# ?????????? EasyDocs ? ????????24

**1. Предварительные требования**

**1.1 Битрикс24**

- Коммерческий тариф с доступом к REST API.
- Пользователь с правами на работу с Диском (scope: disk).
- Входящий вебхук --- для вызова методов REST из Middleware.
- Исходящий вебхук --- для получения событий из Битрикс24 в Middleware.

**1.2 EasyDocs**

- Активная учётная запись EasyDocs с API-доступом.
- API-ключ (выдаётся при регистрации/настройке интеграции).
- Настроенный маршрут согласования и подписания.
- URL для callback-уведомлений (ваш Middleware).

**1.3 Middleware**

- HTTPS-эндпоинт с публичным доступом (для входящих вебхуков от Битрикс24 и callback от EasyDocs).
- База данных или хранилище для маппинга: ID документа EasyDocs ↔ ID файла Битрикс24.
- Очередь задач (опционально) для надёжной обработки асинхронных событий.

**2. Настройка Битрикс24**

**2.1 Создание входящего вебхука**

Входящий вебхук используется Middleware для вызова методов REST API Битрикс24 (скачать файл, загрузить подписанный файл, отправить уведомление).

- Перейдите: Приложения → Разработчикам → Готовые сценарии → Входящий вебхук.
- Задайте название, например: EasyDocs Integration.
- Установите права (scope): disk, crm (если нужны уведомления в CRM).
- Сохраните URL вебхука. Формат:

> https://<your-portal>.bitrix24.ru/rest/<user_id>/<webhook_code>/

> ⚠ **Примечание:** Webhook URL является конфиденциальным. Храните его в переменных окружения, не в исходном коде.

### 2.2 Создание исходящего вебхука

Исходящий вебхук отправляет POST-запрос на ваш сервер при наступлении события в Битрикс24 (например, загрузка нового файла на Диск, смена статуса сделки и т.п.).

- Перейдите: Приложения → Разработчикам → Готовые сценарии → Исходящий вебхук.
- URL обработчика: https://<your-middleware>/bitrix/webhook
- Выберите нужное событие, например: OnCrmDealUpdate или ONDISKFILEADD.
- Сохраните токен вебхука --- используйте для верификации входящих запросов.

Пример тела запроса, который Битрикс24 пришлёт на ваш обработчик:

```
{
  "event": "ONDISKFILEADD",
  "data": { "FIELDS": { "ID": "9043" } },
  "auth": {
  "domain": "your-portal.bitrix24.ru",
  "client_endpoint": "https://your-portal.bitrix24.ru/rest/",
  "application_token": "xxx"
  }
  }
```

## 3. Настройка EasyDocs

### 3.1 Параметры подключения

---

**Base URL** https://internal-api.easydocs.ru/ **Документация (Swagger)** https://internal-api.easydocs.ru/docs/tech.html **Аутентификация** API-ключ в заголовке Authorization **Content-Type** application/json

---

Заголовок авторизации для всех запросов к EasyDocs:

> Authorization: Bearer <your_easydocs_api_key>
>
> Content-Type: application/json

### 3.2 Настройка callback-URL

В настройках интеграции EasyDocs укажите URL вашего Middleware, на который EasyDocs будет отправлять POST-уведомления о смене статуса документа:

> https://<your-middleware>/easydocs/callback
>
> ⚠ **Примечание:** Точное место настройки callback URL уточните в документации EasyDocs (раздел Webhooks / Интеграции) или в поддержке EasyDocs.

## 4. Реализация интеграции

### 4.1 Шаг 1: Получение файла из Битрикс24

После получения события (исходящий вебхук) Middleware получает ID файла и запрашивает его параметры методом disk.file.get.

**Метод: disk.file.get**

> POST https://<portal>.bitrix24.ru/rest/<uid>/<webhook>/disk.file.get
>
> {
>
> "id": 9043
>
> }

Ответ содержит DOWNLOAD_URL --- временную ссылку для скачивания файла:

> {
>
> "result": {

```
"ID": "9043",

"NAME": "contract.pdf",

"SIZE": "102400",

"PARENT_ID": "8996",

"STORAGE_ID": "1357",

"DOWNLOAD_URL":
"https://<portal>.bitrix24.ru/rest/download.json?auth=...&token=..."

}

}
```

Скачайте файл с DOWNLOAD_URL (GET-запрос, токен действует ограниченное время):

```
# Python-пример

import requests

def download_from_bitrix(download_url: str) -> bytes:

resp = requests.get(download_url, timeout=60)

resp.raise_for_status()

return resp.content
```

⚠ **Примечание:** DOWNLOAD_URL содержит токен с ограниченным сроком действия. Скачивайте файл сразу после получения ссылки, не кэшируйте URL.

