Fatoora Partner API — Integration Guide

Audience: integrators, ERPs, accounting firms and agencies subscribed to a **Fatoora Partner API** plan (`partner-starter`, `partner-business`, or `partner-max`) who want to submit, sign and track TEIF electronic invoices on behalf of their client organizations.

Not covered here: the separate **Fatoora Partner Signer** product line (`partner-signer`, `partner-signer-pro`), which is a local PKCS#11/browser signing bridge for ISVs embedding the FatooraSigner desktop agent — a different integration model (see Partner API vs. Partner Signer below).

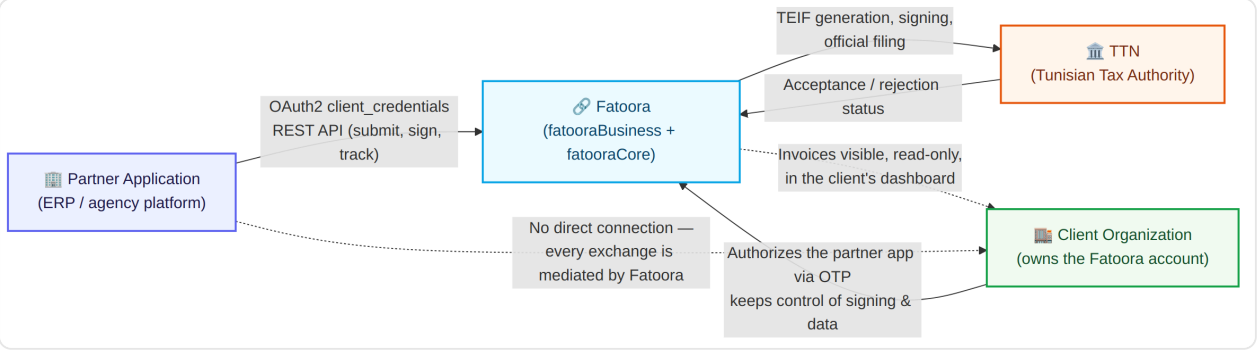
Production base URL: `https://business.fatoora.tn` — every endpoint in this guide (`/oauth/token`, `/api/core-proxy/*`, `/api/partner/*`) is served from this host (source: `fatooraSetup/.env.prod.example`, `VITE_FatooraBusinessBaseUrL`). Code samples below use `<base_url>` / `${BASE}` as a stand-in for this value.

Table of contents

---

1. How it works

Every Fatoora Partner API integration involves exactly **three parties**, and Fatoora always sits in the middle as the single point of trust — the partner and the client organization never exchange data or credentials directly:



- **You (the Partner Application)** — an ERP, agency platform, or accounting system integrating with Fatoora via OAuth2 to push invoices on behalf of your clients.

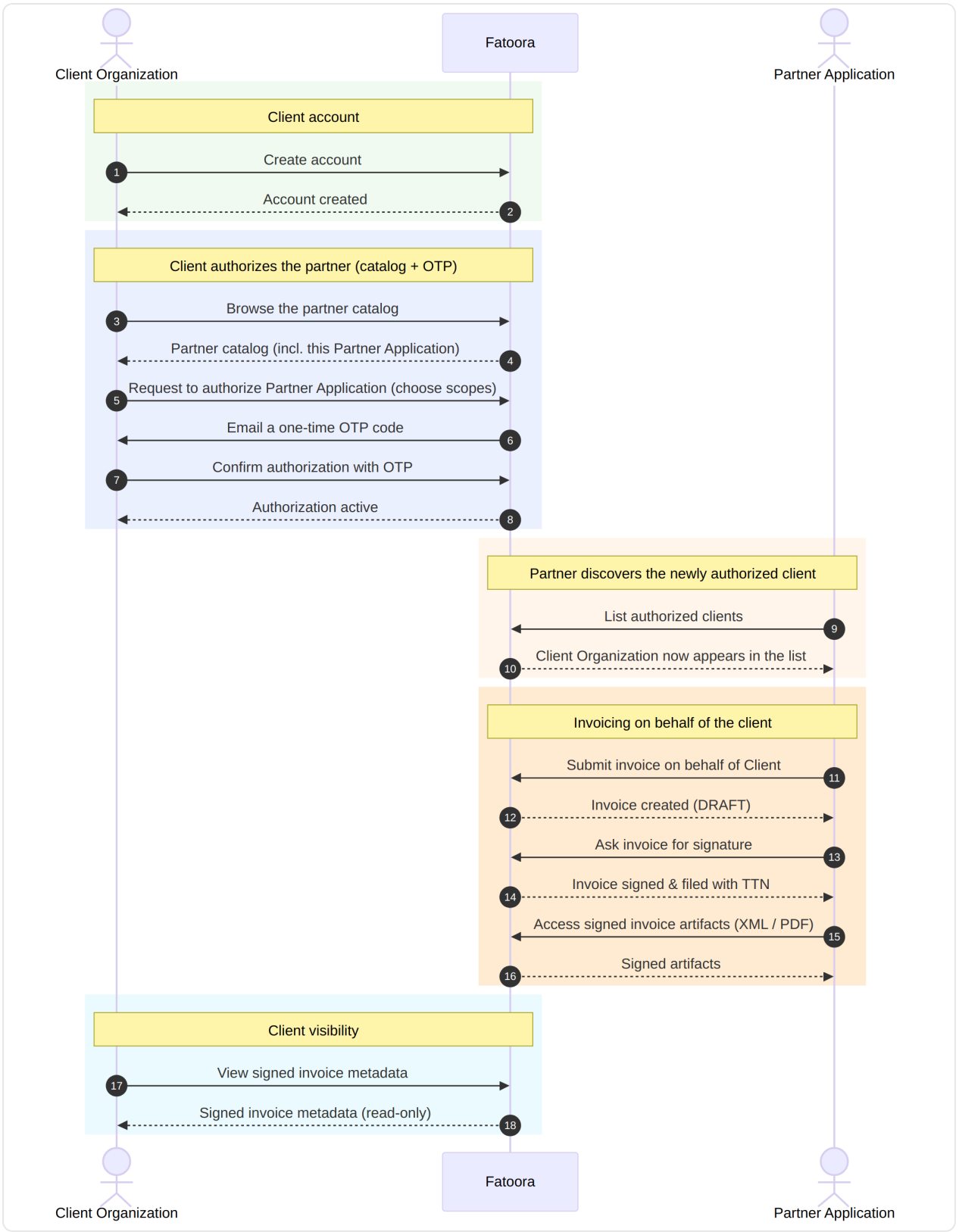
- **Fatoora** — shown here as a single system (internally split into `fatooraBusiness`, the API/BFF layer that issues your tokens and enforces authorization/scopes, and `fatooraCore`, which generates TEIF XML, signs invoices, and files them with the tax authority). From your integration's point of view, it is one API.
- **The Client Organization** — the business the invoice is actually issued for/by. They own their Fatoora account, their signing configuration, and the final invoice records. They authorize you, they can revoke you, and they see (read-only) everything you submit on their behalf.
- **TTN** (Tunisia Trade Net) — the Tunisian tax authority. Only Fatoora talks to TTN; neither you nor the client submits to TTN directly.

Four principles fall out of this model:

- Each client organization must explicitly **authorize your application** via a one-time-password (OTP) email confirmation sent to their own Fatoora dashboard — there is no shared login or impersonation.
- A client can **revoke your authorization at any time**.
- **Signing follows the client's own plan and settings** (manual, automatic SEAL, or DigiGO) — your application never handles the client's signing credentials, and cannot submit directly to TTN on their behalf; it can only trigger the client-configured sign-and-send flow.
- Invoices submitted through the API are visible in the client's Fatoora account but **cannot be edited by the client** from the UI — the API is the system of record for those documents.

Core interaction sequence

At the level of actors rather than API calls, every integration boils down to the same UML sequence: the client creates a Fatoora account, then browses Fatoora's partner catalog and authorizes the partner itself (confirmed by an OTP code — the partner is never contacted directly). The partner then discovers the newly authorized client by listing its clients, invoices and signs on the client's behalf, and the client keeps read access to the results.



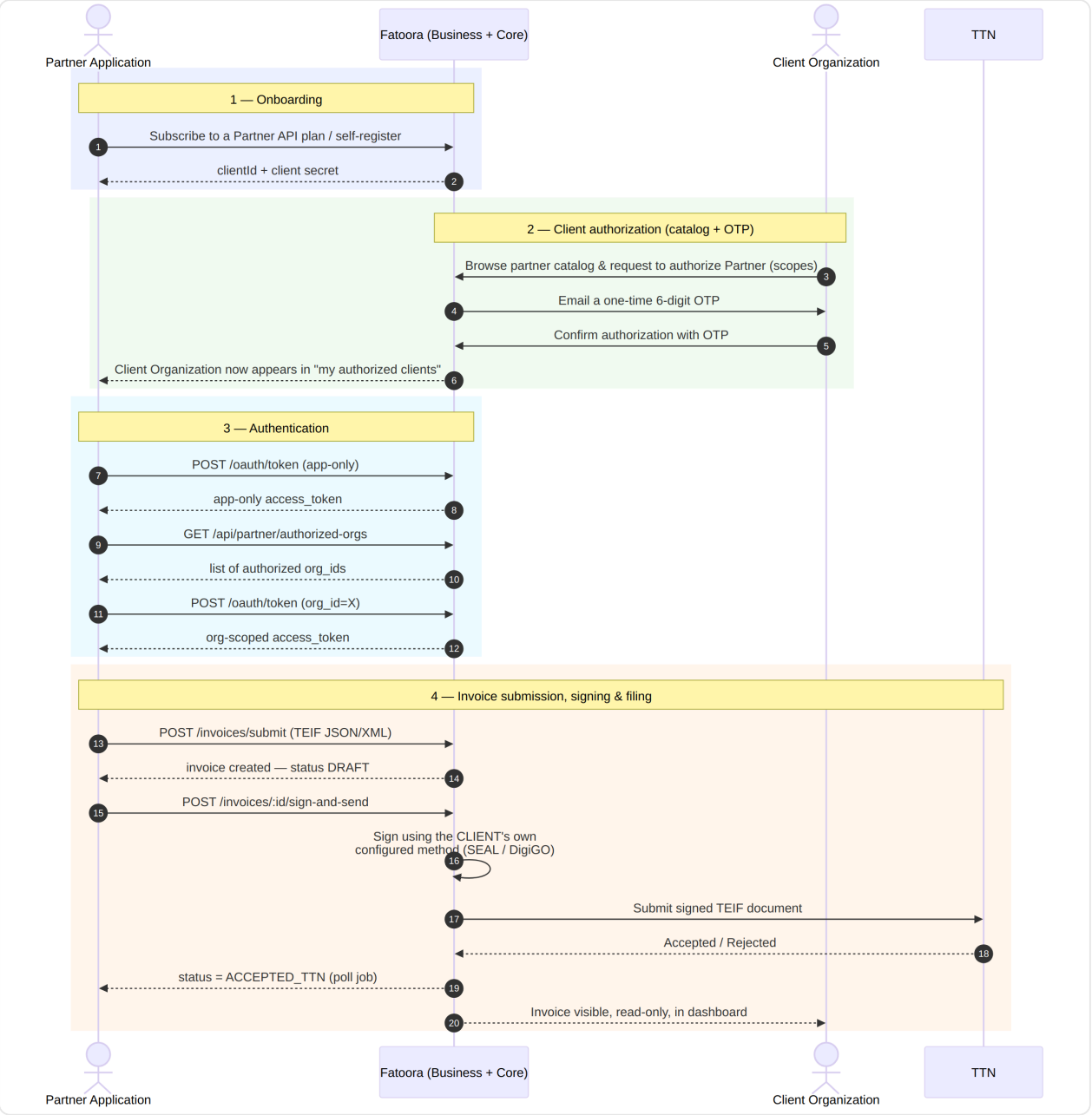
Step	Actor	Action
1	Client	Creates a Fatoora account

Step	Actor	Action
2	Client → Fatoora	Browses the partner catalog and requests to authorize the Partner Application (choosing scopes)
3	Client → Fatoora	Confirms the authorization with an OTP code emailed by Fatoora
4	Partner → Fatoora	Lists its authorized clients — the Client Organization now appears in that list
5	Partner → Fatoora	Submits an invoice on behalf of the client
6	Partner → Fatoora	Asks the invoice to be signed
7	Partner → Fatoora	Accesses the signed invoice artifacts (XML / PDF)
8	Client → Fatoora	Views and accesses the signed invoice's metadata, independently, from their own dashboard

> This is the simplified, actor-level view of the **client-initiated** authorization flow — the client finds the partner in Fatoora's catalog and grants access itself; the partner never contacts the client directly and only discovers the new authorization by listing its clients (step 4). A partner-initiated request flow also exists (partner asks first, client still approves by OTP) — see §2 Onboarding for both paths. Every partner call from step 4 onward is authenticated with an OAuth2 token (see §3 Authentication) — expanded below.

The full technical workflow, end to end

The sequence below walks through the same journey at the protocol level — every API call, from getting your credentials to a client seeing an accepted invoice in their dashboard. Each phase is expanded in its own section later in this guide.



Phase	What happens	Covered in
1 — Onboarding	You subscribe/register and Fatoora issues your `clientId` + client secret.	§2 Onboarding
2 — Client authorization	The client finds you in Fatoora's partner catalog and authorizes you itself, confirming with an OTP code emailed by Fatoora and choosing which scopes to grant (a partner-initiated request flow also exists, still OTP-gated). You then discover the newly authorized client by listing your clients.	§2 Onboarding
3 — Authentication	You exchange your credentials for an app-only token, discover which orgs authorize you, then exchange again for an org-scoped token.	§3 Authentication, §5

Phase	What happens	Covered in
		Discovering client organizations
4 — Submission, signing & filing	You submit a TEIF invoice, trigger sign-and-send, Fatoora signs it using the client's own configuration and files it with TTN, and the client sees the result in their dashboard.	§6 Submitting invoices, §8 Signing and TTN submission

Typical use case: a Tunisian travel agency generating 2,000 invoices/month for 80 hotel and transport partners integrates once with the Partner API and pushes invoices on behalf of each client without asking every client to touch their own Fatoora account.

---

2. Onboarding

Step 1 — Get your credentials

Subscribing to a Partner API plan (or calling `POST /api/partner/register` from an existing Fatoora account) immediately provisions:

- `clientId` — a public application identifier (e.g. `ak_partner_...`).
- `client secret` — an HMAC secret, shown **once**, at generation/regeneration time only.

