

Integration checklist

From an API key to the first invoice reported to the FTA, in order, with what to check at each step.

Source: <https://alyabooks.ai/docs/guides/integration-checklist>

Work through these in order. Steps 1 to 6 run entirely in the simulated mode and need nothing from the service provider; steps 7 to 9 need the company's Taxilla credentials.

1. Company profile

In Alya Books, under **Settings** → **Company**, the company needs its legal name, TRN, Peppol ID (0235: plus the first ten digits of the TRN), address with emirate, trade licence number and licensing authority, and a bank account if invoices carry payment details. Every e-invoice takes the seller data from here, not from your request. `GET /me` shows what is set.

2. API key

Create a **test key** under **API access** with the scopes you need (`invoices:read`, `invoices:write`, `incoming:read`, `incoming:write`, `webhooks:write` for most integrations). Test keys only see test documents, which can only be submitted in the simulated mode. See [Getting started](#).

3. First document

`POST /invoices/validate` with a request shaped like one of the [sample documents](#). Fix every error it returns; warnings do not block. Then `POST /invoices` with the same body and `submit: true`, and read `GET /invoices/{id}/status` until `invoiceStatus` is `accepted` (the simulated ASP answers within seconds).

4. Every document kind you issue

Repeat step 3 for each kind your business uses: credit notes with `precedingInvoice`, exports in a foreign currency, reverse charge, summary invoices, deemed supplies, free zone supplies, profit margin. The [outgoing invoices](#) and [VAT and exports](#) guides cover the fields; the simulated mode has scenarios for rejections at each stage so your error handling can be exercised (`Settings` → `E-invoicing` → `Simulated outcome`).

5. Webhooks

Create a webhook for `invoice.reported`, `invoice.rejected` and `invoice.paid` (and `incoming_invoice.received` if you receive documents). Verify the signature with the secret shown once at

creation. Send a test delivery with `POST /webhooks/{id}/test`. See [Webhooks](#).

6. Incoming documents

If suppliers send you e-invoices, run `POST /incoming/simulate` to receive sample supplier invoices through the same pipeline a real Peppol delivery uses, then read them with `GET /import/incoming/{importId}/content` and accept or reject with `PATCH /import/incoming/{importId}`. See [Incoming invoices](#).

7. Onboard with the service provider

The company's owner registers the company with the FTA's e-invoicing programme on EmaraTax and links it to Taxilla, then enters the client id and secret under **Settings** → **E-invoicing** in mode **Taxilla UAT**. The help centre article "Onboard with Taxilla on EmaraTax" walks through the portal. **Test connection** must report a token before going on.

8. UAT round

With a **live key** (UAT submissions are real documents in Alya Books, not test documents), submit one document of each kind from step 4. Each should reach `accepted`; a `rejected` with `stage: taxilla` names the rule in `error`. `GET /invoices/{id}/submissions` shows the service provider's document flow and lets you download the delivered XML, the buyer's response, the Tax Data Document and the FTA's response for your records. See [ASP and FTA lifecycle](#).

9. Go live

Switch the connection to **Taxilla Live** with the production credentials, create a live key, and send the first real invoice. Keep polling `status` or rely on the webhooks; automatic status checks continue for seven days per submission. Test documents are refused in live mode, so remove `isTest` and test keys from the production configuration.

Keep in reach

- The [integration pack](#): these guides as PDFs, the OpenAPI document, the Postman collection, the samples and the PINT AE workbook in one download for people who work offline or need to pass the material on.
- [Sample documents](#) to compare your XML field by field.
- [PINT AE field reference](#) and the [error reference](#).
- The [MCP server](#) and [CLI](#) for operating the same API from an assistant or a shell.