SCANEE

WHITE PAPER

Scanee API and webhook integration

PUBLISHED BY

Scanee (scanee.ai)

© 2026 Scanee.ai — All rights reserved.

DOCUMENT STRUCTURE

Contents

	Scope	03
01	Strict isolation by project	03
02	The principle: three beats	04
03	Step 1: get your URL and API key	04
04	Step 2: send a document	05
05	Step 3: set up the fields to extract	10
06	Step 4: receive the extracted data (webhooks)	11
07	Step 5: reliability, log and errors	16
08	The file life cycle	17
09	Cheat sheet	18
	Conclusion: one loop, three calls, no data entry	19

© 2026 Scanee.ai — All rights reserved.

Scope

This guide covers the ingestion API and outgoing webhooks. The examples use keys of the form `scanee_...` and the placeholders `<workspace>` and `<project>`: replace them with the values from your project's "Receiving" tab.

In short

- Each project is a silo: its own API key, fields, webhooks and capture address.
- You send a document to the project's address. Scanee answers right away with `202 Accepted`, before extraction.
- Scanee extracts the fields defined in the project, then pushes the result to your webhook at the status you choose.
- Six formats accepted, 25 MB per file, 25 documents per request.

01 Strict isolation by project

In Scanee, everything starts with creating a project: "Vendor invoices", "Purchase orders", "Expense reports", "Bank statements". Each project is a self-contained silo with its own configuration.

What belongs to each project	Detail
An HTTP endpoint	`https://<workspace>.docs.scanee.ai/<project>/ingest`, built from the project's acronym
Its API keys	A key can be restricted to a single project. A key for the "Invoices" project cannot drop a document into "Expense reports": the request is rejected with `403 KEY_PROJECT_MISMATCH`
Its data schema	The fields to extract: text, amounts, dates, lists, multi-page tables

© 2026 Scanee.ai — All rights reserved.

What belongs to each project	Detail
Its AI instruction	Business rules in plain language, for example Quebec tax rules for one project and Ontario rules for another
Its webhooks and export mappings	Where to send the data, with which JSON schema and which authentication
Its email capture address	Its own address and its list of approved senders

Key point. One project means one key, one schema, one webhook and one capture address. Nothing crosses the project boundary.

02 The principle: three beats

Scanee works as an asynchronous loop: your system doesn't wait for extraction to finish, it is called back when it's done.

Beat	Who	What
1	Your system (ERP, script, app)	Sends the file to the project's address: file, URL or base64
2	The Scanee project	Extracts the data according to the configured fields
3	Your server (API, Zapier, Make, database)	Receives the JSON and the file's signed links, at the chosen status

Implementation takes five steps: get the URL and key (section 3), send a document (section 4), set up the fields (section 5), receive the data (section 6), handle errors (section 7).

03 Step 1: get your URL and API key

1. Open your project in Scanee, then the **"Receiving"** tab.

© 2026 Scanee.ai — All rights reserved.

2. Your project's ingestion URL is there:

`https://<your-workspace>.docs.scanee.ai/<your-project>/ingest`

Example: `https://acme.docs.scanee.ai/invoices/ingest`. The `/documents` alias also works in place of `/ingest`.

3. Click **"Create API Key"**, give it a label (for example `ERP-Prod`) and copy the generated token (format `scanee_...`).
4. Provide this key in every request, with either of these headers:

`X-API-Key: scanee_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

`Authorization: Bearer scanee_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

Treat the key like a password: keep it server-side, never in a mobile app or a public web page.

04 Step 2: send a document

Every method targets the same endpoint, the project's.

Method	Format	When to use it
A — Direct file upload	`multipart/form-data`	Files generated locally or stored on your server
B — Remote or presigned URL	JSON	Files already on AWS S3, Google Cloud Storage, Azure Blob, or generated by your ERP
C — Base64	JSON	Mobile apps, connected scanners, serverless functions without a file system
D — Inbound email (no code)	Email	Each project has its capture address and its list of approved senders ("Receiving" tab)

The API also accepts a single file sent as a raw binary stream in the request body.