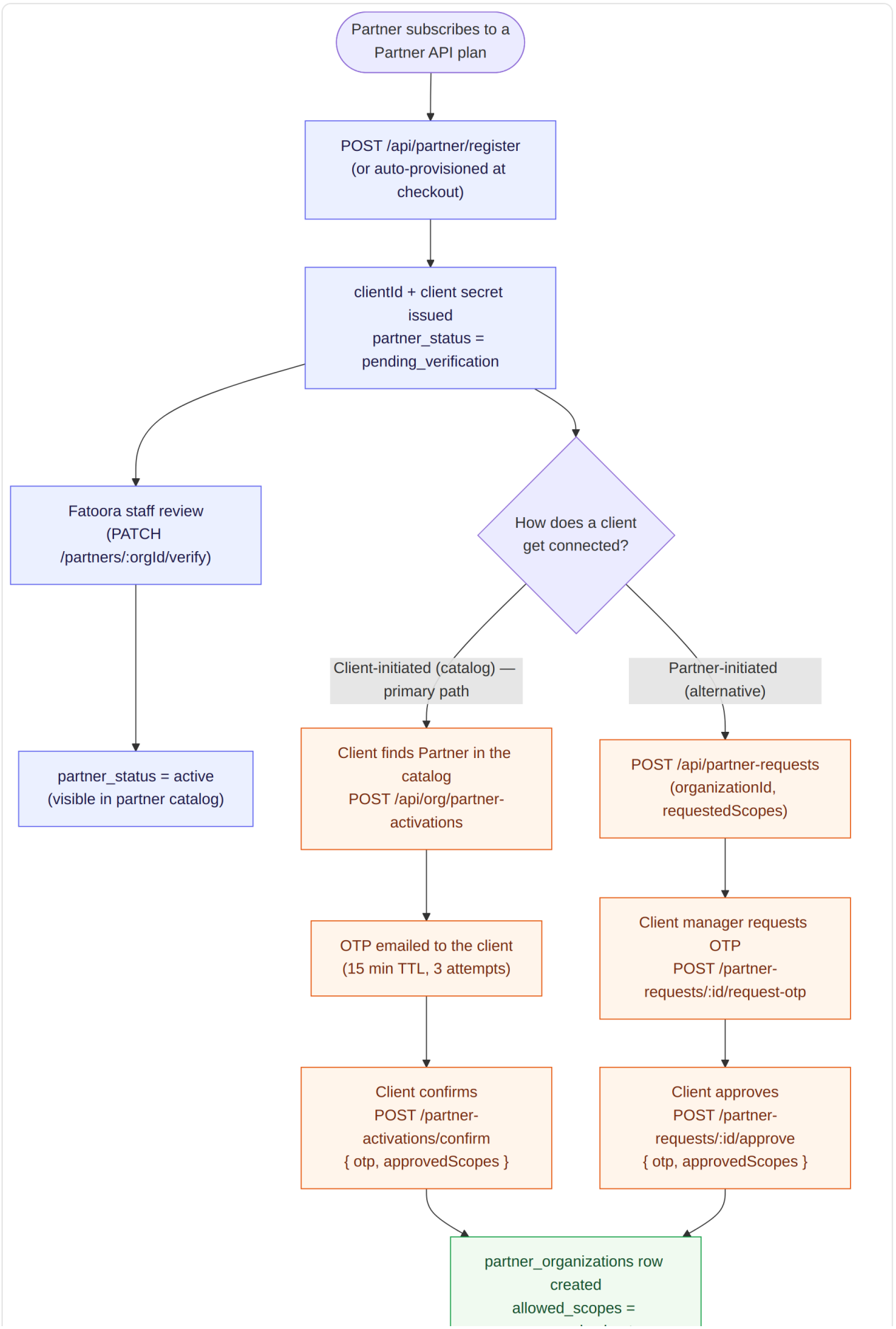
```
GET  /api/partner/credentials             → { clientId, partnerAppId, secretHint, allowedScopes, isActive }
POST /api/partner/credentials/regenerate  → { clientId, newSecret, warning }   # secret shown once, store it securely
```

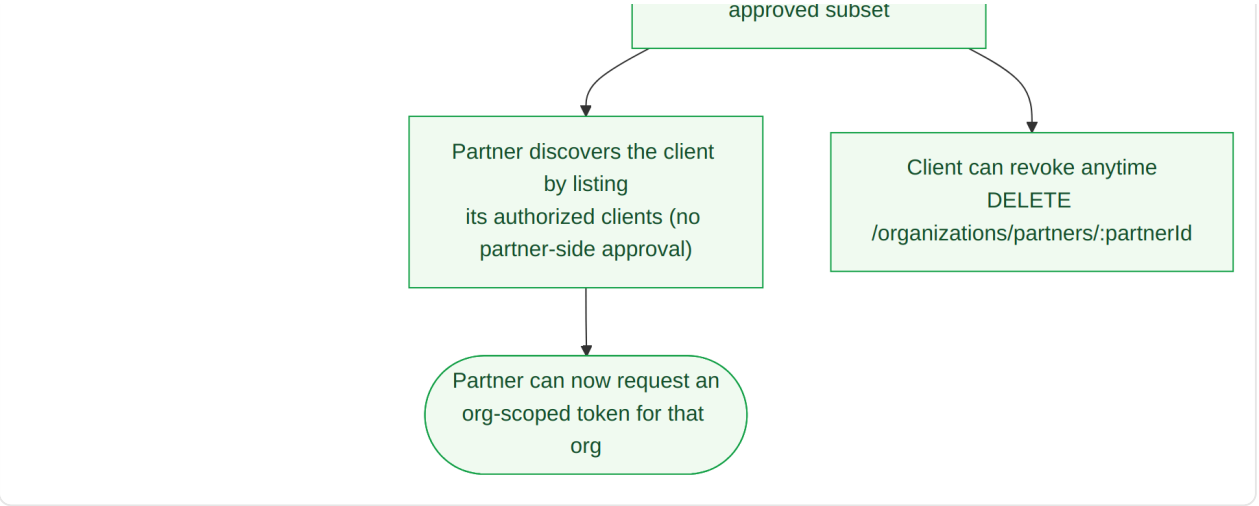
Your account is marked `pending_verification` until Fatoora staff review it — this does **not** block API calls, but it does keep you out of the public partner catalog until verified (`active`).

> Manage credentials from **Fatoora dashboard → Partner → Credentials**. There is no separate sandbox base URL — production (`https://business.fatoora.tn`) is used for all environments; talk to Fatoora support if you need an isolated test tenant.

Step 2 — Get each client organization to authorize your app

Two paths exist; both end with the client granting your app a specific, revocable set of scopes for their organization.


Partner subscribes to a Partner API plan
POST /api/partner/register (or auto-provisioned at checkout)
clientId + client secret issued partner_status = pending_verification
Fatoora staff review (PATCH /partners/:orgId/verify)
How does a client get connected?
partner_status = active (visible in partner catalog)
Client-initiated (catalog) — primary path
Partner-initiated (alternative)
Client finds Partner in the catalog POST /api/org/partner-activations
POST /api/partner-requests (organizationId, requestedScopes)
OTP emailed to the client (15 min TTL, 3 attempts)
Client manager requests OTP POST /partner-requests/:id/request-otp
Client confirms POST /partner-activations/confirm { otp, approvedScopes }
Client approves POST /partner-requests/:id/approve { otp, approvedScopes }
partner_organizations row created allowed_scopes =



A. (Primary path) The client finds you in the Fatoora partner catalog and authorizes you itself

The client browses the Fatoora partner catalog, selects your application, and requests to authorize it — you are never contacted directly and take no action at this stage. Fatoora emails the client's manager an OTP; once they confirm it, the authorization is created immediately and your app can discover the new client by listing its authorized clients (no approval step on your side).

```
POST /api/org/partner-activations                { partnerAppId, requestedScopes[] }
POST /api/org/partner-activations/confirm         { partnerAppId, otp, approvedScopes[] }
```

B. (Alternative) You request access, the client approves by OTP

If a client isn't browsing the catalog themselves, you can proactively request access to their organization; the same OTP confirmation is required from the client before the authorization becomes active.

```
POST /api/partner-requests                       { organizationId, requestedScopes[], message? }
POST /api/partner-requests/:id/request-otp        # client manager triggers a 6-digit OTP by email
(15 min TTL, 3 attempts)
POST /api/partner-requests/:id/approve            { otp, approvedScopes? }   # called by the
client, from their dashboard
```

In both cases, the resulting authorization grants your app a specific, revocable set of scopes for that one organization — and in both cases it's exclusively the client who approves, by OTP; your app never "accepts" anything. The client can revoke the authorization anytime:

```
DELETE /api/organizations/partners/:partnerId      { reason? }   # called by the client
```

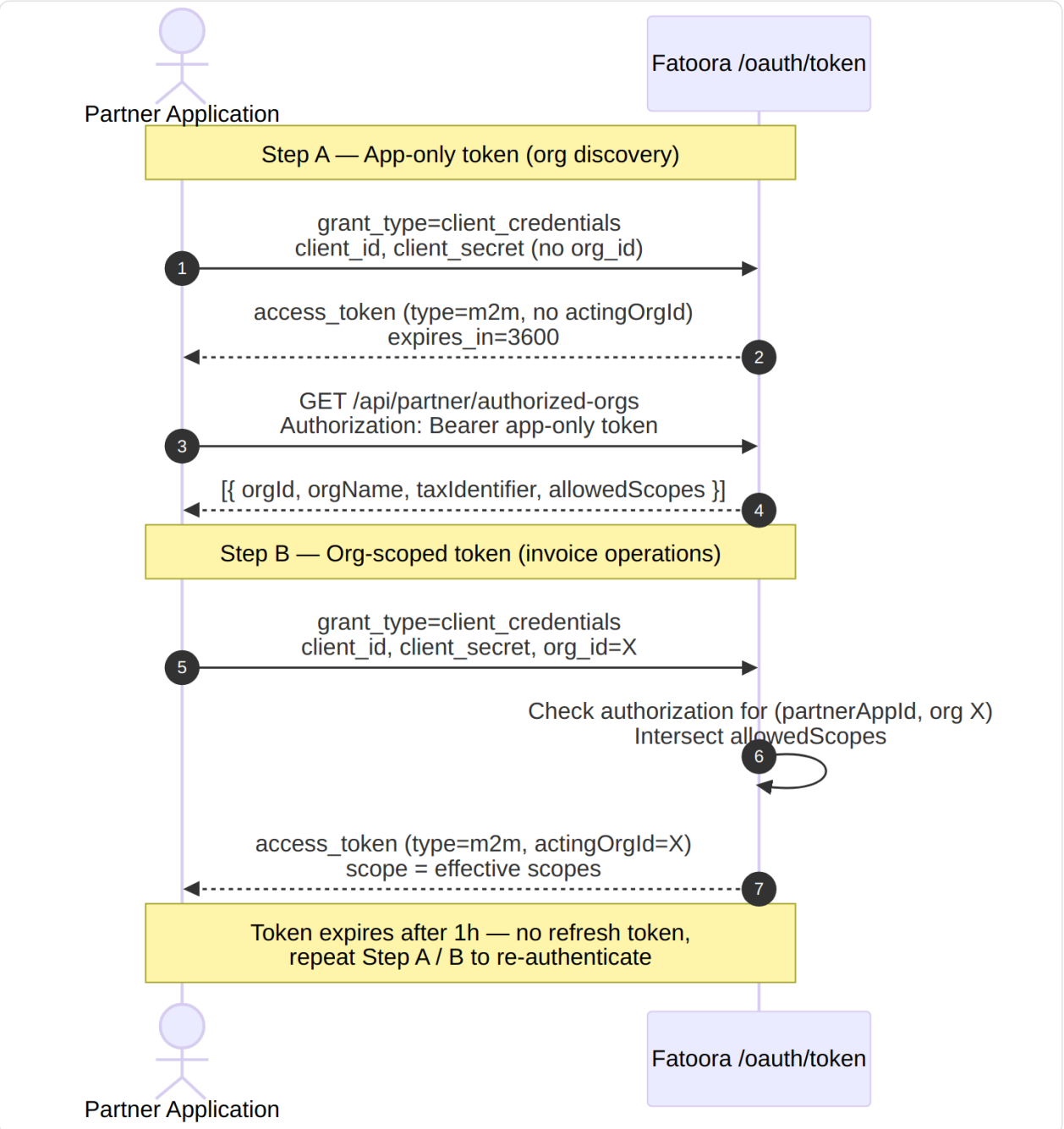
Step 3 — Start integrating

Once at least one client has authorized your app, proceed to Authentication and Discovering client organizations.

---

3. Authentication (OAuth2 client credentials)

Fatoora Partner API uses the OAuth2 **Client Credentials** grant (RFC 6749 §4.4). There are no user logins, cookies, or sessions involved — every call is authenticated with your `clientId` / `client secret` pair.



`POST /oauth/token`

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

grant_type    = client_credentials      (required)
client_id     = <your clientId>         (required)
client_secret = <your client secret>    (required)
org_id        = <client org UUID>       (optional — omit for an app-only token)
```

Two token flavors, both valid for exactly **1 hour** (`expires_in: 3600`), with **no refresh token** — simply call `/oauth/token` again when a token expires:

Token type	`org_id` sent?	Use it for
App-only token	No	`GET /api/partner/authorized-orgs`, `GET /api/partner/resolve-org` (org discovery only)
Org-scoped token	Yes	All invoice/signing endpoints, acting on behalf of that one organization

Success response:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "invoice:write invoice:read ttn:submit"
}
```

`scope` reflects the **effective** scopes. For an **org-scoped** token, that's the intersection of your app's `allowedScopes` and the scopes that specific client actually granted you. For an **app-only** token (no `org_id`), `scope` is just your app's own `allowedScopes` — confirmed live, this field is present on app-only tokens too, not only org-scoped ones. Requesting `org_id` for an organization that hasn't authorized your app (or has revoked it) returns a `403`.

Errors:

Status	`error`	When
400	`unsupported_grant_type`	`grant_type` is not `client_credentials`

Status	`error`	When
400	`invalid_request`	missing `client_id` / `client_secret`
401	`invalid_client`	unknown/inactive client, or wrong secret
403	`access_denied`	`org_id` given but not authorized for your app (revoked, never granted, or inactive)
429	`TOO_MANY_REQUESTS`	rate limit exceeded (see §13)
500	`server_error`	unexpected error

Using the token

Send it as a standard bearer token on every subsequent call:

```
Authorization: Bearer <access_token>
```

Under the hood, fatooraBusiness signs a JWT carrying your app id and (for org-scoped tokens) the target organization id and effective scopes; it forwards requests to fatooraCore over an independent, HMAC-signed service-to-service channel that re-validates your authorization and scopes on every call. You don't need to implement HMAC yourself — the REST endpoints below only require the bearer token.

---

4. Scopes

Scope	Grants
`invoice:write`	Create, update, submit, duplicate, delete draft invoices; import TEIF; trigger sign-and-send
`invoice:read`	List and retrieve invoices, statuses, PDFs, generated XML, jobs, audit activity
`ttn:submit`	Manually (re)submit a signed invoice to TTN or sync its TTN status (org-owned keys only — not usable by pure partner tokens for direct submission, see §11)
`seal:sign`	Trigger SEAL (server-side certificate) signing as part of sign-and-send
`digigo:sign`	Trigger DigiGO signing as part of sign-and-send

A client only grants the scopes they're comfortable with — check `allowedScopes` on the discovery endpoints (§5) before calling an endpoint that needs a scope you weren't granted; you'll get a `403 INSUFFICIENT_SCOPES` otherwise.

---

5. Discovering client organizations

Use your **app-only token** (no `org_id`) for these two endpoints.

`GET /api/partner/authorized-orgs`

Lists every organization that currently authorizes your app.

```
Authorization: Bearer <app-only token>
```

```
{
  "authorizedOrgs": [
    {
      "orgId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "orgName": "Société ABC SARL",
      "taxIdentifier": "1234567ABCD000",
      "allowedScopes": ["invoice:write", "seal:sign", "ttn:submit"],
      "authorizedAt": "2026-05-01 10:00:00.000000"
    }
  ],
  "total": 1
}
```

> `authorizedAt` **is a raw timestamp, not strict ISO 8601** — confirmed live it comes back as `"2026-05-11 20:59:43.542231"` (space-separated, microsecond precision, no `T` / `Z`), not `"2026-05-11T20:59:43Z"`. Most date libraries parse it fine, but don't validate it against a strict ISO-8601 regex.