### 4.2 Шаг 2: Отправка файла в EasyDocs на подпись

Передайте содержимое файла в EasyDocs API. Файл кодируется в Base64 и включается в тело запроса (либо загружается multipart/form-data --- уточните по Swagger).

**Эндпоинт: POST /api/v1/documents (уточнить по Swagger)**

```
POST https://internal-api.easydocs.ru/api/v1/documents

Authorization: Bearer <api_key>

Content-Type: application/json

{

"name": "contract.pdf",

"file_content": "<base64-encoded-content>",

"file_name": "contract.pdf",

"route_id": "<signing_route_id>",

"external_id": "b24-file-9043",

"callback_url": "https://<your-middleware>/easydocs/callback"

}
```

Ответ EasyDocs содержит идентификатор документа:

```
{

"id": "ed-doc-uuid-1234",

"status": "pending",

"name": "contract.pdf",

"created_at": "2025-06-01T10:00:00Z"

}
```

Сохраните маппинг в БД:

```
# Псевдокод

db.save({
```

> "easydocs_doc_id": "ed-doc-uuid-1234",
>
> "bitrix_file_id": 9043,
>
> "bitrix_folder_id": 8996,
>
> "bitrix_file_name": "contract.pdf",
>
> "status": "pending",
>
> "created_at": datetime.utcnow()
>
> })

**Параметры запроса к EasyDocs**

---

**Параметр Тип Описание** name string Имя документа (отображается в EasyDocs) file_content string (base64) Содержимое файла, закодированное в Base64 file_name string Имя файла с расширением (например, contract.pdf) route_id string ID маршрута согласования/подписания в EasyDocs external_id string Ваш внешний идентификатор (для связи с Битрикс24) callback_url string URL Middleware для получения уведомлений о статусе

---

> ⚠ **Примечание:** Точные названия полей уточните в Swagger-документации EasyDocs по адресу https://internal-api.easydocs.ru/docs/tech.html

**4.3 Шаг 3: Обработка callback от EasyDocs**

EasyDocs отправляет POST-запрос на callback_url при каждой смене статуса документа. Middleware должен обработать два конечных статуса:

- signed (подписан) --- документ прошёл весь маршрут и подписан.
- rejected / declined (отклонён) --- один из участников отклонил документ.

Пример callback-запроса от EasyDocs:

> POST https://<your-middleware>/easydocs/callback
>
> Content-Type: application/json
>
> X-EasyDocs-Signature: <hmac_signature>

```
{

"event": "document.status_changed",

"document_id": "ed-doc-uuid-1234",

"external_id": "b24-file-9043",

"status": "signed",

"signed_file_url": "https://internal-api.easydocs.ru/documents/ed-doc-uuid-1234/download",

"timestamp": "2025-06-01T12:30:00Z"

}
```

Callback-обработчик должен:

- Верифицировать подпись (X-EasyDocs-Signature) --- проверьте способ верификации в документации EasyDocs.
- Немедленно вернуть HTTP 200, до начала обработки (чтобы EasyDocs не повторял запрос).
- Поставить задачу в очередь для асинхронной обработки.

```
# Python / FastAPI --- пример обработчика

@app.post('/easydocs/callback')

async def easydocs_callback(request: Request):

body = await request.json()

# 1. Верификация подписи

verify_signature(request.headers, body)

# 2. Немедленный ответ 200

task_queue.enqueue(process_easydocs_event, body)

return {'ok': True}
```

**4.4 Шаг 4а: Загрузка подписанного файла в Битрикс24**

Если статус signed: скачайте подписанный файл из EasyDocs и загрузите его на Диск Битрикс24 методом disk.folder.uploadfile.

**4a.1 --- Скачать подписанный файл из EasyDocs**

> GET https://internal-api.easydocs.ru/documents/<doc_id>/download
>
> Authorization: Bearer <api_key>
>
> → Бинарное содержимое файла (PDF)

**4a.2 --- Загрузить в Битрикс24 (disk.folder.uploadfile)**

Загрузка через Base64 (для файлов до ~10 МБ):

> POST https://<portal>.bitrix24.ru/rest/<uid>/<webhook>/disk.folder.uploadfile
>
> {
>
> "id": 8996,
>
> "data": { "NAME": "contract_signed.pdf" },
>
> "fileContent": [
>
> "contract_signed.pdf",
>
> "<base64-encoded-signed-file>"
>
> ],
>
> "generateUniqueName": true
>
> }

Для файлов более ~10 МБ используйте двухэтапную загрузку через UploadUrl:

