main class for the PDF Viewer. It provides methods to render the PDF Viewer and interact with it.

Constructors

new View(params)

new View(`params`): `View`

Constructor for the PDF Viewer

Parameters

• **params**

• **params.clientId**: `string`

the client id

• **params.divId**: `string`

the div id to render the viewer

• **params.locale?**: `Locale`

the locale for the viewer

Returns

`View`

Methods

GetReadAloud()

GetReadAloud(): `Promise` < `ReadAloudManager` >

Get read aloud manager for the viewer.

The Embed API provides a read aloud feature that

allows you to read the text of a PDF file. This method helps you to get the ReadAloudManager instance to interact with the read aloud feature.

Returns

`Promise` < `ReadAloudManager` >

- Return a promise that resolves with the ReadAloudManager instance, or rejects with an error.

UpdatePreference()

UpdatePreference(`preferences`):
`Promise` < `void` >

Update the preference of the viewer.

Parameters

• **preferences**: `Preferences`

the preferences for the viewer

Returns

`Promise` < `void` >

- Return a promise that resolves when the preference is updated, or rejects with an error if the preference cannot be updated.

addListener()

addListener(`event` , `callback`):
`Promise` < `any` >

Add an event listener to the viewer.

Parameters

• **event**: `"File_Opened"` | `"Viewer_Zoomed"` | `"Viewmode_Changed"` | `"Text_Selected"` | `"PageNum_Changed"` | `"Viewer_Rotated"` | `"Annotation_Clicked"` | `"Annotation_Activated"` | `"Annotation_Unactivated"` | `"Annotation_Updated"` | `"Annotation_Added"` | `"Annotation_Deleted"`

the event to listen to

• **callback**: `Function`

the callback function to be called when the event is triggered

Returns

`Promise`<`any`>

- Return a promise that resolves when the event listener is added, or rejects with an error.

createPDFViewer()

createPDFViewer(): `Promise`<`any`>

Returns

`Promise`<`any`>

downloadCurrentPDF()

downloadCurrentPDF(`fileName`): `Promise`<`boolean`>

Download the current PDF file opened in the viewer.

Returns

`Promise`<`boolean`>

- Return a promise that resolves with true when the PDF file is downloaded, or rejects with an error.

getAnnotationManager()

getAnnotationManager():
`Promise` < `AnnotationManager` >

Get annotation manager for the viewer.

This method helps you to get the AnnotationManager instance to interact with the annotation feature.

Returns

`Promise` < `AnnotationManager` >

- Return a promise that resolves with the AnnotationManager instance, or rejects with an error.

getBookmarks()

getBookmarks(): `Promise` < `Bookmark` []>

Get the bookmarks in the PDF file.

Returns

`Promise` < `Bookmark` []>

- Return a promise that resolves with the bookmarks, or rejects with an error.

getCurrentPage()

getCurrentPage(): `Promise` < `number` >

Get the current page number in the viewer. page number starts from 1.

Returns

`Promise` < `number` >

- Return a promise that resolves with the current page number, or rejects with an error.

getPageZoom()

getPageZoom(`pageNumber`): `Promise` < `number` >

Get the current zoom scale of a specific page.

Parameters

• **pageNumber**: `number`

the page number to get the zoom scale, starting from 1.

Returns

`Promise` < `number` >

- Return a promise that resolves with the zoom scale, or rejects with an error.

gotoLocation()

gotoLocation(`pageNumber` , `xCoordinate` , `yCoordinate`): `Promise` < `boolean` >

Goto specific location in the PDF file.

Parameters

• **pageNumber**: `number`

the page number to go to, starting from 1. default is 1

• **xCoordinate**: `number` = `0`

the x coordinate to go to, default is 0

• **yCoordinate**: `number` = `0`

the y coordinate to go to, default is 0

Returns

`Promise` < `boolean` >

openBookmark()

openBookmark(`bookmarkId`): `Promise` < `boolean` >

Jump to the bookmark by id.

Parameters

• **bookmarkId**: `number`

the bookmark id to jump to. It is the id of the bookmark returned by getBookmarks method.

Returns

`Promise` < `boolean` >

- Return a promise that resolves with true when the bookmark is opened, or rejects with an error.

previewFile()

previewFile(`params` , `options` ?): `Promise` < `void` >

Open a PDF file in the viewer

Parameters

• **params**: `Params`

the params for the viewer

• **options?**

the options for customizing the viewer

• **options.defaultViewColorStyle?**: `string`

• **options.defaultViewMode?**: `"FIT_WIDTH"` | `"FIT_PAGE"`

• **options.deviceType?**: `string`

• **options.embedMode?**: `EmbedMode`

the embed mode for the viewer

• **options.enableTextSelection?**: `boolean`

• **options.showDownloadPDF?**: `boolean`

• **options.showIcon?**: `string`

• **options.showLeftHandPanel?**: `boolean`

• **options.showPrintPDF?**: `boolean`

• **options.showSelectTool?**: `boolean`

• **options.showSnapshotTool?**: `boolean`

• **options.showToolControls?**: `boolean`

Returns

`Promise<void>`

search()

search(`keywords`, `startPageNumber`, `endPageNumber`, `matchRule`): `Promise<SearchAPI>`

Search the keywords in the PDF file.

Parameters

• **keywords**: `string`

the keywords to search.

• **startPageNumber**: `number`

the start page number to search, starting from 1.

• **endPageNumber**: `number`

the end page number to search, starting from 1.

• **matchRule**: `"WholeWordsOnly"` | `"CaseSensitive"`

the match rule for the search.

Returns

`Promise<SearchAPI>`

- Return a promise that resolves with the Search object, or rejects with an error.

setCustomLogo()

setCustomLogo(`logo`): `void`

Set the custom logo for the viewer.

Parameters

• **logo**

the logo url, must be a valid url that can be displayed by the img tag.

• **logo.url**: `string`