`GET /api/partner/resolve-org?taxId=<matricule>`

Resolves a Tunisian tax ID (matricule fiscal) to the corresponding `orgId`, restricted to organizations that authorize your app. Accepts both formats:

- **8 characters** (short): `NNNNNNNL`
- **13 characters** (full): `NNNNNNNLCCCNNN`

Matching is bidirectional between the two formats (an 8-char query matches a 13-char stored ID by prefix, and vice versa).

```
{
  "orgId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "orgName": "Société ABC SARL",
  "taxIdentifier": "1234567ABCD000",
  "allowedScopes": ["invoice:write", "seal:sign"],
  "authorizedAt": "2026-05-01 10:00:00.000000"
}
```

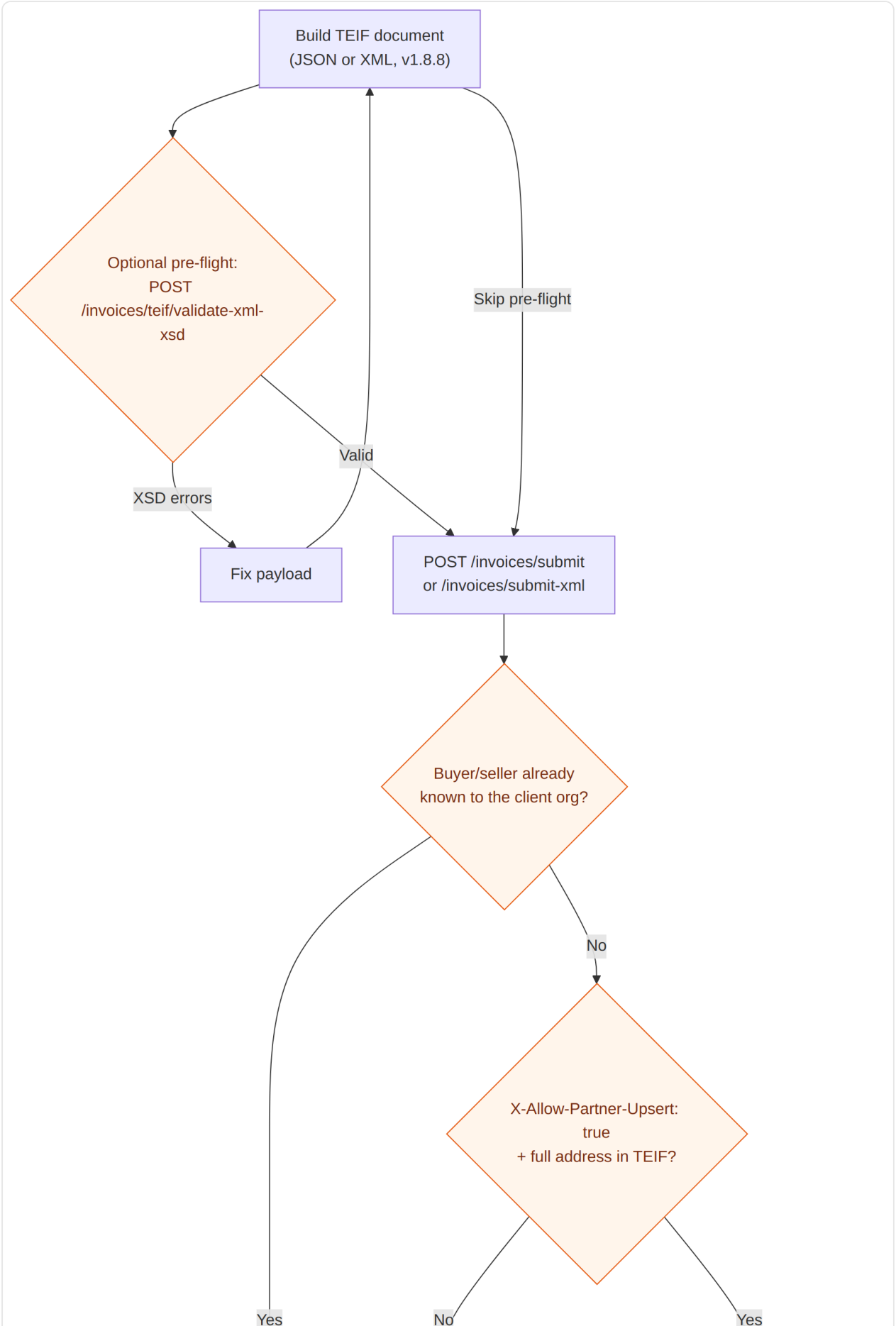
`404 ORG_NOT_FOUND` if no authorizing org matches that tax ID — the body also includes a descriptive `message` (confirmed live): `{ "error": "ORG_NOT_FOUND", "message": "Aucune organisation autorisée trouvée pour le matricule \"<taxId>\". Assurez-vous que l'organisation a bien activé votre application partenaire." }`. `400 INVALID_TAX_ID` if `taxId` is missing or under 4 characters.

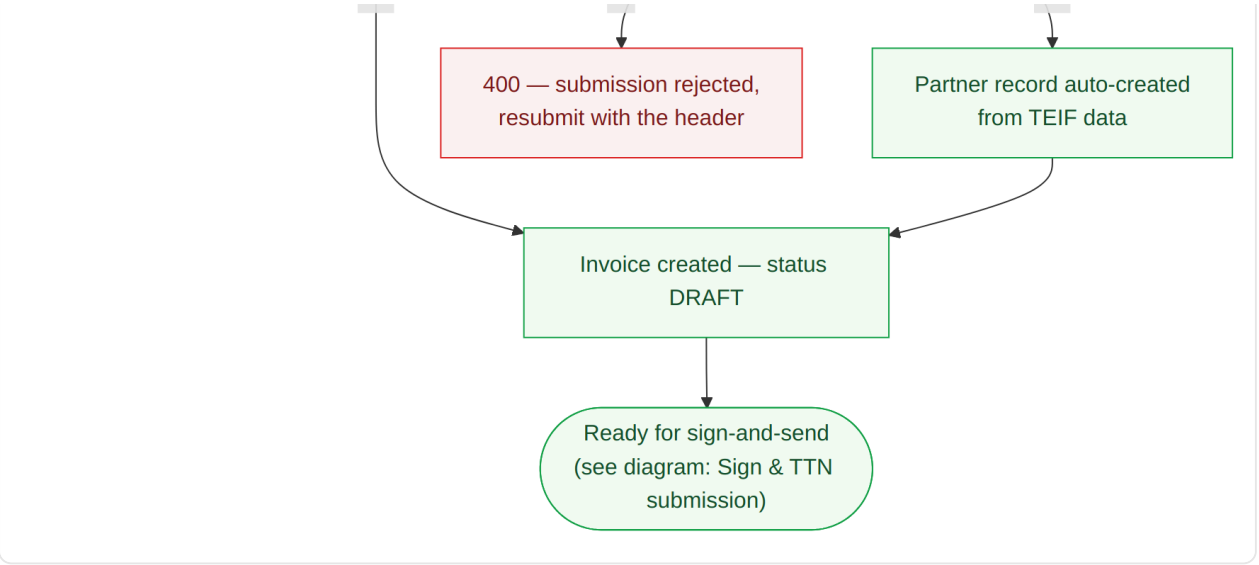
Once you have an `orgId`, request an **org-scoped token** (`POST /oauth/token` with `org_id=<orgId>`) and proceed to submit invoices for that organization.

---

6. Submitting invoices

TEIF (Tunisian Electronic Invoicing Format) is an XML-based standard, currently at **version 1.8.8**. You can submit either structured JSON (which Fatoora converts to TEIF XML) or a pre-built TEIF XML document directly.


Build TEIF document
(JSON or XML, v1.8.8)
Optional pre-flight:
POST
/invoices/teif/validate-xml-
xsd
Skip pre-flight
Valid
XSD errors
Fix payload
POST /invoices/submit
or /invoices/submit-xml
Buyer/seller already
known to the client org?
No
X-Allow-Partner-Upsert:
true
+ full address in TEIF?
Yes
No
Yes



`POST /api/core-proxy/invoices/submit` — JSON TEIF

Requires scope `invoice:write`.

```
Authorization: Bearer <org-scoped token>
Content-Type: application/json
X-Allow-Partner-Upsert: true      # optional — see note below
```

Minimal body:

```
{
  "version": "1.8.8",
  "controlingAgency": "TTN",
  "header": {
    "MessageSenderIdentifier":  { "@type": "I-01", "#text": "MATRICULE_SELLER" },
    "MessageRecieverIdentifier": { "@type": "I-01", "#text": "MATRICULE_BUYER" }
  },
  "body": {
    "bgm": { "documentIdentifier": "INV-2026-001", "documentTypeCode": "I-11", "documentType":
        "Facture" },
    "dtm": { "dateText": [{ "functionCode": "I-31", "format": "ddMMyy", "value": "120526" }] },
    "partnerSection": {
      "partnerDetails": [
        { "functionCode": "SE", "identifier": { "@type": "I-01", "#text": "MATRICULE_SELLER" },
          "name": "My Company Ltd", "address": { "streetAddress": "123 Av. Bourguiba", "city":
        "Tunis", "country": "TN" } },
        { "functionCode": "BY", "identifier": { "@type": "I-01", "#text": "MATRICULE_BUYER" },
          "name": "Client Ltd", "address": { "streetAddress": "456 Av. Liberté", "city": "Sfax",
        "country": "TN" } }
      ]
    },
    "linSection": {
      "lin": [{ "lineNumber": 1, "itemDescription": "Service rendered",
                "quantity": 1, "unitCode": "C62", "unitPrice": 1000.000,
                "netAmount": 1000.000, "taxRate": 19, "linAlc": [] }]
    },
    "invoiceMoa": {
      "amountDetails": [
        { "functionCode": "I-125", "amount": 1000.000 },
        { "functionCode": "I-176", "amount": 190.000 },
        { "functionCode": "I-77",  "amount": 1190.000 }
      ]
    },
    "invoiceTax": {
      "invoiceTaxDetails": [{ "categoryCode": "S", "taxRate": 19, "taxableAmount": 1000.000,
        "taxAmount": 190.000 }]
    },
    "pyt": { "paymentMeans": [{ "meansCode": "42" }] }
  }
}
```

Response — `201 Created` — the **full invoice object** (same shape returned by the list/detail endpoints, see §7), not a minimal envelope. Verified against a live dev instance:


```
{
  "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "orgId": "63b0c8c0-e0da-4797-a341-0564ad5e6c3a",
  "invoiceNumber": "FACT-2026-000001",
  "invoiceTypeCode": "I-11",
  "invoiceDate": "2026-07-12",
  "status": "VALIDATED",
  "senderName": "Ma Société SARL",
  "receiverName": "Mon Client SARL",
  "receiverIdentifierType": "I-01",
  "receiverIdentifierValue": "1234567RAM000",
  "totalAmountExclTax": 1000,
  "totalTaxAmount": null,
  "totalAmountInclTax": null,
  "currencyCode": "TND",
  "ttnGeneratedRef": null,
  "ttnIdSaveEfact": null,
  "createdAt": "2026-07-11T20:04:58.237547",
  "submittedAt": "2026-07-11T20:04:58.223037",
  "signedAt": null,
  "createdBy": "ak_partner_xxxxxxxxxxxxxxxxxxxx",
  "isQuotation": false,
  "quotationStatus": null,
  "duplicateCopy": false,
  "importSource": null,
  "importBlockedResign": false,
  "direction": "SENT",
  "partnerAppId": "partn_xxxxxxxxxxxx"
}
```


Status starts at `VALIDATED`, not `DRAFT`. Submission runs synchronous TEIF validation as part of the call — a successful `201` already reflects a validated invoice, ready for sign-and-send. `totalTaxAmount` / `totalAmountInclTax` may come back `null` immediately after a JSON submission (observed live) even though `totalAmountExclTax` is populated — re-fetch the invoice if you need the computed totals right away.

`POST /api/core-proxy/invoices/submit-xml` — raw TEIF XML

Same auth/scope, same full-invoice-object response shape as above. `Content-Type: application/xml`, body is the raw TEIF document.

```
curl -X POST "https://<base_url>/api/core-proxy/invoices/submit-xml" \
  -H "Authorization: Bearer ACCESS_TOKEN" \
  -H "Content-Type: application/xml" \
  --data-binary @invoice.xml
```

`POST /api/core-proxy/teif/validate` — validate only

> **Corrected path** — this is mounted at `/api/core-proxy/teif/validate`, **not** `/api/core-proxy/invoices/teif/validate-xml-xsd`. The latter is Core's *internal* path (`/api/v1/invoices/teif/validate-xml-xsd`) and 404s if called directly on the BFF — verified live; this was also wrong in the app's own generated Postman collection until fixed alongside this guide.

Body: `{ "xml": "<TEIF>...</TEIF>" }` (field is `xml`, not `xmlContent`) → runs XSD validation against the TEIF 1.8.8 schema without creating an invoice. Useful for pre-flight checks. Requires `invoice:read` or `invoice:write`.

Response:

```
{ "valid": true, "issues": [] }
```

or, on failure — `issues[]` items are `{ type, message, severity, xpath }`:

```
{
  "valid": false,
  "issues": [
    { "type": "XML_SCHEMA_VIOLATION", "message": "schema rule (cvc-)complex-type.2.4.a: Invalid
        content was found starting with element 'NotValid'. One of '{InvoiceHeader}' is
        expected.", "severity": "ERROR", "xpath": "line:1" }
  ]
}
```

The `X-Allow-Partner-Upsert` header

When the TEIF document references a buyer/seller (a "TEIF business partner") that doesn't yet exist in the client organization's partner registry, submission fails unless this header is set to `true` **and** the TEIF block includes a full address — in which case Fatoora auto-creates the missing partner record from the TEIF data. Set it on `submit` / `submit-xml` calls whenever you can't guarantee the counterparty already exists as a partner record in the client's account.

Document type codes (`documentTypeCode` in `bgm`)

Code	Meaning
`I-11`	Invoice (Facture)
`I-12`	Credit note (Avoir) — requires a `BillingReference` linking the original invoice
`I-13`	Proforma invoice
`I-14`	Advance invoice
`I-15`	Balance invoice
`I-16`	Other

7. Listing and retrieving invoices

All require `invoice:read` and an **org-scoped** token.

Method	Path	Notes
GET	`/api/core-proxy/invoices`	Paginated list, see filters below
GET	`/api/core-proxy/invoices/:id`	Full invoice detail — **wrapped response**, see below
~~GET~~	~~`/api/core-proxy/invoices/:id/status`~~	**Removed** — never had a real implementation, always returned `500`. Use `GET /api/core-proxy/invoices/:id` and read `invoice.status` instead; the path now returns a clean `404`.
GET	`/api/core-proxy/invoices/:id/generated-xml`	Returns the generated TEIF XML (`application/xml`)
GET	`/api/core-proxy/invoices/:`	Returns a PDF

Method	Path	Notes
	`id/pdf`	
GET	`/api/core-proxy/invoices/stats`	Dashboard-style aggregate stats
GET	`/api/partner/invoices`	Aggregated list across **all** organizations that authorize your app (session-authenticated dashboard use, not for the org-scoped M2M token)

`GET /api/core-proxy/invoices/:id` — full detail (wrapped)

Unlike list items, the single-invoice detail response **wraps** the invoice object alongside the generated TEIF payloads — confirmed against a live dev instance:

```
{
  "invoice": { /* same shape as a list item, see below — includes id, orgId, invoiceNumber,
        status, ... */ },
  "teifJson": "{\"body\":{...},\"header\":{...},\"version\":\"1.8.8\"}",
  "teifXml": "<TEIF controlingAgency=\"TTN\" version=\"1.8.8\">...</TEIF>",
  "ttnQrCodeBase64": null,
  "localSignedXml": null,
  "validationErrors": null,
  "ttnAcknowledgments": null
}
```

`teifJson` and `teifXml` are the generated TEIF document in both formats, already available without a separate call to `generated-xml`. Read the invoice fields (`status`, etc.) from the `invoice` sub-object, not the top level.

> **`GET /api/core-proxy/invoices/:id/status` has been removed.** It never had a real implementation on fatooraCore (no matching `@GetMapping` existed on `InvoiceController`), so it always returned `500 INTERNAL_ERROR`. Rather than implementing a redundant endpoint, the dead route was deleted outright — it now returns a clean `404 { "error": "NOT_FOUND" }`. Use `GET /api/core-proxy/invoices/:id` and read `invoice.status` instead.

Retrieving generated artifacts (XML / PDF)

```
# PDF — works regardless of the invoice's status (drafts included)
curl -X GET "https://<base_url>/api/core-proxy/invoices/<invoiceId>/pdf" \
  -H "Authorization: Bearer ACCESS_TOKEN" \
  -o invoice.pdf

# Generated TEIF XML
curl -X GET "https://<base_url>/api/core-proxy/invoices/<invoiceId>/generated-xml" \
  -H "Authorization: Bearer ACCESS_TOKEN"
```

Both require `invoice:read` and return the raw file directly (`application/pdf` binary, `application/xml` text) — not a JSON envelope.

