Общие сведения

Документация по интеграционному API iDocs

Данная документация предназначена для технических специалистов компаний, планирующих интегрировать свои информационные системы с API iDocs для автоматизации процессов электронного документооборота (ЭДО) через REST HTTP запросы.

Основные принципы интеграции:

- **RESTful API:** Взаимодействие с API iDocs осуществляется посредством стандартных HTTP методов (GET, POST, PUT, DELETE) и кодов ответов.
- **JSON:** Формат данных для запросов и ответов - JSON.
- **Аутентификация:** Для доступа к API требуется аутентификация. Детали аутентификации будут предоставлены компанией iDocs. Как правило, используется токен авторизации (Bearer token) в заголовке `Authorization`.
- **Безопасность:** Рекомендуется использовать HTTPS для обеспечения безопасной передачи данных.

Рекомендации по использованию Swagger:

- **Просмотр API:** Используйте онлайн Swagger UI (например, REST API Documentation Tool | Swagger UI) или установите локально для удобного просмотра структуры API. Просто вставьте ссылку на swagger.json.
- **Исследование эндпоинтов:** Изучите доступные эндпоинты, их HTTP методы, URI и описание.
- **Параметры запросов:** Обратите внимание на обязательные и необязательные параметры запросов, их типы данных и форматы.
- **Тело запроса (Request Body):** Для методов POST и PUT изучите структуру JSON тела запроса и требования к данным.
- **Ответы API (Responses):** Ознакомьтесь с возможными кодами HTTP ответов и структурой JSON тела ответа для каждого эндпоинта. Это поможет вам правильно обрабатывать результаты запросов.
- **Примеры:** Swagger specification часто содержит примеры запросов и ответов, которые могут быть полезны для понимания работы API.

API

Swagger документация:

Полное описание API, включая все доступные эндпоинты, параметры, модели данных и коды ответов, доступно по следующей ссылке:

https://ext-edo-integration.idocs.kz/swagger/index.html

Тестовый API и его описание: idocs-integration-v3

Виды ответов:

Виды ответов

HTTP код 200 (Success):

- **Описание:** Успешная загрузка. Возвращает информацию о загруженном blob-объекте, включая его идентификатор.
- **Content-Type:** `text/plain`, `application/json`, `text/json`
- **Схема:** `#/components/schemas/ClientUploadedBlobVMSyncOperationResponseVM` - Объект, содержащий информацию о загруженном blob-объекте (например, `blobId`). Детальное описание структуры смотрите в разделе "Схемы".
- **HTTP код 400 (Bad Request):**
  - **Описание:** Некорректный запрос. Возможные причины: неверный формат файла, превышение размера файла.
  - **Content-Type:** `text/plain`, `application/json`, `text/json`
  - **Схема:** `#/components/schemas/ClientUploadedBlobVMSyncOperationResponseVM` - Объект, содержащий информацию об ошибке. Детальное описание структуры смотрите в разделе "Схемы".
- **HTTP код 401 (Unauthorized):**
  - **Описание:** Неавторизованный запрос. Требуется авторизация для доступа к данному ресурсу.
  - **Content-Type:** `text/plain`, `application/json`, `text/json`
  - **Схема:** `#/components/schemas/ClientUploadedBlobVMSyncOperationResponseVM` - Объект, содержащий информацию об ошибке авторизации. Детальное описание структуры смотрите в разделе "Схемы".

Типы методов

Одиночные операции

Данная группа методов выполняет операции в синхронном режиме. Запросы к этим методам инициируют немедленное выполнение операции, и ответ возвращается после её завершения.

Массовые операции

Данная группа методов предназначена для выполнения массовых действий над документами и другими объектами в асинхронном режиме. Запросы к этим методам инициируют фоновые задачи, статус которых можно отслеживать с помощью соответствующих методов. Асинхронные методы ожидают вызов существующего метода API по протоколу HTTP и в ответе всегда возвращают идентификатор асинхронной операций. Так как асинхронные методы спроектированы с учетом того, что обработка запроса клиента может занять продолжительное время - от нескольких секунд до нескольких часов, то клиент должен получить

идентификатор запущенной асинхронной операций и опрашивать модуль о результате выполнения асинхронной операции в методе-компаньоне, который существует для каждого асинхронного метода в API.