> # Шаг 1: получить UploadUrl
>
> POST .../disk.folder.uploadfile

```
{ "id": 8996 }

→ { "result": { "uploadUrl": "https://...", "field": "file" } }

# Шаг 2: загрузить файл multipart/form-data на uploadUrl

POST <uploadUrl>

Content-Type: multipart/form-data

file=@contract_signed.pdf
```

Ответ disk.folder.uploadfile содержит ID нового файла на Диске:

```
{

"result": {

"ID": 9100,

"NAME": "contract_signed.pdf",

"DOWNLOAD_URL": "https://...",

"DETAIL_URL": "https://..."

}

}
```

**4.5 Шаг 4b: Уведомление об отклонении**

Если статус rejected --- отправьте уведомление в Битрикс24. Варианты:

**Вариант 1: Уведомление через im.notify**

```
POST .../im.notify

{

"to": <user_id>,
```

```
"message": "Документ contract.pdf отклонён в EasyDocs.",
"type": "SYSTEM"
}
```

**Вариант 2: Комментарий в таймлайне CRM (crm.timeline.comment.add)**

```
POST .../crm.timeline.comment.add
{
"fields": {
"ENTITY_TYPE": "deal",
"ENTITY_ID": <deal_id>,
"COMMENT": "Документ contract.pdf отклонён в EasyDocs."
}
}
```

⚠ **Примечание:** Сохраните связь между файлом Битрикс24 и сущностью CRM (сделка, контакт и т.п.) на момент отправки, чтобы корректно добавить комментарий.

**5. Полный цикл --- псевдокод**

```
# === ШАГ 1: обработка вебхука от Битрикс24 ===
def handle_bitrix_webhook(event_data):
file_id = event_data['data']['FIELDS']['ID']
# 2. Получить параметры файла
file_info = bitrix_call('disk.file.get', {'id': file_id})
download_url = file_info['DOWNLOAD_URL']
```

```
file_name = file_info['NAME']

folder_id = file_info['PARENT_ID']

# 3. Скачать файл

file_bytes = requests.get(download_url).content

file_b64 = base64.b64encode(file_bytes).decode()

# 4. Отправить в EasyDocs

ed_doc = easydocs_post('/api/v1/documents', {

'name': file_name,

'file_content': file_b64,

'file_name': file_name,

'route_id': ROUTE_ID,

'external_id': f'b24-{file_id}',

'callback_url': CALLBACK_URL

})

# 5. Сохранить маппинг

db.save(easydocs_id=ed_doc['id'], bitrix_file_id=file_id,

folder_id=folder_id, file_name=file_name)

# === ШАГ 2: обработка callback от EasyDocs ===

def process_easydocs_event(event):

mapping = db.get(easydocs_id=event['document_id'])

if event['status'] == 'signed':

# Скачать подписанный файл

signed_bytes = easydocs_get(event['signed_file_url'])

signed_b64 = base64.b64encode(signed_bytes).decode()
```

```
signed_name = mapping.file_name.replace('.pdf', '_signed.pdf')

# Загрузить в Битрикс24

bitrix_call('disk.folder.uploadfile', {

'id': mapping.folder_id,

'data': {'NAME': signed_name},

'fileContent': [signed_name, signed_b64],

'generateUniqueName': True

})

db.update(easydocs_id=event['document_id'], status='signed')

elif event['status'] in ('rejected', 'declined'):

# Уведомить ответственного

bitrix_call('im.notify', {

'to': mapping.responsible_user_id,

'message': f"{mapping.file_name} отклонён в EasyDocs.",

'type': 'SYSTEM'

})

db.update(easydocs_id=event['document_id'], status='rejected')
```

### 6. Справочник используемых методов Битрикс24

---

**Метод Scope Назначение** disk.file.get disk Получить параметры файла, в т.ч. DOWNLOAD_URL disk.folder.uploadfile disk Загрузить файл в папку (Base64 или multipart) disk.storage.getchildren disk Получить список файлов в хранилище disk.folder.getchildren disk Получить список файлов в папке im.notify im Отправить уведомление пользователю crm.timeline.comment.add crm Добавить комментарий в таймлайн CRM event.bind --- Подписаться на событие программно (в приложении)

---

### 6.1 Входящий вебхук --- формат URL

```
```

> https://<portal>.bitrix24.ru/rest/<user_id>/<webhook_code>/<method>
>
> Пример:
>
> https://mycompany.bitrix24.ru/rest/1/abc123xyz/disk.file.get