> ⚠️ **`generated-xml` is not the signed/final document once an invoice has moved past `SIGNED`.** Internally, `TeifGenerationService.generateTeifXml` only returns the *stored* `teifXml` verbatim when it is still unsigned; the moment the stored XML contains a signature block (i.e. the invoice has been signed, and especially once TTN has accepted it and stamped a QR code), this endpoint instead **regenerates a fresh, unsigned XML on the fly from `teifJson`** — silently dropping the signature and the TTN QR code. For the actual final artifact (with signature, and QR code once TTN-accepted), read the `teifXml` field from `GET /api/core-proxy/invoices/:id` above instead — that field is the real persisted snapshot, not a live regeneration.

`GET /api/core-proxy/invoices/stats`

Dashboard-style aggregate stats for the org-scoped token. Response — confirmed live:

```
{
  "totalCount": 4,
  "sentCount": 0,
  "errorCount": 0,
  "totalRevenue": 2390,
  "byMonth": [
    { "month": "2026-05", "created": 2, "sent": 0, "errors": 0 },
    { "month": "2026-07", "created": 2, "sent": 0, "errors": 0 }
  ]
}
```

List filters (`GET /api/core-proxy/invoices`)

Query param	Notes
`direction`	`SENT` (issued by the org) or `RECEIVED`
`status`	one of the invoice status values
`startDate` / `endDate` (or `dateFrom` / `dateTo`)	`YYYY-MM-DD`
`senderMatricule` / `receiverMatricule`	tax ID filters
`page`	0-indexed, default `0`
`pageSize`	default `20`, clamped 1–100

Response:

```
{
  "content": [ /* invoice objects */ ],
  "page": 0,
  "pageSize": 20,
  "totalElements": 123,
  "totalPages": 7
}
```

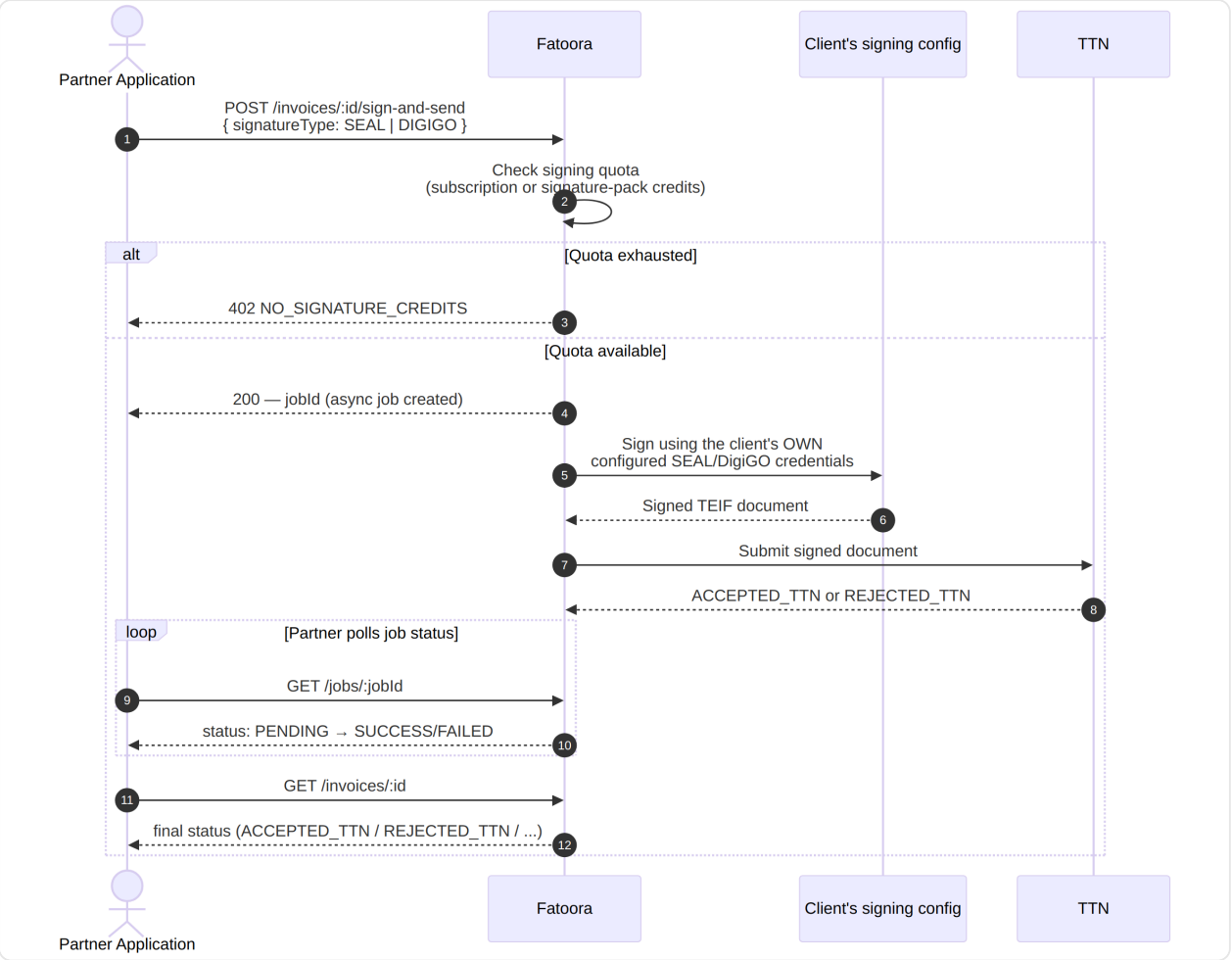
Each invoice object includes: `id`, `invoiceNumber`, `invoiceTypeCode`, `invoiceDate`, `status`, `senderName`, `receiverName`, `receiverIdentifierType`, `receiverIdentifierValue`, `totalAmountExclTax`, `totalTaxAmount`, `totalAmountInclTax`, `currencyCode`, `ttnGeneratedRef`, `createdAt`, `submittedAt`, `signedAt`, `direction`, `partnerAppId`.

> `listCreatorScope=ORG` and the associated `createdBy` / `humanCreatorsOnly` / `apiKeyCreatorsOnly` filters are **not available to partner tokens** — they're restricted to org-owned API keys (see §11).

---

8. Signing and TTN submission

Signing and TTN submission always follow **the client organization's own configuration** — your app triggers the process but never handles the client's signing credentials directly.



`POST /api/core-proxy/invoices/:id/sign-and-send`

The primary endpoint for the Partner API: signs the invoice and submits it to TTN in one asynchronous operation.

Requires `invoice:write`, plus `seal:sign` if `signatureType` is `SEAL`, or `digigo:sign` if `DIGIGO` (whichever the client has authorized and configured).

```
{ "signatureType": "SEAL" }
```

Response — `200 OK` (job created, not the final result). `jobType` is **provider-specific**, not a generic `"SIGN_AND_SEND"` — confirmed live (`SIGN_AND_SUBMIT_SEAL` for `signatureType: "SEAL"`, `SIGN_AND_SUBMIT_DIGIGO` for `"DIGIGO"`):

```
{
  "jobId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "invoiceId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "jobType": "SIGN_AND_SUBMIT_SEAL",
  "status": "PENDING",
  "success": true,
  "message": "Invoice queued for signature and TTN submission"
}
```


Poll the job until it completes:

```
GET /api/core-proxy/jobs/:jobId
GET /api/core-proxy/jobs/:jobId/logs?limit=50
```

`GET /jobs/:jobId` returns a richer shape than the initial POST response — confirmed live:

```
{
  "jobId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "invoiceId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "jobType": "SIGN_AND_SUBMIT_DIGIGO",
  "status": "FAILED",
  "progress": 0,
  "currentStep": "CERT_VERIFICATION",
  "errorMessage": null,
  "errorData": null,
  "outputData": {
    "step0CertVerification": {
      "error": "Credential ID DigiGO (email) non configuré pour cette organisation",
      "valid": false,
      "success": false,
      "errorCode": "MISSING_DIGIGO_CREDENTIAL_ID",
      "certificates": []
    }
  },
  "signatureResult": null,
  "ttnSubmissionResult": null,
  "ttnFinalResult": null,
  "createdAt": "2026-07-11T20:07:14.706Z",
  "startedAt": "2026-07-11T20:07:14.836Z",
  "completedAt": "2026-07-11T20:07:14.943Z",
  "retryCount": 0
}
```


`status` transitions `PENDING` → `COMPLETED` or `FAILED`. On failure, `outputData` names the failing step (`step0CertVerification` above shows a client org missing its DigiGO credential configuration — a common real-world failure, not a bug). `currentStep` values observed include `CERT_VERIFICATION`; expect further steps (signing, TTN submission) to appear here as the job progresses on success.

Once complete, re-fetch the invoice — `status` will be `ACCEPTED_TTN` on success, or one of the failure states (see Appendix).

Allowed source statuses: sign-and-send is only accepted when the invoice is currently `DRAFT`, `VALIDATED`, `VALIDATION_FAILED`, `SIGNATURE_FAILED`, `REJECTED_TTN`, `ERROR_TTN`, or `TIMEOUT_TTN` — otherwise you get a `409` conflict. It's also blocked on invoices imported with `importBlockedResign = true`.

`POST /api/core-proxy/invoices/batch/sign-and-send`

Same as above for multiple invoices at once:

```
{ "invoiceIds": ["inv_1", "inv_2"], "signatureType": "SEAL" }
```

Returns a `BatchJobResponse` with per-invoice job results — same shape as `POST /invoices/batch/delete` below, but with `JobCreated` entries instead of delete results.

`POST /api/core-proxy/invoices/batch/delete`

```
["inv_1", "inv_2"]
```

Response — confirmed live:

```
{
  "totalRequested": 2,
  "successCount": 2,
  "failedCount": 0,
  "results": [
    { "invoiceId": "inv_1", "success": true, "message": "Deleted successfully: FACT-2026-000001"
        },
    { "invoiceId": "inv_2", "success": true, "message": "Deleted successfully: FACT-2026-000002" }
  ]
}
```

Signing quota

Each sign-and-send consumes one unit of the client organization's monthly signing quota (or, if exhausted, one signature-pack credit). If both are exhausted you'll get:

```
{
  "error": "NO_SIGNATURE_CREDITS",
  "message": "No signing quota or signature-pack credits available. Buy a pack or
        subscribe/upgrade to sign and submit to TTN.",
  "packsRemaining": 0
}
```

with HTTP `402`. This is a **client-org** quota (tied to the client's own plan), not your Partner API plan's `monthlyDocLimit`.

9. Webhooks

Instead of polling, register a webhook to receive a push notification whenever an invoice you submitted (or one submitted by the client organization you act on behalf of) changes status. Webhook management uses the exact same org-scoped `access_token` as every other endpoint in this guide — `fatooraBusiness` resolves the target organization from the token itself, there is no separate `X-Org-ID` header to manage for these calls.

> **This capability was previously blocked for partner tokens** (`401 PARTNER_CONTEXT_NOT_ALLOWED`). It has since been opened up to authorized partner apps, using the same partner→organization authorization check already enforced on every invoice endpoint (`PartnerAuthorizationService.verifyPartnerAuthorization`) — an unauthorized attempt still returns `401 PARTNER_NOT_AUTHORIZED`. `event-delivery/mode` and the pull-notification endpoints (`GET/POST /api/core-proxy/notifications*`) remain org-owned-key-only for now.

Method	Path	Scope
POST	`/api/core-proxy/webhooks`	`invoice:write`
GET	`/api/core-proxy/webhooks`	`invoice:read`
POST	`/api/core-proxy/webhooks/:id/test`	`invoice:write`
DELETE	`/api/core-proxy/webhooks/:id`	`invoice:write`

Create a subscription

```
curl -X POST "https://<base_url>/api/core-proxy/webhooks" \
  -H "Authorization: Bearer ORG_SCOPED_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "url": "https://your-erp.example.com/webhooks/fatoora",
    "events": ["invoice.validated", "invoice.signed", "invoice.accepted_ttn",
        "invoice.rejected_ttn"]
  }'
```

Response — `201 Created`:

```
{
  "id": "wh_xxxxxxxxxxxxxxx",
  "appId": "partn_xxxxxxxxxxx",
  "url": "https://your-erp.example.com/webhooks/fatoora",
  "events": ["invoice.validated", "invoice.signed", "invoice.accepted_ttn",
        "invoice.rejected_ttn"],
  "status": "ACTIVE",
  "secret": "whsec_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "expiresAt": "2026-08-14T00:00:00",
  "createdAt": "2026-07-15T00:00:00"
}
```

The `secret` is returned **only once**, at creation time — save it, it's needed to verify the signature of every delivered event. `events` defaults to `invoice.created`, `invoice.validated`, `invoice.signed`, `invoice.submitted_ttn`, `invoice.accepted_ttn`, `invoice.rejected_ttn`, `webhook.test` when omitted; add `invoice.validation_failed` explicitly if you also need it (not included by default). A webhook subscription created by your partner app only fires for the organizations it is authorized on and only lists/deletes/tests the subscriptions your own partner app created (not those created by the client organization's own API key, nor by another partner).

Verifying the delivered payload

Every delivery includes `X-Webhook-Id`, `X-Event-Id`, `X-Timestamp` (epoch ms) and `X-HMAC-Signature` (Base64, `HmacSHA256`) headers, computed as:

```
canonical = "POST:" + path + ":" + sha256Hex(body) + ":" + timestamp
X-HMAC-Signature = base64(HMAC-SHA256(canonical, webhook_secret))
```

where `path` is the raw path of your webhook URL (or `/` if empty) and `body` is the exact raw request body bytes.

Example delivered payload:

```
{
  "eventId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "eventType": "invoice.accepted_ttn",
  "occurredAt": "2026-07-15T10:00:00Z",
  "appId": "partn_xxxxxxxxxxxx",
  "resource": { "type": "invoice", "id": "inv_xxxxxxxxxxxxxxxx" },
  "data": {
    "invoiceId": "inv_xxxxxxxxxxxxxxxx",
    "invoiceStatus": "ACCEPTED_TTN",
    "invoiceNumber": "FACT-2026-000001"
  },
  "customData": {}
}
```

Delivery is single-attempt, fire-and-forget — there is no automatic retry on failure. Use `POST /webhooks/:id/test` (fires a `webhook.test` event) to validate your receiver, including signature verification, before relying on real invoice events.

List and manage subscriptions

```
curl -X GET "https://<base_url>/api/core-proxy/webhooks" \
  -H "Authorization: Bearer ORG_SCOPED_TOKEN"

curl -X POST "https://<base_url>/api/core-proxy/webhooks/wh_xxxxxxxxxxxxxxxx/test" \
  -H "Authorization: Bearer ORG_SCOPED_TOKEN"

curl -X DELETE "https://<base_url>/api/core-proxy/webhooks/wh_xxxxxxxxxxxxxxxx" \
  -H "Authorization: Bearer ORG_SCOPED_TOKEN"
```

10. Quotations

The same core-proxy surface supports draft quotations (devis) that can later be converted into invoices — useful if your integration needs a "pending approval" stage before invoicing.

Method	Path	Scope
POST	`/api/core-proxy/quotations/draft`	`invoice:write`

Method	Path	Scope
GET	`/api/core-proxy/quotations`	`invoice:read`
GET	`/api/core-proxy/quotations/:id`	`invoice:read`
PUT	`/api/core-proxy/quotations/:id`	`invoice:write` (draft only)
PATCH	`/api/core-proxy/quotations/:id/status`	`invoice:write` — body `{ "status": "ACCEPTED" | "REFUSED" | "CANCELLED" }`
POST	`/api/core-proxy/quotations/:id/convert-to-invoice`	`invoice:write`

All four return the same full invoice object described in §6, with quotation-specific fields set — confirmed live:

- `POST /draft` → `status: "DRAFT"`, `isQuotation: true`, `quotationStatus: "CREATED"` (unlike a regular invoice submit, which starts at `VALIDATED` — quotations skip validation until conversion)
- `PATCH /:id/status` → same object with `quotationStatus` updated to `ACCEPTED` / `REFUSED` / `CANCELLED`
- `POST /:id/convert-to-invoice` → a **new** invoice object (different `id`), `isQuotation: false`, `quotationStatus: null`, `status: "DRAFT"`, `createdBy: null` (not the partner's `clientId` — worth noting if you rely on that field to filter your own submissions)

---

11. What partner apps cannot do

By design, several capabilities are restricted to organization-owned API keys and are unavailable to a pure partner (M2M, org-scoped) token — even with all scopes granted:

- **Direct TTN submission** — `POST /api/core-proxy/invoices/:id/submit-ttn` returns `403 PARTNER_NOT_ALLOWED` for partner tokens. Always use `sign-and-send` instead, which routes signing + submission through Fatoora.
- **Event-delivery mode and pull notifications** — `GET/PUT /api/core-proxy/event-delivery/mode` and `GET/POST /api/core-proxy/notifications*` return `401 PARTNER_CONTEXT_NOT_ALLOWED` for partner tokens. Use webhooks instead (now available to authorized partner apps) to track invoice status changes in real time.
- **Organization-scoped list filters** — `listCreatorScope=ORG` and the `createdBy` / `humanCreatorsOnly` / `apiKeyCreatorsOnly` list filters are blocked for partner

tokens (`401 LIST_ORG_SCOPE_NOT_ALLOWED`).