Работа с файлами

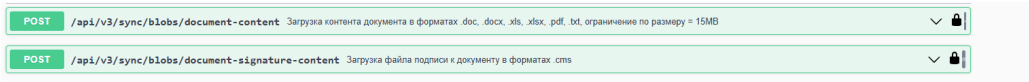
Данная группа методов позволяет загружать файлы в файловое хранилище для дальнейшей работы с ними в системе

Справочники

Данная группа методов нужна для получения справочных значений

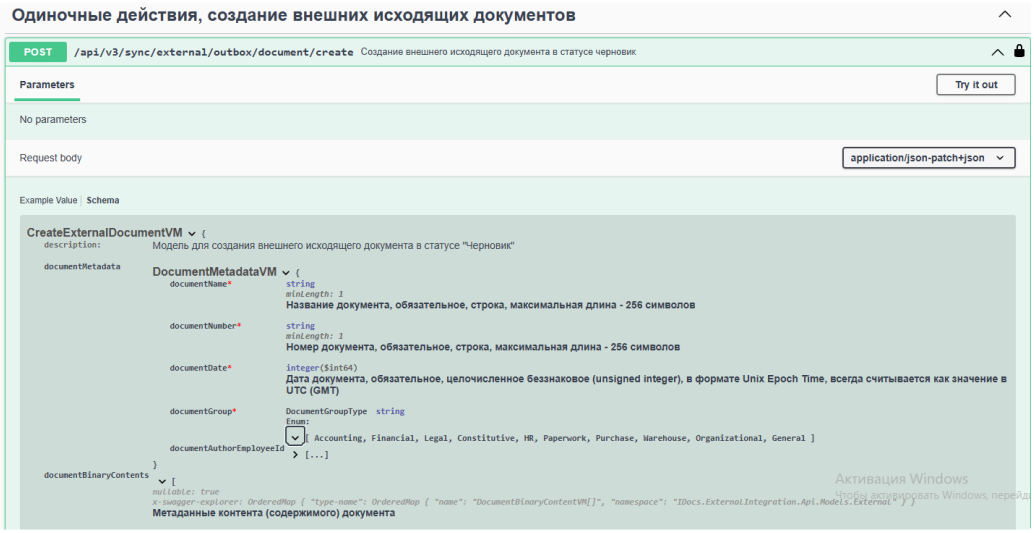
Описание методов

Описание всех методов можно найти в swagger документации



Описание полей

Описание всех входящих и исходящих моделей в методах можно найти в swagger документации во вкладке Shema



WebHooks

Что такое Webhooks?

Webhooks — это механизм, позволяющий одному приложению (в данном случае, IDocs) автоматически уведомлять другое приложение (вашу систему) о произошедших событиях в режиме реального времени. Вместо того, чтобы вашей системе постоянно опрашивать API IDocs на предмет изменений, IDocs сам отправляет уведомление, когда происходит нужное событие.

Как это работает:

1. **Подписка:** Ваша система регистрирует URL-адрес (subscriptionUrl) в IDocs, на который будут отправляться уведомления.
2. **Событие:** Когда в IDocs происходит событие, на которое вы подписались (например, изменение статуса документа), IDocs отправляет HTTP-запрос (обычно POST) на ваш зарегистрированный URL-адрес.
3. **Получение уведомления:** Ваша система получает HTTP-запрос с данными о событии и обрабатывает их.

Преимущества Webhooks:

- **Реальное время:** Получение уведомлений сразу после события.
- **Эффективность:** Нет необходимости в постоянном опросе API, что снижает нагрузку на обе системы.
- **Автоматизация:** Упрощает автоматизацию процессов, связанных с изменениями в IDocs.

.