© 2026 Scanee.ai — All rights reserved.

Quotas and rules

Rule	Value
Accepted formats	PDF, PNG, JPEG, WEBP, TIFF, HEIC (HEIF)
Maximum size per file	25 MB (after decoding, for base64)
Documents per request	25 at most

For large volumes, send several requests, or use presigned URLs (method B) rather than base64.

Common parameters. `autoProcess` starts extraction on receipt; it is on by default, and `false` turns it off. `metadata.externalId` keeps your business reference: it comes back unchanged in the response and at export.

Method A — Direct file upload (multipart/form-data)

```bash
curl -X POST "https://<workspace>.docs.scanee.ai/<project>/ingest" \
  -H "X-API-Key: scanee_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -F "files=@./invoice_01.pdf" \
  -F 'metadata={"externalId": "ERP-INV-1092", "department": "Accounting"}' \
  -F "autoProcess=true"
```

Field	Role
`files`	The file or files in the batch. Repeat the field to send several documents in the same call, up to 25
`metadata`	Free-form JSON object. `externalId` carries your business reference
`autoProcess`	On by default; `false` stores the document without starting extraction

Beyond 25 files, the request is rejected with `400 BATCH_LIMIT_EXCEEDED`: split into batches of 25.

© 2026 Scanee.ai — All rights reserved.

Node.js / TypeScript (Axios and form-data; the file is streamed, never fully loaded into memory):

```typescript
import axios from 'axios';
import FormData from 'form-data';
import fs from 'fs';

async function sendInvoice() {
  const form = new FormData();
  form.append('files', fs.createReadStream('./invoice_01.pdf'));
  form.append('metadata', JSON.stringify({ externalId: 'INV-1092' }));
  form.append('autoProcess', 'true');

  const res = await axios.post(
    'https://<workspace>.docs.scanee.ai/<project>/ingest',
    form,
    {
      headers: {
        ...form.getHeaders(),
        'X-API-Key': process.env.SCANEE_API_KEY,
      },
    }
  );

  console.log('Document received by Scanee:', res.data);
}
```

Python (Requests):

```python
import json
import os
import requests

url = "https://<workspace>.docs.scanee.ai/<project>/ingest"
headers = {"X-API-Key": os.environ["SCANEE_API_KEY"]}

with open("invoice_01.pdf", "rb") as f:
    files = [("files", ("invoice_01.pdf", f, "application/pdf"))]
    data = {
```

© 2026 Scanee.ai — All rights reserved.

```
        "metadata": json.dumps({"externalId": "INV-1092"}),
        "autoProcess": "true",
    }
    res = requests.post(url, headers=headers, files=files, data=data)

print(res.status_code, res.json())
```

Method B — Remote or presigned URL (JSON)

```BASH
curl -X POST "https://<workspace>.docs.scanee.ai/<project>/ingest" \
  -H "X-API-Key: scanee_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "autoProcess": true,
    "documents": [
      {
        "url": "https://my-bucket.s3.ca-central-1.amazonaws.com/invoices/inv_1092.pdf?X-Amz-
Signature=...",
        "filename": "inv_1092.pdf",
        "externalId": "INV-1092"
      }
    ]
  }'
```

What Scanee does on receipt:

1. **Direct server-to-server download.** Scanee makes a `GET` request to the URL provided, with a 30-second timeout and a 25 MB cap.
2. **Real type check.** Scanee doesn't rely only on the extension or declared type: it reads the file's first bytes (the `%PDF` signature for a PDF, image headers) to confirm the format.
3. **Copy into the project's space.** The file is stored, encrypted, in the project's isolated space.
4. `202 Accepted` **response** with the batch and the tasks created.

If the URL doesn't respond, or returns an error, the request is rejected with `422`.

© 2026 Scanee.ai — All rights reserved.

Security benefit. You open neither your network nor an SFTP access. A 5- or 10-minute presigned URL is enough: once you receive `202 Accepted`, Scanee has its working copy and the URL can expire.