- **Editing client invoices from the Fatoora UI** — once submitted via your API, invoices are read-only to the client in their own dashboard; they can view and export but not edit.
- **Acting without an authorized `org_id`** — an app-only token can only call the two discovery endpoints (§5); every invoice/signing/webhook endpoint requires an org-scoped token for an organization that has actively authorized your app and granted the relevant scope.

---

12. Error handling

OAuth errors (`/oauth/token`)

```
{ "error": "invalid_client", "error_description": "Invalid client credentials." }
```

Core-proxy / BFF errors

```
{ "error": "INSUFFICIENT_SCOPES", "message": "Required scopes: invoice:write", "requiredScopes":
        ["invoice:write"] }
```

fatooraCore errors

```
{
  "code": "INSUFFICIENT_SCOPES",
  "error": "INSUFFICIENT_SCOPES",
  "message": "This service app does not have invoice:write permission",
  "details": [],
  "path": "/api/v1/invoices/submit",
  "timestamp": "2026-07-11T12:00:00Z",
  "status": 401
}
```

Common error codes

Code	HTTP	Meaning / fix
`invalid_client`	401	Wrong `client_id` / `client_secret`

Code	HTTP	Meaning / fix
`access_denied`	403	Org not authorized, or app is not a partner app
`INSUFFICIENT_SCOPES`	401/403	Request the missing scope from the client (re-approval flow)
`ORG_NOT_FOUND`	404	Tax ID doesn't match an org that authorized your app
`NO_ORGANIZATION`	400	Org-scoped token required but missing/invalid
`PARTNER_APP_NOT_FOUND`	401	Service app record missing for this token
`PARTNER_NOT_ALLOWED`	403	Endpoint not available to partner tokens (e.g. direct TTN submit)
`PARTNER_CONTEXT_NOT_ALLOWED`	401	Event-delivery-mode / pull-notification endpoints only — org-owned keys only. Webhooks themselves no longer return this for authorized partner apps, see §9
`PARTNER_NOT_AUTHORIZED`	401	The client organization has not authorized your partner app for this scope/endpoint (webhooks included)
`VALIDATION_ERROR` / `INVOICE_VALIDATION_ERROR`	400/422	TEIF payload fails schema/business validation — check `details[]`
`NO_SIGNATURE_CREDITS`	402	Client org's signing quota and signature packs are exhausted
`SUBSCRIPTION_INACTIVE`	402	Your own Partner API subscription is paused/cancelled/expired
`RATE_LIMIT_EXCEEDED` / `TOO_MANY_REQUESTS`	429	Back off and retry (see §13)

13. Rate limits and quotas

Limiter	Scope	Limit
`POST /oauth/token`	per IP	30 requests / 15 min
All other `/api/*` calls (including all invoice/quotation/job endpoints)	per IP	300 requests / 15 min (configurable server-side)

Rate-limit responses include a `Retry-After` header and:

```
{ "error": "RATE_LIMIT_EXCEEDED", "message": "Too many requests, please try again later.",
        "retryAfter": 900 }
```

Plan-level quotas (independent of rate limits):

- `monthlyDocLimit` caps the total invoices you submit across all client orgs per calendar month (varies by your Partner API plan tier: starter/business/max).
- `maxOrgs` caps how many client organizations you can have authorized at once (also tier-dependent).
- Each client organization's own **signing quota** (subscription-based or signature-pack credits) separately gates `sign-and-send` calls for that org — see §8.
- During the trial period, every partner plan gets a fixed cap of **100 documents**, regardless of the plan's normal monthly limit.

---

14. Code examples

Get an app-only token, then discover organizations

```
# 1. App-only token (no org_id)
curl -X POST "https://<base_url>/oauth/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=client_credentials" \
  -d "client_id=YOUR_CLIENT_ID" \
  -d "client_secret=YOUR_CLIENT_SECRET"

# 2. List authorized organizations
curl -X GET "https://<base_url>/api/partner/authorized-orgs" \
  -H "Authorization: Bearer APP_ONLY_TOKEN"
```

```
// 1. App-only token
const tokenRes = await fetch(`${BASE}/oauth/token`, {
  method: 'POST',
  headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
  body: new URLSearchParams({
    grant_type: 'client_credentials',
    client_id: 'YOUR_CLIENT_ID',
    client_secret: 'YOUR_CLIENT_SECRET',
  }),
});
const { access_token } = await tokenRes.json();

// 2. Cache the authorized orgs, keyed by tax ID
const { authorizedOrgs } = await (await fetch(`${BASE}/api/partner/authorized-orgs`, {
  headers: { Authorization: `Bearer ${access_token}` },
})).json();
const taxIdMap = Object.fromEntries(authorizedOrgs.map(o => [o.taxIdentifier.toUpperCase(),
        o.orgId]));
```

```
import requests

token = requests.post(f"{BASE}/oauth/token", data={
    "grant_type": "client_credentials",
    "client_id": "YOUR_CLIENT_ID",
    "client_secret": "YOUR_CLIENT_SECRET",
}).json()["access_token"]

orgs = requests.get(f"{BASE}/api/partner/authorized-orgs",
                    headers={"Authorization": f"Bearer {token}"}).json()
```

Get an org-scoped token and submit an invoice

```
curl -X POST "https://<base_url>/oauth/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=client_credentials" \
  -d "client_id=YOUR_CLIENT_ID" \
  -d "client_secret=YOUR_CLIENT_SECRET" \
  -d "org_id=TARGET_ORG_UUID"

curl -X POST "https://<base_url>/api/core-proxy/invoices/submit" \
  -H "Authorization: Bearer ACCESS_TOKEN" \
  -H "Content-Type: application/json" \
  -d @invoice.json
```

```
token = requests.post(f"{BASE}/oauth/token", data={
    "grant_type": "client_credentials",
    "client_id": "YOUR_CLIENT_ID",
    "client_secret": "YOUR_CLIENT_SECRET",
    "org_id": "TARGET_ORG_UUID",
}).json()["access_token"]

res = requests.post(f"{BASE}/api/core-proxy/invoices/submit",
                    headers={"Authorization": f"Bearer {token}"},
                    json=invoice)
print(res.json())
```

Sign and send to TTN

```
curl -X POST "https://<base_url>/api/core-proxy/invoices/INVOICE_ID/sign-and-send" \
  -H "Authorization: Bearer ORG_SCOPED_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"signatureType":"SEAL"}'
```

```
const signRes = await fetch(`${BASE}/api/core-proxy/invoices/${invoiceId}/sign-and-send`, {
  method: 'POST',
  headers: { Authorization: `Bearer ${orgScopedToken}`, 'Content-Type': 'application/json' },
  body: JSON.stringify({ signatureType: 'SEAL' }),
});
const { jobId } = await signRes.json();
// then poll GET /api/core-proxy/jobs/{jobId} until status is terminal
```

A ready-to-import Postman collection with all of the above pre-wired (including test scripts that auto-populate `access_token` / `org_id`) is available from **Fatoora dashboard → Partner → Credentials → API guide → Download Postman collection**.

---

15. Partner API vs. Partner Signer

These are two distinct product lines — pick the one that matches your integration model:

	Partner API (this guide)	**Partner Signer**
Plans	`partner-starter/business/max`	`partner-signer`, `partner-signer-pro`
Integration model	Stateless server-to-server REST API (OAuth2 client_credentials)	Local PKCS#11/browser signing bridge embedded in your desktop/web app, talking to the FatooraSigner agent
What you submit	Full TEIF invoices, on behalf of clients	Nothing — you only broker local signing sessions for pre-registered client tax IDs
Client onboarding	OTP-based authorization request/approval per organization	Pre-register client tax IDs directly (`POST /api/partner/signer-orgs`), no OTP step
Domain restriction	None	Requests must originate from a whitelisted domain

If your product needs both (some clients integrate via API, others need local signing), subscribe with `partnerType: "both"` during registration — this unlocks both capability sets on your account, subject to your plan's API-integration support.

---

16. Appendix — reference tables

Invoice status values

Status	Meaning
`DRAFT`	Unvalidated draft
`SUBMITTED`	Submitted for validation
`VALIDATED`	Validated, ready for signature
`VALIDATION_FAILED`	Validation failed
`SIGNED`	Signed, ready for TTN
`SIGNATURE_FAILED`	Signature failed
`SENT_TTN`	Sent to TTN, awaiting response
`ACCEPTED_TTN`	Accepted by TTN
`REJECTED_TTN`	Rejected by TTN
`ERROR_TTN`	TTN sync error
`TIMEOUT_TTN`	TTN response timeout
`CANCELLED`	Cancelled
`ARCHIVED`	Archived, out of the active flow

MOA (amount) function codes used in TEIF JSON

Code	Meaning
`I-125` / `I-172`	Total excl. tax (net taxable amount)
`I-176`	Total VAT / taxable base
`I-77` / `I-180`	Total incl. tax (amount due)
`I-181`	Total taxes
`I-131`	Total discounts

Scopes quick reference

Scope	Endpoints gated
`invoice:write`	submit, submit-xml, draft, update, delete, duplicate, import, sign-and-send, batch operations
`invoice:read`	list, get, status, PDF, generated XML, stats, jobs, audit
`ttn:submit`	manual `submit-ttn` / `sync-ttn` (org-owned keys only)
`seal:sign`	sign-and-send with `signatureType: "SEAL"`
`digigo:sign`	sign-and-send with `signatureType: "DIGIGO"`

Base structure of a TEIF JSON document

```
{
  "version": "1.8.8",
  "controlingAgency": "TTN",
  "header": { "MessageSenderIdentifier": {}, "MessageRecieverIdentifier": {} },
  "body": {
    "bgm": {},
    "dtm": {},
    "partnerSection": { "partnerDetails": [] },
    "linSection": { "lin": [] },
    "invoiceMoa": { "amountDetails": [] },
    "invoiceTax": { "invoiceTaxDetails": [] },
    "pyt": {}
  }
}
```

---

17. OpenAPI spec and Postman collection

OpenAPI specification

fatooraBusiness serves a live, generic OpenAPI 3.0 JSON document at `GET https://business.fatoora.tn/open-api` (source: `fatooraBusiness/src/openapi.ts`, mounted in `src/index.ts`). At the time of writing it documents only the general platform API

(`/api/auth/*`, `/api/plans`, `/api/org`, `/api/subscription`, `/api/teif`, `/api/admin/auth/login`, …) — it does **not** yet include the Partner API routes (`/oauth/token`, `/api/core-proxy/*`, `/api/partner/*`) covered in this guide.

The specification below was authored from every endpoint documented throughout this guide, so partners have a machine-readable spec they can import into Postman, Insomnia, or an OpenAPI code generator today. It's a candidate to merge into `fatooraBusiness/src/openapi.ts` so the live `/open-api` endpoint eventually covers the Partner API natively.