Основные возможности:

- **Получение уведомлений об изменении статуса документов:** Получайте мгновенные уведомления, когда статус документа изменяется в системе IDocs.
- **Управление подписками на Webhooks:** Подписывайтесь на события изменения статуса документов и управляйте URL-адресами, на которые отправляются уведомления.
- **Получение истории событий документов:** Получайте историю изменений статуса конкретного документа или список событий за определенный период.

Базовый URL: `https://webhooks.idocs.kz/api/1/`

Аутентификация:

Все запросы к API требуют аутентификации с использованием токена Bearer.

- **Заголовок:** `Authorization: Bearer {token}`
- Получите токен Bearer у вашего менеджера в IDocs.

Для подключения webhook в вашей системе необходима подписка на событие "Изменение статуса документа"

Метод: `POST /webhooks/document-events/subscription/document-status-changed-event`

Описание: Создает или обновляет подписку на событие изменения статуса документа.

Сценарии использования

Отправка внешнего исходящего документа контрагенту, через быстрое подписание (одиночные действия)

1. Загрузка оригинальных файлов в систему

```
curl -X 'POST' \
'https://ext-edo-integration.idocs.kz/api/v3/sync/blobs/document-content' \
-H 'accept: text/plain' \   -H 'Authorization: Bearer <токен>' \
-H 'Content-Type: multipart/form-data' \
-F 'documentContent=<файл>;type=application/pdf'
```

2. Создание документа

```
curl -X 'POST' \
  'https://ext-edo-integration.idocs.kz/api/v3/sync/external/outbox/document/create' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer <токен>' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
  "documentMetadata": {
    "documentName": "<Имя документа>",
    "documentNumber": "<Номер документа>",
    "documentDate": <Дата документа в формате Unix timestamp>,
    "documentGroup": <Группа документа, справочное значение>,
    "documentAuthorEmployeeId": <id сотрудника создавшего документ>
  },
  "documentBinaryContents": [
    {
      "blobId": <id документа в хранилище файлов, ранее загруженного>
    }
  ]
}'
```

3. Создание быстрого маршрута

```
curl -X 'POST' \
  'https://ext-edo-integration.idocs.kz/api/v3/sync/external/outbox/route/quick-route/create' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer <токен>' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
  "documentId": <id документа ранее созданного>,
  "employeeId": <id сотрудника создавшего маршрут>,
  "route": {
    "external": {
      "nodes": [
        {
          "order": 0,
          "counterpartyBin": <bin компании которой отправляем документ>,
```

```
          "comment": "string",
          "isIndividual": <флаг определяющий является ли компания физ лицом>,
          "counterpartyEmails": [
            <почта на которую контрагент получит документ>
          ],
          "requiredActionType": <тип действия, справочное значение>
        }
      ]
    }
  }
}'
```

4. Запрос на получение контента для подписания

```
curl -X 'POST' \
  'https://ext-edo-integration.idocs.kz/api/v3/sync/external/outbox/signature/content-to-sign/generate' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer <токен>' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
  "documentId": <id документа, созданного ранее>,
  "signedByEmployeeId": <id сотрудника, который хочет подписать документ>
}'
```

5. Загрузка файла подписи

```
curl -X 'POST' \
  'https://ext-edo-integration.idocs.kz/api/v3/sync/blobs/document-signature-content' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer <токен>' \
  -H 'Content-Type: multipart/form-data' \
  -F 'signatureContent=<файл подписи>;type=application/pdf'
```

6. Сохранение подписи

```
curl -X 'POST' \
  'https://ext-edo-integration.idocs.kz/api/v3/sync/external/outbox/signature/quick-sign/save' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <токен>' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
  "documentId": <id документа, созданного ранее>,
  "signedByEmployeeId": <id сотрудника, который хочет подписать документ>,
  "signatureBinaryContent": {
    "blobId": <файл подписи загруженный ранее>
  },
  "idempotencyTicket": <токен идепотентности, необходимый для контроля процесса подписания>
}'
```