Method C — Base64 (JSON)

```bash
curl -X POST "https://<workspace>.docs.scanee.ai/<project>/ingest" \
  -H "X-API-Key: scanee_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "autoProcess": true,
    "documents": [
      {
        "filename": "cash_receipt.jpg",
        "data": "...",
        "externalId": "REC-8801"
      }
    ]
  }'
```

The `data` field is a full data URL, MIME prefix included (``), followed by the encoded payload. The `base64` field also accepts the payload alone. Same limits: 25 MB per file after decoding, 6 formats, 25 documents per request.

The immediate response: 202 Accepted

Whatever the method, the response has the same shape:

```json
{
  "status": "success",
  "batchId": "batch_9a4f21de88b0",
  "project": "invoices",
  "tenantId": "acme",
```

© 2026 Scanee.ai — All rights reserved.

```
  "totalFiles": 1,
  "tasks": [
    {
      "taskId": "1024",
      "filename": "invoice_01.pdf",
      "status": "pending",
      "trackingUrl": "https://app.scanee.ai/projects/invoices/tasks/67d8f1e2700bd45d79cc1918",
      "externalId": "INV-1092"
    }
  ]
}
```

- `taskId` : a short sequential number given to the document in your project.
- `trackingUrl` : a direct link to the document in the Scanee interface.
- `externalId` : your business reference, returned unchanged.

05 Step 3: set up the fields to extract

You don't code OCR rules or models. You define the fields you want in the project's **"Fields"** tab.

Field type	Use	Example extracted value
Input / TextArea	Single or multi-line text	`"INV-2026-089"`, `"ACME Supplies Inc."`
Number	Amount or quantity	`1437.50` (separators normalized)
DatePicker	Date	`2026-09-22T00:00:00.000Z` (ISO)
Select / Radio	Choice from a list	Currency: `CAD` , `USD` , `EUR`
List (master list)	The AI matches the text to your official list (vendors, accounts)	`"Hydro-Qc"` → `"Hydro-Québec"`

© 2026 Scanee.ai — All rights reserved.

Field type	Use	Example extracted value
Table (multi-page)	Each line item: description, quantity, price, total	`[{"desc": "Item A", "qty": 2, "price": 50}]`
Tags (2D boxes)	Where the value sits on the document	normalized `[ymin, xmin, ymax, xmax]`

Project AI instruction. In the project's "General" tab, add a business instruction in plain language. For example: "For invoices with Canadian taxes, isolate GST (5%) and QST (9.975%). If the purchase order number appears in the notes, extract it into po_number."

06 Step 4: receive the extracted data (webhooks)

As soon as a document reaches the chosen status, Scanee pushes it to your server with an HTTP `POST` request.

Set up the webhook

1. Open the project's **"Exports"** tab.
2. Click **"Add configuration"** and choose the **"API / Zapier / Make"** (webhook) format.
3. Fill in the name (for example `ERP Production Webhook`), the URL (for example `https://api.your-domain.com/webhooks/scanee`) and the authentication.

Authentication	Effect
None	No authentication header
Bearer	Adds `Authorization: Bearer <token>`
API key	Adds a custom header, `x-api-key` by default, or a name of your choice (for example `X-Webhook-Secret`)

© 2026 Scanee.ai — All rights reserved.

Automatic triggers

Status	When
New (`new`)	As soon as the document enters the project, before extraction
Reviewing (`reviewing`)	The AI has extracted the data; the document is ready for approval. For fully hands-off automation
Completed (`completed`)	Someone has reviewed the document and moved it to Completed: this is the sign-off

You can also trigger a send by hand, one document at a time.

Field mapping

In the export editor, you map each Scanee field to the JSON key your API expects.

Source field in Scanee	Target JSON key (example)	Role
ID	`scanee_doc_id`	Short sequential number (for example 1024)
`metadata.externalId`	`external_id`	Your original reference
`details.<vendor_field_id>`	`vendor_name`	Extracted vendor name
`details.<total_field_id>`	`total_amount`	Total amount
`details.<lines_id>`	`items`	Line items, as an array of objects; `table.<lines_id>.<column>` gives only one column's values
`fileData.signedUrl`	`document_download_url`	Signed link to download the original PDF
`fileData.annotatedSignedUrl`	`annotated_pdf_url`	Link to the PDF with detection boxes

© 2026 Scanee.ai — All rights reserved.