Also saved as a standalone file: `partner-api-openapi.yaml`.

```yaml
openapi: 3.0.0

info:

  title: Fatoora Partner API

  version: "1.0"

  description: >
    REST API for Fatoora Partner API integrators (partner-starter / partner-business / partner-max
      plans).
    Authenticate with OAuth2 client_credentials, discover authorized client organizations, then
      submit,
    sign and track TEIF invoices on their behalf. Not authored from a single generated source file
      —
    fatooraBusiness's live spec at GET /open-api currently documents only the general platform API
      and
    does not yet include these partner routes; this file was written to close that gap and is a
      candidate
    to merge into fatooraBusiness/src/openapi.ts. Endpoint paths, field names and response shapes
      below
    were verified against a live dev instance (fatooraLive :3600 / fatooraBusiness :4100) on 2026-
      07-11 —

    see inline notes for corrections found during that pass.

  contact:

    name: Fatoora Support

    email: support@fatoora.tn

servers:

  - url: https://business.fatoora.tn

    description: Production

  - url: https://staging-business.fatoora.tn

    description: Staging (if provisioned for your account)


security:

  - partnerOAuth: []


paths:

  /oauth/token:

    post:

      summary: Exchange client credentials for an access token

      description: >
        Omit org_id for an app-only token (org discovery only). Include org_id for an org-scoped
        token
        (invoice/signing endpoints), valid only if the target organization currently authorizes
        this app.

      security: []

      requestBody:

        required: true

        content:

          application/x-www-form-urlencoded:
```

```
            schema:
              type: object
              required: [grant_type, client_id, client_secret]
              properties:
                grant_type:
                  type: string
                  enum: [client_credentials]
                client_id:
                  type: string
                client_secret:
                  type: string
                org_id:
                  type: string
                  format: uuid
                  description: Optional — omit for an app-only token
        responses:
          "200":
            description: Token issued
            content:
              application/json:
                schema:
                  $ref: "#/components/schemas/TokenResponse"
          "400":
            description: unsupported_grant_type / invalid_request
            content:
              application/json:
                schema:
                  $ref: "#/components/schemas/OAuthError"
          "401":
            description: invalid_client
            content:
              application/json:
                schema:
                  $ref: "#/components/schemas/OAuthError"
          "403":
            description: access_denied — org not authorized for this app
            content:
              application/json:
                schema:
                  $ref: "#/components/schemas/OAuthError"
```

```
      "429":
        description: Rate limit exceeded (30 req / 15 min per IP)

  /api/partner/authorized-orgs:
    get:
      summary: List organizations that authorize this partner app
      description: Requires an app-only bearer token (no org_id).
      responses:
        "200":
          description: OK
          content:
            application/json:
              schema:
                type: object
                properties:
                  authorizedOrgs:
                    type: array
                    items:
                      $ref: "#/components/schemas/AuthorizedOrg"
                  total:
                    type: integer
        "401":
          $ref: "#/components/responses/Unauthorized"

  /api/partner/resolve-org:
    get:
      summary: Resolve a Tunisian tax ID to an org_id
      description: Requires an app-only bearer token (no org_id).
      parameters:
        - name: taxId
          in: query
          required: true
          schema:
            type: string
          description: 8-char (NNNNNNNL) or 13-char (NNNNNNNLCCCNNN) matricule fiscal
      responses:
        "200":
          description: OK
          content:
            application/json:
```

```
              schema:
                $ref: "#/components/schemas/AuthorizedOrg"
        "400":
          description: INVALID_TAX_ID
        "404":
          description: ORG_NOT_FOUND


  /api/core-proxy/invoices/submit:
    post:
      summary: Submit a TEIF invoice (JSON)
      description: Requires scope invoice:write and an org-scoped bearer token.
      security: [{ partnerOAuth: [invoice:write] }]
      parameters:
        - name: X-Allow-Partner-Upsert
          in: header
          required: false
          schema:
            type: string
            enum: ["true", "false"]
          description: Auto-create an unknown buyer/seller partner record from the TEIF data
        (requires a full address in the TEIF block)
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/TeifInvoiceJson"
      responses:
        "201":
          description: >
            Invoice created. Status is "VALIDATED" immediately (synchronous validation runs as

            part of submit) — verified live; NOT "DRAFT" as an earlier draft of this spec claimed.
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/InvoiceSummary"
        "400":
          description: TEIF validation error
        "401":
          $ref: "#/components/responses/Unauthorized"
        "403":
```

```
          $ref: "#/components/responses/Forbidden"


  /api/core-proxy/invoices/submit-xml:
    post:
      summary: Submit a TEIF invoice (raw XML)
      security: [{ partnerOAuth: [invoice:write] }]
      requestBody:
        required: true
        content:
          application/xml:
            schema:
              type: string
      responses:
        "201":
          description: Invoice created. Same status behavior as JSON submit — see above.
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/InvoiceSummary"


  /api/core-proxy/teif/validate:
    post:
      summary: Validate a TEIF XML document against the XSD (no invoice created)
      description: >
        Corrected path — verified live. This is mounted at /api/core-proxy/teif/validate, NOT
        /api/core-proxy/invoices/teif/validate-xml-xsd (that 404s; it's Core's internal path,
        /api/v1/invoices/teif/validate-xml-xsd, mistakenly exposed at the same path in an earlier
        version of the partner-facing Postman collection — since fixed).
      security: [{ partnerOAuth: [invoice:read] }]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [xml]
              properties:
                xml: { type: string, description: "Field name is 'xml', not 'xmlContent' —
      verified live." }
      responses:
        "200":
```

```
          description: Validation result
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/TeifValidationResult"

  /api/core-proxy/invoices:
    get:
      summary: List invoices for the org-scoped token
      security: [{ partnerOAuth: [invoice:read] }]
      parameters:
        - name: direction
          in: query
          schema: { type: string, enum: [SENT, RECEIVED] }
        - name: status
          in: query
          schema: { type: string }
        - name: startDate
          in: query
          schema: { type: string, format: date }
        - name: endDate
          in: query
          schema: { type: string, format: date }
        - name: senderMatricule
          in: query
          schema: { type: string }
        - name: receiverMatricule
          in: query
          schema: { type: string }
        - name: page
          in: query
          schema: { type: integer, default: 0 }
        - name: pageSize
          in: query
          schema: { type: integer, default: 20, minimum: 1, maximum: 100 }
      responses:
        "200":
          description: Paginated invoice list
          content:
            application/json:
```

```
            schema:
              type: object
              properties:
                content:
                  type: array
                  items: { $ref: "#/components/schemas/InvoiceSummary" }
                page: { type: integer }
                pageSize: { type: integer }
                totalElements: { type: integer }
                totalPages: { type: integer }

/api/core-proxy/invoices/{id}:
  get:
    summary: Get full invoice detail
    description: >
      Response is WRAPPED — verified live — not a flat InvoiceSummary like list items.
      Read invoice fields from the `invoice` sub-object.
    security: [{ partnerOAuth: [invoice:read] }]
    parameters:
      - $ref: "#/components/parameters/InvoiceId"
    responses:
      "200":
        description: OK
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/InvoiceDetailResponse"
      "404":
        description: Not found

# Note: /api/core-proxy/invoices/{id}/status has been removed — it never had a real
# implementation (no matching backend mapping), so it always returned 500 INTERNAL_ERROR.
# Use GET /api/core-proxy/invoices/{id} and read `invoice.status` instead; the path now
# returns a clean 404.

/api/core-proxy/invoices/{id}/sign-and-send:
  post:
    summary: Sign an invoice (using the client's own configured method) and submit it to TTN
    security: [{ partnerOAuth: [invoice:write, seal:sign, digigo:sign] }]
    parameters:
```

```
        - $ref: "#/components/parameters/InvoiceId"
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required: [signatureType]
            properties:
              signatureType:
                type: string
                enum: [SEAL, DIGIGO]
    responses:
      "200":
        description: >
          Async job created. jobType is PROVIDER-SPECIFIC — verified live as
          SIGN_AND_SUBMIT_SEAL / SIGN_AND_SUBMIT_DIGIGO, not a generic "SIGN_AND_SEND".
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/JobCreated"
      "402":
        description: NO_SIGNATURE_CREDITS — client org's signing quota/packs exhausted
      "409":
        description: Invoice not in an allowed source status for sign-and-send

  /api/core-proxy/invoices/batch/sign-and-send:
    post:
      summary: Sign and send multiple invoices at once
      security: [{ partnerOAuth: [invoice:write, seal:sign, digigo:sign] }]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                invoiceIds:
                  type: array
                  items: { type: string }
```

```
              signatureType:
                type: string
                enum: [SEAL, DIGIGO]
    responses:
      "200":
        description: Batch job results

/api/core-proxy/jobs/{jobId}:
  get:
    summary: Poll an async job's status (e.g. sign-and-send)
    description: >
      Response is richer than the initial sign-and-send POST response — verified live, includes
      progress/currentStep/outputData with per-step failure detail.
    security: [{ partnerOAuth: [invoice:read] }]
    parameters:
      - name: jobId
        in: path
        required: true
        schema: { type: string }
    responses:
      "200":
        description: OK
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/JobStatus"

/api/core-proxy/invoices/{id}/pdf:
  get:
    summary: Download the invoice PDF
    description: >
      Binary response, works regardless of the invoice's status (drafts included).
    security: [{ partnerOAuth: [invoice:read] }]
    parameters:
      - $ref: "#/components/parameters/InvoiceId"
    responses:
      "200":
        description: application/pdf binary
        content:
          application/pdf:
```

```
            schema: { type: string, format: binary }
      "401": { $ref: "#/components/responses/Unauthorized" }


/api/core-proxy/invoices/{id}/generated-xml:
  get:
    summary: Get the generated TEIF XML
    description: >
      ⚠️ Past the SIGNED status, this regenerates a fresh UNSIGNED XML on the fly (drops the
      signature
      and the TTN QR code). For the actual final, signed/TTN-timestamped document, read the
      teifXml
      field from GET /api/core-proxy/invoices/{id} instead — see §7 in the guide body.
    security: [{ partnerOAuth: [invoice:read] }]
    parameters:
      - $ref: "#/components/parameters/InvoiceId"
    responses:
      "200":
        description: application/xml body
        content:
          application/xml:
            schema: { type: string }
      "401": { $ref: "#/components/responses/Unauthorized" }


/api/core-proxy/webhooks:
  post:
    summary: Create a webhook subscription
    description: >
      Requires your partner app to be authorized for the target organization (same authorization
      as
      invoices, resolved from the org-scoped access_token). The secret is returned only once, at
      creation time.
    security: [{ partnerOAuth: [invoice:write] }]
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required: [url]
            properties:
              url: { type: string, format: uri }
              events:
```

```
                type: array
                items: { type: string }
                description: Defaults to the 7 standard event types when omitted (see §9 in the
      guide body).
              customData: { type: object }
              ttlSeconds: { type: integer, description: "Defaults to 30 days" }
    responses:
      "201":
        description: Created
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/WebhookSubscription"
      "401": { $ref: "#/components/responses/Unauthorized" }
  get:
    summary: List webhook subscriptions created by your partner app for the target organization
    security: [{ partnerOAuth: [invoice:read] }]
    responses:
      "200":
        description: OK
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: "#/components/schemas/WebhookSubscription"

/api/core-proxy/webhooks/{webhookId}:
  delete:
    summary: Delete a webhook subscription
    security: [{ partnerOAuth: [invoice:write] }]
    parameters:
      - name: webhookId
        in: path
        required: true
        schema: { type: string }
    responses:
      "200":
        description: OK

/api/core-proxy/webhooks/{webhookId}/test:
```

```yaml
    post:
      summary: Send a test event (webhook.test) to one subscription
      security: [{ partnerOAuth: [invoice:write] }]
      parameters:
        - name: webhookId
          in: path
          required: true
          schema: { type: string }
      responses:
        "200":
          description: OK

components:
  parameters:
    InvoiceId:
      name: id
      in: path
      required: true
      schema:
        type: string

  responses:
    Unauthorized:
      description: Missing/invalid token, or insufficient scope
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/ErrorResponse"
    Forbidden:
      description: Not authorized for this organization or endpoint
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/ErrorResponse"

  securitySchemes:
    partnerOAuth:
      type: oauth2
      flows:
        clientCredentials:
```

```
        tokenUrl: /oauth/token
        scopes:
          invoice:write: Create, submit, sign, and delete invoices
          invoice:read: List and read invoices, jobs, statuses
          ttn:submit: Manually (re)submit to TTN (org-owned keys only)
          seal:sign: Trigger SEAL signing
          digigo:sign: Trigger DigiGO signing

schemas:
  TokenResponse:
    type: object
    properties:
      access_token: { type: string }
      token_type: { type: string, example: Bearer }
      expires_in: { type: integer, example: 3600 }
      scope: { type: string }

  OAuthError:
    type: object
    properties:
      error: { type: string }
      error_description: { type: string }

  ErrorResponse:
    type: object
    properties:
      error: { type: string }
      message: { type: string }
      code: { type: string }

  AuthorizedOrg:
    type: object
    properties:
      orgId: { type: string, format: uuid }
      orgName: { type: string }
      taxIdentifier: { type: string }
      allowedScopes:
        type: array
        items: { type: string }
      authorizedAt:
```

```
        type: string
        description: >
          Raw timestamp, NOT strict ISO-8601 — verified live as "2026-05-11 20:59:43.542231"
          (space-separated, microsecond precision, no T/Z). Do not validate against a strict
          ISO-8601 regex.
        example: "2026-05-11 20:59:43.542231"


    TeifInvoiceJson:
      type: object
      required: [version, controlingAgency, header, body]
      properties:
        version: { type: string, example: "1.8.8" }
        controlingAgency: { type: string, example: TTN }
        header: { type: object }
        body: { type: object }


    InvoiceSummary:
      type: object
      description: >
        Returned by list items AND by submit/submit-xml (full object, not a minimal envelope —
        verified live). Status starts at VALIDATED after a successful submit, not DRAFT.
      properties:
        id: { type: string, format: uuid }
        orgId: { type: string, format: uuid }
        invoiceNumber: { type: string }
        invoiceTypeCode: { type: string }
        invoiceDate: { type: string, format: date }
        status:
          type: string
          enum: [DRAFT, SUBMITTED, VALIDATED, VALIDATION_FAILED, SIGNED, SIGNATURE_FAILED,
        SENT_TTN, ACCEPTED_TTN, REJECTED_TTN, ERROR_TTN, TIMEOUT_TTN, CANCELLED, ARCHIVED]
        senderName: { type: string }
        receiverName: { type: string }
        receiverIdentifierType: { type: string }
        receiverIdentifierValue: { type: string }
        totalAmountExclTax: { type: number }
        totalTaxAmount: { type: number, nullable: true, description: "May be null immediately
        after a JSON submit — verified live" }
        totalAmountInclTax: { type: number, nullable: true }
        currencyCode: { type: string, example: TND }
        ttnGeneratedRef: { type: string, nullable: true }
```

```
        ttnIdSaveEfact: { type: integer, nullable: true }
        createdAt: { type: string, format: date-time }
        submittedAt: { type: string, format: date-time, nullable: true }
        signedAt: { type: string, format: date-time, nullable: true }
        createdBy: { type: string, description: "The clientId (ak_partner_...) that created it,
        for partner-submitted invoices" }
        isQuotation: { type: boolean }
        quotationStatus: { type: string, nullable: true }
        duplicateCopy: { type: boolean }
        importSource: { type: string, nullable: true }
        importBlockedResign: { type: boolean }
        direction: { type: string, enum: [SENT, RECEIVED] }
        partnerAppId: { type: string, nullable: true }

    InvoiceDetailResponse:
      type: object
      description: The wrapped shape returned by GET /api/core-proxy/invoices/{id} — verified
        live.
      properties:
        invoice: { $ref: "#/components/schemas/InvoiceSummary" }
        teifJson: { type: string, description: "Generated TEIF document as a JSON string" }
        teifXml: { type: string, description: "Generated TEIF document as raw XML" }
        ttnQrCodeBase64: { type: string, nullable: true }
        localSignedXml: { type: string, nullable: true }
        validationErrors: { type: array, items: { type: object }, nullable: true }
        ttnAcknowledgments: { type: array, items: { type: object }, nullable: true }

    TeifValidationResult:
      type: object
      properties:
        valid: { type: boolean }
        issues:
          type: array
          items:
            type: object
            properties:
              type: { type: string, example: XML_SCHEMA_VIOLATION }
              message: { type: string }
              severity: { type: string, example: ERROR }
              xpath: { type: string, example: "line:1" }
```

```
    WebhookSubscription:
      type: object
      description: >
        Response shape for webhook endpoints. `secret` is only present in the POST /webhooks
        creation response (returned once, never again on subsequent GETs).
      properties:
        id: { type: string }
        appId: { type: string, description: "Your partner app's client_id" }
        url: { type: string, format: uri }
        events:
          type: array
          items: { type: string }
        customData: { type: object }
        status: { type: string, example: ACTIVE }
        secret: { type: string, description: "Only present once, on the creation response" }
        expiresAt: { type: string }
        createdAt: { type: string }
        updatedAt: { type: string }

    JobCreated:
      type: object
      description: Initial response from POST sign-and-send.
      properties:
        jobId: { type: string, format: uuid }
        invoiceId: { type: string, format: uuid }
        jobType:
          type: string
          enum: [SIGN_AND_SUBMIT_SEAL, SIGN_AND_SUBMIT_DIGIGO]
          description: Provider-specific — verified live. Not a generic "SIGN_AND_SEND".
        status: { type: string, example: PENDING }
        success: { type: boolean }
        message: { type: string, example: "Invoice queued for signature and TTN submission" }

    JobStatus:
      type: object
      description: Richer shape returned by GET /jobs/{jobId} — verified live.
      properties:
        jobId: { type: string, format: uuid }
        invoiceId: { type: string, format: uuid }
        jobType: { type: string, enum: [SIGN_AND_SUBMIT_SEAL, SIGN_AND_SUBMIT_DIGIGO] }
```

```
        status: { type: string, enum: [PENDING, COMPLETED, FAILED] }
        progress: { type: integer, description: "0-100" }
        currentStep: { type: string, example: CERT_VERIFICATION }
        errorMessage: { type: string, nullable: true }
        errorData: { type: object, nullable: true }
        outputData:
          type: object
          nullable: true
          description: "Per-step results/failures, e.g. { step0CertVerification: { valid, success,
      errorCode, error } }"
        signatureResult: { type: object, nullable: true }
        ttnSubmissionResult: { type: object, nullable: true }
        ttnFinalResult: { type: object, nullable: true }
        createdAt: { type: string, format: date-time }
        startedAt: { type: string, format: date-time, nullable: true }
        completedAt: { type: string, format: date-time, nullable: true }
        retryCount: { type: integer }
```

Postman collection

The Fatoora dashboard (**Partner → Credentials → API guide → Download Postman collection**) generates a ready-to-import Postman v2.1 collection via `buildPostmanCollection()` in `fatooraLive/src/components/partner/credentials/PartnerApiGuide.tsx`. It ships pre-wired test scripts that auto-populate the `access_token` / `app_only_token` / `org_id` collection variables as you step through requests, and covers every endpoint in this guide: authentication (app-only + org-scoped), organization discovery, JSON/XML invoice submission, invoice listing/detail/status, XSD validation, and sign-and-send.

The JSON below is the **exact, unmodified output of that function** (extracted and executed directly from the source file, not manually transcribed), generated with the production `base_url` and placeholder credentials — replace `client_id` / `client_secret` / `org_id` with your own values, or just download your pre-filled copy from the dashboard.

Every request carries one or more **saved example responses** (success and, where instructive, the most common error — e.g. `invalid_client`, `NO_SIGNATURE_CREDITS`, a validation failure) directly in the collection — Postman shows these under each request's "Examples" tab so partners can see a realistic response body and status code before making a single call.