**6.2 Авторизация через OAuth (для приложения)**

Если интеграция реализуется как полноценное приложение Битрикс24, используйте OAuth 2.0 вместо вебхука --- для поддержки refresh_token и работы на нескольких порталах. Подробности: https://apidocs.bitrix24.ru/settings/oauth/

**7. Обработка ошибок и повторные попытки**

**7.1 Ошибки Битрикс24**

---

**Код ошибки HTTP Рекомендация** QUERY_LIMIT_EXCEEDED 503 Пауза 10 сек, повторить запрос (exponential backoff) OPERATION_TIME_LIMIT 429 Пауза 10 мин (автоснятие блокировки) ACCESS_DENIED 403 Проверить scope вебхука и права пользователя ERROR_NOT_FOUND 400 Файл не найден --- проверить ID файла NO_AUTH_FOUND 401 Неверный вебхук-код или истёк access_token expired_token 401 Обновить access_token через refresh_token (OAuth)

---

**7.2 Ошибки EasyDocs**

Для ошибок EasyDocs API реализуйте:

- Retry с экспоненциальной задержкой для 5xx ошибок (до 3 попыток).
- Логирование всех 4xx ошибок с телом ответа --- они требуют исправления данных.
- Dead Letter Queue для событий, которые не удалось обработать после всех попыток.

**7.3 Идемпотентность callback**

EasyDocs может прислать один callback несколько раз (at-least-once delivery). Проверяйте статус документа в БД перед обработкой:

> def process_easydocs_event(event):
>
> mapping = db.get(easydocs_id=event['document_id'])
>
> if mapping.status in ('signed', 'rejected'):
>
> return # уже обработано, пропустить

```
# ... основная логика ...
```

### 8. Безопасность

---

**Угроза Меры защиты Реализация** Перехват вебхука Битрикс24 HTTPS + верификация application_token Проверять auth.application_token в заголовке Поддельный callback от EasyDocs HMAC-подпись (X-EasyDocs-Signature) Сверять подпись с секретом; отклонять невалидные Утечка credentials Переменные окружения, secrets manager Не хранить ключи в коде/репозитории DOWNLOAD_URL кэширование Использовать сразу после получения URL содержит одноразовый токен с TTL Переполнение очереди Rate limiting на callback-эндпоинте Nginx/API Gateway: max 100 req/min

---

### 9. Тестирование

#### 9.1 Проверка вебхуков Битрикс24

Воспользуйтесь встроенным инструментом в разделе Разработчикам → Входящий вебхук → Выполнить, или используйте curl:

```
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{"id": 9043}' \
  https://<portal>.bitrix24.ru/rest/<uid>/<webhook>/disk.file.get
```

#### 9.2 Симуляция callback от EasyDocs

Для локального тестирования используйте ngrok для проброса порта и симулируйте callback вручную:

```
curl -X POST https://<ngrok-url>/easydocs/callback \
  -H "Content-Type: application/json" \
  -H "X-EasyDocs-Signature: <test_signature>" \
  -d '{
```

```
"event": "document.status_changed",

"document_id": "ed-doc-uuid-1234",

"status": "signed",

"signed_file_url": "https://..."

}'
```

### 9.3 Чеклист перед запуском

- Входящий вебхук Битрикс24 создан, URL сохранён в конфиг.
- Исходящий вебхук настроен, указывает на Middleware.
- API-ключ EasyDocs корректен --- проверить GET /api/v1/me или аналогичный health-check.
- Маршрут подписания в EasyDocs существует, route_id известен.
- Callback URL зарегистрирован в EasyDocs.
- Маппинг БД создаётся и читается корректно.
- Подписанный файл появляется в нужной папке Битрикс24.
- Уведомление об отклонении доставляется ответственному.
- Повторная отправка callback обрабатывается идемпотентно.

## 10. Ссылки

---

**EasyDocs API (Swagger)** https://internal-api.easydocs.ru/docs/tech.html **Битрикс24 REST API** https://apidocs.bitrix24.ru/ **disk.file.get** https://apidocs.bitrix24.ru/api-reference/disk/file/disk-file-get.html **disk.folder.uploadfile** https://apidocs.bitrix24.ru/api-reference/disk/folder/disk-folder-upload-file.html **Как загрузить файлы** https://apidocs.bitrix24.ru/api-reference/files/how-to-upload-files.html **Вебхуки Битрикс24** https://apidocs.bitrix24.ru/local-integrations/local-webhooks.html **Лимиты REST API Битрикс24** https://apidocs.bitrix24.ru/limits.html **im.notify** https://apidocs.bitrix24.ru/api-reference/chats/

---

---

Версия #2
Алексей создал 2026-06-08 13:37:16 UTC
Алексей обновил 2026-06-08 15:52:54 UTC