Output keys are free-form: each project defines its own schema. The same document can feed an ERP with one schema and a payroll system with another, with no transformation on your side.

What your webhook receives

The body is a JSON array, one object per document:

```JSON
[
  {
    "scanee_doc_id": 1024,
    "external_id": "INV-1092",
    "invoice_number": "FA-2026-0042",
    "invoice_date": "2026-09-22",
    "vendor_name": "ACME Supplies Inc.",
    "subtotal": 1250.00,
    "gst": 62.50,
    "qst": 124.69,
    "total_amount": 1437.19,
    "currency": "CAD",
    "items": [
      {
        "description": "Annual software licences",
        "quantity": 5,
        "unit_price": 200.00,
        "line_total": 1000.00
      },
      {
        "description": "Installation fee",
        "quantity": 1,
        "unit_price": 250.00,
        "line_total": 250.00
      }
    ],
    "document_download_url": "https://storage.scanee.ai/storage/export/...?sig=...",
    "annotated_pdf_url": "https://storage.scanee.ai/storage/export-annotated/665f1c...?
sig=..."
  }
]
```

© 2026 Scanee.ai — All rights reserved.

Download links. `document_download_url` and `annotated_pdf_url` are signed by Scanee (HMAC signature): your server or your ERP can `GET` them without authenticating again. They can't be guessed, but they don't expire: treat them as confidential data and never expose them publicly.

Node.js / Express receiver

JAVASCRIPT

```
const express = require('express');
const app = express();
app.use(express.json());

app.post('/webhooks/scanee', (req, res) => {
  // 1. Check the secret ("API key" authentication with the X-Webhook-Secret header)
  const webhookSecret = req.headers['x-webhook-secret'];
  if (webhookSecret !== process.env.SCANEE_WEBHOOK_SECRET) {
    return res.status(401).json({ error: 'Unauthorized' });
  }

  // 2. Process the documents received
  for (const doc of req.body) {
    console.log(`Invoice received: ${doc.invoice_number}, total: ${doc.total_amount}`);
    // Insert into your ERP or database
  }

  // 3. Answer 200 quickly (under 15 seconds)
  res.status(200).json({ status: 'received' });
});

app.listen(3000, () => console.log('Webhook receiver ready on port 3000'));
```

© 2026 Scanee.ai — All rights reserved.

Python / FastAPI receiver

```python
import os
from typing import Any, Dict, List

from fastapi import FastAPI, Header, HTTPException, status

app = FastAPI()
SECRET_TOKEN = os.environ["SCANEE_WEBHOOK_TOKEN"]

@app.post("/webhooks/scanee")
async def receive_scanee_webhook(
    payload: List[Dict[str, Any]],
    authorization: str | None = Header(default=None),
):
    # 1. Check the Bearer token
    if authorization != f"Bearer {SECRET_TOKEN}":
        raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid token")

    # 2. Process the data
    for doc in payload:
        print(f"Document {doc.get('scanee_doc_id')} - amount: {doc.get('total_amount')}")
        # Save to your system

    return {"status": "ok"}
```

Zapier, Make or n8n, without writing code

1. Create a **Webhooks by Zapier** trigger (or a *Custom Webhook* in Make or n8n).
2. Copy the URL it gives you.
3. Paste it into the **Webhook URL** field of Scanee's **Exports** tab.
4. Send a test document: the mapped variables appear, ready to go to Google Sheets, Slack, Airtable, Notion or your ERP.

© 2026 Scanee.ai — All rights reserved.