Also saved as a standalone file: `partner-api-postman-collection.json`.

```
{
  "info": {
    "name": "Fatoora Partner API",
    "description": "Collection pour l'intégration partenaire Fatoora — soumettre des factures TEIF
        au nom d'une organisation cliente autorisée.",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
  },
  "variable": [
    {
      "key": "base_url",
      "value": "https://business.fatoora.tn",
      "type": "string"
    },
    {
      "key": "client_id",
      "value": "ak_partner_sample00000000",
      "type": "string"
    },
    {
      "key": "client_secret",
      "value": "YOUR_SECRET_ID_HERE",
      "type": "string"
    },
    {
      "key": "org_id",
      "value": "TARGET_CLIENT_ORG_UUID",
      "type": "string"
    },
    {
      "key": "access_token",
      "value": "",
      "type": "string"
    },
    {
      "key": "app_only_token",
      "value": "",
      "type": "string"
    },
    {
      "key": "invoice_id",
```

```
      "value": "",
      "type": "string"
    },
    {
      "key": "job_id",
      "value": "",
      "type": "string"
    },
    {
      "key": "webhook_id",
      "value": "",
      "type": "string"
    },
    {
      "key": "webhook_secret",
      "value": "",
      "type": "string"
    },
    {
      "key": "tax_id",
      "value": "1234567ABC000",
      "type": "string"
    },
    {
      "key": "seller_tax_id",
      "value": "1234567ABC000",
      "type": "string"
    },
    {
      "key": "buyer_tax_id",
      "value": "1234567RAM000",
      "type": "string"
    },
    {
      "key": "seller_name",
      "value": "Ma Société SARL",
      "type": "string"
    },
    {
      "key": "buyer_name",
```

```
      "value": "Mon Client SARL",
      "type": "string"
    },
    {
      "key": "seller_address_desc",
      "value": "Avenue Habib, Tunis",
      "type": "string"
    },
    {
      "key": "seller_street",
      "value": "Avenue Habib",
      "type": "string"
    },
    {
      "key": "seller_city",
      "value": "Tunis",
      "type": "string"
    },
    {
      "key": "seller_postal_code",
      "value": "3455",
      "type": "string"
    },
    {
      "key": "buyer_address_desc",
      "value": "RUE Habib, Tunis",
      "type": "string"
    },
    {
      "key": "buyer_street",
      "value": "RUE Habib",
      "type": "string"
    },
    {
      "key": "buyer_city",
      "value": "Tunis",
      "type": "string"
    },
    {
      "key": "buyer_postal_code",
```

```
      "value": "1000",
      "type": "string"
    },
    {
      "key": "seller_phone",
      "value": "23445656",
      "type": "string"
    },
    {
      "key": "seller_email",
      "value": "contact@masociete.tn",
      "type": "string"
    },
    {
      "key": "seller_contact_id",
      "value": "C001",
      "type": "string"
    },
    {
      "key": "seller_contact_name",
      "value": "Service Facturation",
      "type": "string"
    },
    {
      "key": "invoice_number",
      "value": "FACT-2026-000001",
      "type": "string"
    },
    {
      "key": "invoice_date_ddMMyy",
      "value": "150726",
      "type": "string"
    },
    {
      "key": "line1_item_id",
      "value": "1",
      "type": "string"
    },
    {
      "key": "line1_item_code",
```

```
      "value": "ART-001",
      "type": "string"
    },
    {
      "key": "line1_description",
      "value": "Mon article client",
      "type": "string"
    },
    {
      "key": "line1_qty",
      "value": "2",
      "type": "string"
    },
    {
      "key": "line1_unit",
      "value": "PCE",
      "type": "string"
    },
    {
      "key": "line1_ht",
      "value": "1000.000",
      "type": "string"
    },
    {
      "key": "line1_ttc",
      "value": "1190.000",
      "type": "string"
    },
    {
      "key": "total_ht",
      "value": "1000.000",
      "type": "string"
    },
    {
      "key": "total_tva_pct",
      "value": "19",
      "type": "string"
    },
    {
      "key": "total_tva",
```

```
      "value": "190.000",
      "type": "string"
    },
    {
      "key": "total_timbre",
      "value": "5.000",
      "type": "string"
    },
    {
      "key": "total_taxes",
      "value": "195.000",
      "type": "string"
    },
    {
      "key": "total_ttc",
      "value": "1195.000",
      "type": "string"
    },
    {
      "key": "partner_app_id",
      "value": "partn_sample000000000000",
      "type": "string"
    },
    {
      "key": "payment_condition_code",
      "value": "I-122",
      "type": "string"
    }
  ],
  "item": [
    {
      "name": "🔐 Authentification",
      "item": [
        {
          "name": "1 — Token app-only (découverte orgs, sans org_id)",
          "event": [
            {
              "listen": "test",
              "script": {
                "type": "text/javascript",
```

```
            "exec": [
              "const json = pm.response.json();",
              "if (json.access_token) {",
              "  pm.collectionVariables.set('app_only_token', json.access_token);",
              "  console.log('✅ app_only_token sauvegardé. Expire dans ' + json.expires_in +
's');",
              "} else {",
              "  console.error('❌ Pas de access_token:', JSON.stringify(json));",
              "}"
            ]
          }
        }
      ],
      "request": {
        "method": "POST",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/x-www-form-urlencoded"
          }
        ],
        "body": {
          "mode": "urlencoded",
          "urlencoded": [
            {
              "key": "grant_type",
              "value": "client_credentials",
              "description": "Doit être 'client_credentials'"
            },
            {
              "key": "client_id",
              "value": "{{client_id}}",
              "description": "Votre App ID (page identifiants)"
            },
            {
              "key": "client_secret",
              "value": "{{client_secret}}",
              "description": "Votre Secret ID (page identifiants)"
            }
          ]
        },
```

```
      "url": {
        "raw": "{{base_url}}/oauth/token",
        "host": [
          "{{base_url}}"
        ],
        "path": [
          "oauth",
          "token"
        ]
      },
      "description": "Obtenir un token **app-only** (sans org_id) pour les API de découverte des organisations.\n\nCe token est lié à votre application partenaire uniquement — il n'agit pas au nom d'une organisation cliente.\n\n**Utilisez-le pour** :\n- `GET /api/partner/authorized-orgs`\n- `GET /api/partner/resolve-org?taxId=XXX`\n\nLe script de test sauvegarde automatiquement le token dans la variable `app_only_token`."
    },
    "response": [
      {
        "name": "200 OK — app-only token",
        "originalRequest": {
          "method": "POST",
          "header": [],
          "url": {
            "raw": "{{base_url}}/oauth/token",
            "host": [
              "{{base_url}}/oauth/token"
            ]
          }
        },
        "status": "OK",
        "code": 200,
        "_postman_previewlanguage": "json",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/json"
          }
        ],
        "cookie": [],
        "body": "{\n  \"access_token\": \"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6InBhcnRuX3h4eHgifQ.SIGNATURE\",\n  \"token_type\": \"Bearer\",\n  \"expires_in\": 3600,\n  \"scope\": \"invoice:write invoice:read ttn:submit\"\n}"
      },
```

```
      {
        "name": "401 Unauthorized — invalid_client",
        "originalRequest": {
          "method": "POST",
          "header": [],
          "url": {
            "raw": "{{base_url}}/oauth/token",
            "host": [
              "{{base_url}}/oauth/token"
            ]
          }
        },
        "status": "Unauthorized",
        "code": 401,
        "_postman_previewlanguage": "json",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/json"
          }
        ],
        "cookie": [],
        "body": "{\n  \"error\": \"invalid_client\",\n  \"error_description\": \"Invalid
client credentials.\"\n}"
      }
    ]
  },
  {
    "name": "2 — Token org-scoped (factures, signature)",
    "event": [
      {
        "listen": "test",
        "script": {
          "type": "text/javascript",
          "exec": [
            "const json = pm.response.json();",
            "if (json.access_token) {",
            "  pm.collectionVariables.set('access_token', json.access_token);",
            "  console.log('✅ access_token sauvegardé. Expire dans ' + json.expires_in +
's');",
            "  console.log('Scopes effectifs:', json.scope);",
```

```
              "} else {",
              "  console.error('❌ Pas de access_token:', JSON.stringify(json));",
              "}"
            ]
          }
        }
      ],
      "request": {
        "method": "POST",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/x-www-form-urlencoded"
          }
        ],
        "body": {
          "mode": "urlencoded",
          "urlencoded": [
            {
              "key": "grant_type",
              "value": "client_credentials",
              "description": "Doit être 'client_credentials'"
            },
            {
              "key": "client_id",
              "value": "{{client_id}}",
              "description": "Votre App ID (page identifiants)"
            },
            {
              "key": "client_secret",
              "value": "{{client_secret}}",
              "description": "Votre Secret ID (page identifiants)"
            },
            {
              "key": "org_id",
              "value": "{{org_id}}",
              "description": "UUID de l'organisation cliente cible — obtenu via l'étape
découverte"
            }
          ]
        },
```

```
      "url": {
        "raw": "{{base_url}}/oauth/token",
        "host": [
          "{{base_url}}"
        ],
        "path": [
          "oauth",
          "token"
        ]
      },
      "description": "Obtenir un token **org-scoped** pour agir au nom d'une organisation cliente spécifique.\n\n**org_id** : UUID de l'organisation cliente — récupéré via `GET /api/partner/resolve-org` ou `GET /api/partner/authorized-orgs`.\n\n**Utilisez ce token pour** :\n- `POST /api/core-proxy/invoices/submit`\n- `POST /api/core-proxy/invoices/submit-xml`\n- `POST /api/core-proxy/invoices/:id/sign-and-send`\n\nLe script de test sauvegarde automatiquement le token dans la variable `access_token`."
    },
    "response": [
      {
        "name": "200 OK — org-scoped token",
        "originalRequest": {
          "method": "POST",
          "header": [],
          "url": {
            "raw": "{{base_url}}/oauth/token",
            "host": [
              "{{base_url}}/oauth/token"
            ]
          }
        },
        "status": "OK",
        "code": 200,
        "_postman_previewlanguage": "json",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/json"
          }
        ],
        "cookie": [],
        "body": "{\n  \"access_token\": \"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJwYXJ0bl94eHh4Iiwib3JnSWQiOiJvcmdfeHh4eCJ9.SIGNATURE\",\n  \"token_type\": \"Bearer\",\n  \"expires_in\": 3600,\n  \"scope\": \"invoice:write invoice:read ttn:submit seal:sign\"\n}"
```

```
            },
            {
              "name": "403 Forbidden — org not authorized",
              "originalRequest": {
                "method": "POST",
                "header": [],
                "url": {
                  "raw": "{{base_url}}/oauth/token",
                  "host": [
                    "{{base_url}}/oauth/token"
                  ]
                }
              },
              "status": "Forbidden",
              "code": 403,
              "_postman_previewlanguage": "json",
              "header": [
                {
                  "key": "Content-Type",
                  "value": "application/json"
                }
              ],
              "cookie": [],
              "body": "{\n  \"error\": \"access_denied\",\n  \"error_description\": \"Organization
      has not authorized this partner application.\"\n}"
            }
          ]
        }
      ]
    },
    {
      "name": "🏢 Découverte des organisations",
      "item": [
        {
          "name": "Lister les organisations autorisées",
          "event": [
            {
              "listen": "test",
              "script": {
                "type": "text/javascript",
                "exec": [
```

```
            "const json = pm.response.json();",
            "const orgs = json.authorizedOrgs || [];",
            "if (orgs.length > 0) {",
            "  pm.collectionVariables.set('org_id', orgs[0].orgId);",
            "  pm.collectionVariables.set('seller_tax_id', orgs[0].taxIdentifier);",
            "  console.log('✅ org_id sauvegardé (1ère org) :', orgs[0].orgId, '—',
orgs[0].orgName);",
            "  console.log('✅ seller_tax_id sauvegardé :', orgs[0].taxIdentifier);",
            "  console.log('   Matricule :', orgs[0].taxIdentifier);",
            "  console.log('   Scopes :', (orgs[0].allowedScopes || []).join(', '));",
            "  if (orgs.length > 1) {",
            "    console.log('ℹ️ ' + (orgs.length - 1) + ' autre(s) org(s) disponible(s).
Modifiez {{org_id}} manuellement si besoin.');",
            "  }",
            "} else {",
            "  console.warn('⚠️ Aucune organisation autorisée trouvée.');",
            "}"
          ]
        }
      }
    ],
    "request": {
      "method": "GET",
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer {{app_only_token}}"
        }
      ],
      "url": {
        "raw": "{{base_url}}/api/partner/authorized-orgs",
        "host": [
          "{{base_url}}"
        ],
        "path": [
          "api",
          "partner",
          "authorized-orgs"
        ]
      },
```

```
    "description": "Récupérer toutes les organisations qui ont autorisé cette application partenaire.\n\n**Token requis** : `app_only_token` (obtenu via l'authentification #1 sans org_id).\n\n**Script de test automatique** : sauvegarde l'`org_id` de la première organisation dans `{{org_id}}` — utilisé par l'authentification #2 pour obtenir un token org-scoped."
  },
  "response": [
    {
      "name": "200 OK",
      "originalRequest": {
        "method": "GET",
        "header": [],
        "url": {
          "raw": "{{base_url}}/api/partner/authorized-orgs",
          "host": [
            "{{base_url}}/api/partner/authorized-orgs"
          ]
        }
      },
      "status": "OK",
      "code": 200,
      "_postman_previewlanguage": "json",
      "header": [
        {
          "key": "Content-Type",
          "value": "application/json"
        }
      ],
      "cookie": [],
      "body": "{\n  \"authorizedOrgs\": [\n    {\n      \"orgId\": \"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\",\n      \"orgName\": \"Société ABC SARL\",\n      \"taxIdentifier\": \"1234567ABCD000\",\n      \"allowedScopes\": [\n        \"invoice:write\",\n        \"invoice:read\",\n        \"seal:sign\",\n        \"ttn:submit\"\n      ],\n      \"authorizedAt\": \"2026-05-01 10:00:00.000000\"\n    }\n  ],\n  \"total\": 1\n}"
    }
  ]
},
{
  "name": "Résoudre un matricule fiscal → org_id",
  "event": [
    {
      "listen": "test",
      "script": {
        "type": "text/javascript",
```

```
            "exec": [
              "const json = pm.response.json();",
              "if (json.orgId) {",
              "  pm.collectionVariables.set('org_id', json.orgId);",
              "  console.log('✅ org_id sauvegardé :', json.orgId, '—', json.orgName);",
              "  console.log('   Matricule :', json.taxIdentifier);",
              "  console.log('   Scopes :', (json.allowedScopes || []).join(', '));",
              "} else if (json.error === 'ORG_NOT_FOUND') {",
              "  console.error('❌ Matricule non trouvé. Vérifiez {{tax_id}} et que l\\'organisation a bien autorisé votre app.');",
              "} else {",
              "  console.error('❌ Erreur :', JSON.stringify(json));",
              "}"
            ]
          }
        }
      ],
      "request": {
        "method": "GET",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer {{app_only_token}}"
          }
        ],
        "url": {
          "raw": "{{base_url}}/api/partner/resolve-org?taxId={{seller_tax_id}}",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "partner",
            "resolve-org"
          ],
          "query": [
            {
              "key": "taxId",
              "value": "{{seller_tax_id}}",
              "description": "Matricule fiscal (8 ou 13 caractères)"
            }
```

```