07 Step 5: reliability, log and errors

Ingestion API HTTP codes

Code	Cause	Action
`202`	Documents accepted and queued	Keep the `taskId` or `batchId`
`400`	Empty request, document with neither `url` nor `data`, or more than 25 documents (`BATCH_LIMIT_EXCEEDED`)	Split into batches of 25, check the body
`401`	Key missing (`MISSING_API_KEY`) or invalid, revoked (`INVALID_API_KEY`)	Check the `X-API-Key` or `Authorization` header
`403`	Key from another workspace (`KEY_TENANT_MISMATCH`) or not allowed on this project (`KEY_PROJECT_MISMATCH`)	Check the workspace and project acronym in the URL
`404`	Workspace or project not found	Check the project URL in the "Receiving" tab
`413`	File over 25 MB	Lower the resolution or split the document
`415`	Format not accepted	Provide a PDF, PNG, JPEG, WEBP, TIFF or HEIC
`422`	Remote URL unreachable (method B)	Check that the URL is valid and not yet expired

If your webhook server doesn't respond

1. **Automatic retries.** If your server returns a `5xx` error, doesn't answer within 15 seconds or can't be reached, Scanee keeps the message and retries automatically, up to 5 attempts in total.
2. **No retry on a 4xx error.** A `4xx` response (for example `401` or `404`) is treated as final: fix the configuration, then resend by hand.
3. **Full log.** Every send is recorded in the project's history: date and time, HTTP response code, payload sent and error message.

© 2026 Scanee.ai — All rights reserved.

4 **Manual resend.** If your server was down for maintenance, open the webhook history in Scanee and click **Resend**.

Your receiver should therefore answer `200` quickly, ideally before doing heavy processing, and be able to receive the same document twice: deduplicate on `scanee_doc_id` or `external_id`.

08 The file life cycle

When Scanee receives your document, it keeps an encrypted working copy, isolated in the project, for extraction. You decide what happens to it.

Automatic deletion

Automatic deletion is off by default: documents stay in the project until you delete them. In the project's **"Storage"** tab, under "Data Retention & Privacy", you can turn it on and set a period in days (7 by default; for example 30 or 90), with one of these two modes:

Mode	Effect
Delete files only	The PDF is destroyed; the metadata and extracted data remain available
Purge everything	The file, the extracted data and the document's history are deleted

Bring your own storage (BYOS)

You can connect your own storage: a Google Drive, Microsoft OneDrive, SharePoint or Dropbox folder, or, on the Scale plan, AWS S3, Google Cloud Storage, Azure Blob Storage or an enterprise SFTP server.

#	Step	Detail
1	Extraction	Processing of the document is complete
2	Delay before transfer	In hours (24 by default), and can be set to 0

© 2026 Scanee.ai — All rights reserved.

#	Step	Detail
3	Transfer cycle	Transfers run in hourly cycles, which leaves time to cancel them
4	Transfer	The document goes to your bucket or folder, then the internal working copy is deleted

Key point. The document ends its life with you without disappearing from Scanee: once transferred, it stays visible in the project, which keeps the link to the file. Connections follow the provider's protocol: OAuth, service account or private key.

09 Cheat sheet

Send a document:

```bash
curl -X POST "https://<workspace>.docs.scanee.ai/<project>/ingest" \
  -H "X-API-Key: scanee_..." \
  -F "files=@my_document.pdf" \
  -F 'metadata={"externalId": "REF-001"}'
```

Accepted authentication headers: `X-API-Key: scanee_...` or `Authorization: Bearer scanee_...`

Formats and limits: PDF, PNG, JPEG, WEBP, TIFF, HEIC. 25 MB per file, 25 documents per request.

Webhook trigger statuses:

Value	Status	When
`new`	New	As soon as the document is received
`reviewing`	Reviewing	Ready for approval, for full automation
`completed`	Completed	After someone has reviewed it

© 2026 Scanee.ai — All rights reserved.

Conclusion: one loop, three calls, no data entry

You send a document to the project's address. Scanee extracts it according to the fields you defined. Your server receives the JSON and the file's signed links. The rest (retries, log, project isolation) is handled by the platform.

Next step. Open your project's "Receiving" tab, create an `ERP-Prod` key and send a first test document with the cheat-sheet command. The webhook is then set up in the "Exports" tab.

© 2026 Scanee.ai — All rights reserved.

SCANEE

scanee.ai

© 2026 Scanee.ai — All rights reserved.