            ]
          },
          "description": "Résoudre un matricule fiscal tunisien (8 ou 13 caractères) en org_id Fatoora.\n\n**Token requis** : `app_only_token` (obtenu via l'authentification #1 sans org_id).\n\n**Formats acceptés** :\n- 8 car. : `NNNNNNNL` (ancien format)\n- 13 car. : `NNNNNNNLCCCNNN` (nouveau format)\n\n**Script de test automatique** : sauvegarde l'`org_id` résolu dans `{{org_id}}` — utilisé ensuite par l'authentification #2."
        },
        "response": [
          {
            "name": "200 OK — resolved",
            "originalRequest": {
              "method": "GET",
              "header": [],
              "url": {
                "raw": "{{base_url}}/api/partner/resolve-org?taxId={{seller_tax_id}}",
                "host": [
                  "{{base_url}}/api/partner/resolve-org?taxId={{seller_tax_id}}"
                ]
              }
            },
            "status": "OK",
            "code": 200,
            "_postman_previewlanguage": "json",
            "header": [
              {
                "key": "Content-Type",
                "value": "application/json"
              }
            ],
            "cookie": [],
            "body": "{\n  \"orgId\": \"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\",\n  \"orgName\": \"Société ABC SARL\",\n  \"taxIdentifier\": \"1234567ABCD000\",\n  \"allowedScopes\": [\n    \"invoice:write\",\n    \"seal:sign\"\n  ],\n  \"authorizedAt\": \"2026-05-01 10:00:00.000000\"\n}"
          },
          {
            "name": "404 Not Found — ORG_NOT_FOUND",
            "originalRequest": {
              "method": "GET",
              "header": [],
              "url": {
                "raw": "{{base_url}}/api/partner/resolve-org?taxId={{tax_id}}",
                "host": [
```

```
                "{{base_url}}/api/partner/resolve-org?taxId={{tax_id}}"
              ]
            }
          },
          "status": "Not Found",
          "code": 404,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "{\n  \"error\": \"ORG_NOT_FOUND\",\n  \"message\": \"Aucune organisation autorisée trouvée pour le matricule \\\"{{tax_id}}\\\". Assurez-vous que l'organisation a bien activé votre application partenaire.\"\n}"
        }
      ]
    }
  ]
},
{
  "name": "📄 Factures — JSON TEIF",
  "item": [
    {
      "name": "Soumettre une facture (JSON TEIF)",
      "event": [
        {
          "listen": "test",
          "script": {
            "type": "text/javascript",
            "exec": [
              "if (pm.response.code === 201 || pm.response.code === 200) {",
              "  const json = pm.response.json();",
              "  if (json.id) {",
              "    pm.collectionVariables.set('invoice_id', json.id);",
              "    console.log('✅ invoice_id sauvegardé :', json.id);",
              "  }",
              "} else {",
              "  console.error('❌ Facture non créée, invoice_id inchangé:', pm.response.text());",
```

```
            "}"
          ]
        }
      }
    ],
    "request": {
      "method": "POST",
      "header": [
        {
          "key": "Content-Type",
          "value": "application/json"
        },
        {
          "key": "Authorization",
          "value": "Bearer {{access_token}}"
        },
        {
          "key": "X-Allow-Partner-Upsert",
          "value": "true"
        }
      ],
      "body": {
        "mode": "raw",
```

```
        "raw": "{\n  \"version\": \"1.8.8\",\n  \"controlingAgency\": \"TTN\",\n
\"header\": {\n    \"MessageSenderIdentifier\": {\n      \"@type\": \"I-01\",\n
\"#text\": \"{{seller_tax_id}}\"\n    },\n    \"MessageRecieverIdentifier\": {\n
\"@type\": \"I-01\",\n      \"#text\": \"{{buyer_tax_id}}\"\n    }\n  },\n  \"body\": {\n
\"bgm\": {\n      \"documentIdentifier\": \"{{invoice_number}}\",\n
\"documentTypeCode\": \"I-11\",\n      \"documentType\": \"Facture\"\n    },\n    \"dtm\":
{\n      \"dateText\": [\n        {\n          \"functionCode\": \"I-31\",\n
\"format\": \"ddMMyy\",\n          \"value\": \"{{invoice_date_ddMMyy}}\"\n        }\n
]\n    },\n    \"partnerSection\": {\n      \"partnerDetails\": [\n        {\n
\"functionCode\": \"I-62\",\n          \"nad\": {\n            \"partnerIdentifier\": {\n
\"type\": \"I-01\",\n              \"value\": \"{{seller_tax_id}}\"\n            },\n
\"partnerName\": {\n              \"nameType\": \"Qualification\",\n
\"value\": \"{{seller_name}}\"\n            },\n            \"partnerAdresses\": [\n
{\n                \"adressDescription\": \"Siège social\",\n                \"street\":
\"{{seller_street}}\",\n                \"cityName\": \"{{seller_city}}\",\n
\"postalCode\": \"{{seller_postal_code}}\",\n                \"country\": {\n
\"codeList\": \"ISO_3166-1\",\n                \"value\": \"TN\"\n              }\n
}\n            ]\n          },\n          \"ctaSection\": [],\n          \"rffSection\":
[],\n          \"loc\": []\n        },\n        {\n          \"functionCode\": \"I-61\",\n
\"nad\": {\n            \"partnerIdentifier\": {\n              \"type\": \"I-01\",\n
\"value\": \"{{buyer_tax_id}}\"\n            },\n            \"partnerName\": {\n
\"nameType\": \"Qualification\",\n              \"value\": \"{{buyer_name}}\"\n
},\n            \"partnerAdresses\": [\n              {\n
\"adressDescription\": \"Siège social\",\n                \"street\": \"
{{buyer_street}}\",\n                \"cityName\": \"{{buyer_city}}\",\n
\"postalCode\": \"{{buyer_postal_code}}\",\n                \"country\": {\n
\"codeList\": \"ISO_3166-1\",\n                \"value\": \"TN\"\n              }\n
}\n            ]\n          },\n          \"ctaSection\": [],\n          \"rffSection\":
[],\n          \"loc\": []\n        }\n      ]\n    },\n    \"linSection\": {\n
\"lin\": [\n        {\n          \"itemIdentifier\": \"1\",\n          \"linImd\": {\n
\"lang\": \"fr\",\n            \"itemCode\": \"{{line1_item_code}}\",\n
\"itemDescription\": \"{{line1_description}}\"\n          },\n          \"linQty\": {\n
\"quantity\": {\n              \"measurementUnit\": \"C62\",\n              \"value\": \"
{{line1_qty}}\"\n            }\n          },\n          \"linTax\": {\n
\"taxTypeName\": {\n              \"code\": \"I-66\",\n              \"value\": \"TVA\"\n
},\n            \"taxDetails\": {\n              \"taxRate\": \"19\",\n
\"taxRateBasis\": \"Percentage\"\n            }\n          },\n          \"linMoa\": {\n
\"moaDetails\": [\n            {\n              \"moa\": {\n
\"amountTypeCode\": \"I-188\",\n                \"currencyCodeList\": \"ISO_4217\",\n
\"amount\": {\n                  \"currencyIdentifier\": \"TND\",\n
\"value\": \"{{line1_ht}}\"\n                }\n              }\n            }\n
]\n          },\n          \"subLin\": []\n        }\n      ]\n    },\n    \"invoiceTax\":
{\n      \"invoiceTaxDetails\": [\n        {\n          \"tax\": {\n
\"taxTypeName\": {\n              \"code\": \"I-66\",\n              \"value\": \"TVA\"\n
},\n            \"taxCategory\": \"I-1602\",\n            \"taxDetails\": {\n
\"taxRate\": \"19.000\",\n              \"taxRateBasis\": \"Percentage\"\n            }\n
},\n          \"amountDetails\": [\n            {\n              \"moa\": {\n
\"amountTypeCode\": \"I-177\",\n                \"currencyCodeList\": \"ISO_4217\",\n
\"amount\": {\n                  \"currencyIdentifier\": \"TND\",\n
\"value\": \"{{total_ht}}\"\n                }\n              }\n            },\n
{\n              \"moa\": {\n                \"amountTypeCode\": \"I-178\",\n
\"currencyCodeList\": \"ISO_4217\",\n                \"amount\": {\n
\"currencyIdentifier\": \"TND\",\n                  \"value\": \"{{total_tva}}\"\n
}\n              }\n            }\n          ]\n        }\n      ]\n    },\n
\"invoiceMoa\": {\n      \"amountDetails\": [\n        {\n          \"moa\": {\n
\"amountTypeCode\": \"I-172\",\n            \"currencyCodeList\": \"ISO_4217\",\n
\"amount\": {\n              \"currencyIdentifier\": \"TND\",\n              \"value\": \"
{{total_ht}}\"\n            }\n          }\n        },\n        {\n          \"moa\": {\n
\"amountTypeCode\": \"I-181\",\n            \"currencyCodeList\": \"ISO_4217\",\n
\"amount\": {\n              \"currencyIdentifier\": \"TND\",\n              \"value\": \"
{{total_taxes}}\"\n            }\n          }\n        },\n        {\n          \"moa\":
{\n            \"amountTypeCode\": \"I-180\",\n            \"currencyCodeList\":
\"ISO_4217\",\n            \"amount\": {\n              \"currencyIdentifier\": \"TND\",\n
\"value\": \"{{total_ttc}}\"\n            }\n          }\n        },\n        {\n
\"moa\": {\n            \"amountTypeCode\": \"I-177\",\n            \"currencyCodeList\":
\"ISO_4217\",\n            \"amount\": {\n              \"currencyIdentifier\": \"TND\",\n
\"value\": \"{{total_ht}}\"\n            }\n          }\n        },\n        {\n
\"moa\": {\n            \"amountTypeCode\": \"I-176\",\n            \"currencyCodeList\":
\"ISO_4217\",\n            \"amount\": {\n              \"currencyIdentifier\": \"TND\",\n
\"value\": \"{{total_ht}}\"\n            }\n          }\n        }\n      ]\n    },\n
\"pyt\": {\n      \"pytSectionDetails\": [\n        {\n          \"pytPai\": {\n
\"paiConditionCode\": \"{{payment_condition_code}}\",\n            \"paiMeansCode\":
\"42\"\n          }\n        }\n      ]\n    }\n  },\n  \"additionnalDocuments\": null,\n
\"refTtnVal\": null,\n  \"signatures\": []\n}",
```

```
            "options": {
              "raw": {
                "language": "json"
              }
            }
          },
          "url": {
            "raw": "{{base_url}}/api/core-proxy/invoices/submit",
            "host": [
              "{{base_url}}"
            ],
            "path": [
              "api",
              "core-proxy",
              "invoices",
              "submit"
            ]
          },
          "description": "Soumettre une facture TEIF au nom de l'organisation cliente spécifiée
dans le token.\n\n**Scope requis** : `invoice:write`\n\n**Types de document
(documentTypeCode)** :\n- `I-11` → Facture standard\n- `I-12` → Note de débit\n- `I-13` →
Avoir (crédit)"
        },
        "response": [
          {
            "name": "201 Created",
            "originalRequest": {
              "method": "POST",
              "header": [],
              "url": {
                "raw": "{{base_url}}/api/core-proxy/invoices/submit",
                "host": [
                  "{{base_url}}/api/core-proxy/invoices/submit"
                ]
              }
            },
            "status": "Created",
            "code": 201,
            "_postman_previewlanguage": "json",
            "header": [
              {
                "key": "Content-Type",
```

```
            "value": "application/json"
          }
        ],
        "cookie": [],
        "body": "{\n  \"id\": \"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\",\n  \"orgId\": \"
{{org_id}}\",\n  \"invoiceNumber\": \"FACT-2026-000001\",\n  \"invoiceTypeCode\": \"I-
11\",\n  \"invoiceDate\": \"2026-07-12\",\n  \"status\": \"VALIDATED\",\n  \"senderName\":
\"Ma Société SARL\",\n  \"receiverName\": \"Mon Client SARL\",\n
\"receiverIdentifierType\": \"I-01\",\n  \"receiverIdentifierValue\": \"1234567RAM000\",\n
\"totalAmountExclTax\": 1000,\n  \"totalTaxAmount\": null,\n  \"totalAmountInclTax\":
null,\n  \"currencyCode\": \"TND\",\n  \"ttnGeneratedRef\": null,\n  \"ttnIdSaveEfact\":
null,\n  \"createdAt\": \"2026-07-11T20:04:58.237547\",\n  \"submittedAt\": \"2026-07-
11T20:04:58.223037\",\n  \"signedAt\": null,\n  \"createdBy\": \"{{client_id}}\",\n
\"isQuotation\": false,\n  \"quotationStatus\": null,\n  \"duplicateCopy\": false,\n
\"importSource\": null,\n  \"importBlockedResign\": false,\n  \"direction\": \"SENT\",\n
\"partnerAppId\": \"{{partner_app_id}}\"\n}"
    },
    {
      "name": "400 Bad Request — validation error",
      "originalRequest": {
        "method": "POST",
        "header": [],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices/submit",
          "host": [
            "{{base_url}}/api/core-proxy/invoices/submit"
          ]
        }
      },
      "status": "Bad Request",
      "code": 400,
      "_postman_previewlanguage": "json",
      "header": [
        {
          "key": "Content-Type",
          "value": "application/json"
        }
      ],
      "cookie": [],
      "body": "{\n  \"code\": \"VALIDATION_ERROR\",\n  \"error\": \"VALIDATION_ERROR\",\n
\"message\": \"TEIF payload failed schema validation.\",\n  \"details\": [\n    {\n
\"field\": \"body.partnerSection.partnerDetails[1].nad.partnerAdresses\",\n
\"reason\": \"Address is required to auto-create a new partner (X-Allow-Partner-
Upsert).\"\n    }\n  ],\n  \"path\": \"/api/v1/invoices/submit\",\n  \"timestamp\":
\"2026-07-11T00:00:00Z\",\n  \"status\": 400\n}"
    }
  ]
},
```

```
{
  "name": "Lister les factures du partenaire",
  "request": {
    "method": "GET",
    "header": [
      {
        "key": "Authorization",
        "value": "Bearer {{access_token}}"
      }
    ],
    "url": {
      "raw": "{{base_url}}/api/partner/invoices",
      "host": [
        "{{base_url}}"
      ],
      "path": [
        "api",
        "partner",
        "invoices"
      ]
    },
    "description": "Récupérer toutes les factures soumises par votre app sur toutes les organisations autorisées.\n\n**Scope requis** : `invoice:read`"
  },
  "response": [
    {
      "name": "200 OK",
      "originalRequest": {
        "method": "GET",
        "header": [],
        "url": {
          "raw": "{{base_url}}/api/partner/invoices",
          "host": [
            "{{base_url}}/api/partner/invoices"
          ]
        }
      },
      "status": "OK",
      "code": 200,
      "_postman_previewlanguage": "json",
      "header": [
```

```
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "{\n  \"invoices\": [\n    {\n      \"id\": \"inv_xxxxxxxxxxxxxxxx\",\n
      \"teifId\": \"FACT-2026-000001\",\n      \"invoiceNumber\": \"FACT-2026-000001\",\n
      \"invoiceDate\": \"2026-07-11\",\n      \"organizationId\": \"org_xxxx\",\n
      \"organizationName\": \"Mon Client SARL\",\n      \"status\": \"ACCEPTED_TTN\",\n
      \"totalAmount\": 1195,\n      \"totalAmountExclTax\": 1000,\n      \"totalTaxAmount\":
      195,\n      \"currency\": \"TND\",\n      \"ttnGeneratedRef\": \"TTN-REF-000123\",\n
      \"submittedAt\": \"2026-07-11T00:00:00Z\",\n      \"signedAt\": \"2026-07-
      11T00:05:00Z\",\n      \"createdAt\": \"2026-07-11T00:00:00Z\"\n    }\n  ]\n}"
        }
      ]
    }
  ]
},
{
  "name": "📋 Factures — Consultation",
  "item": [
    {
      "name": "Lister les factures (paginé)",
      "request": {
        "method": "GET",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer {{access_token}}"
          }
        ],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices?page=0&pageSize=20&direction=SENT",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "core-proxy",
            "invoices"
          ],
          "query": [
```

```
            {
              "key": "page",
              "value": "0",
              "description": "Numéro de page (commence à 0)"
            },
            {
              "key": "pageSize",
              "value": "20",
              "description": "Nombre de résultats par page (max 100)"
            },
            {
              "key": "direction",
              "value": "SENT",
              "description": "SENT = factures émises, RECEIVED = reçues"
            },
            {
              "key": "status",
              "value": "",
              "description": "Filtrer par statut : DRAFT, SIGNED, ACCEPTED_TTN, REJECTED —
laisser vide pour tous",
              "disabled": true
            },
            {
              "key": "invoiceNumber",
              "value": "",
              "description": "Recherche par numéro de facture (partiel)",
              "disabled": true
            },
            {
              "key": "startDate",
              "value": "",
              "description": "Date de début ISO 8601 (ex: 2026-01-01)",
              "disabled": true
            },
            {
              "key": "endDate",
              "value": "",
              "description": "Date de fin ISO 8601 (ex: 2026-12-31)",
              "disabled": true
            }
          ]
```

```
    },
    "description": "Lister les factures de l'organisation cliente liée au token (org-
scoped), avec pagination.\n\n**Scope requis** : `invoice:read`\n\n**Paramètres de
pagination** :\n- `page` : numéro de page (0-indexé)\n- `pageSize` : résultats par page
(défaut 20, max 100)\n\n**Filtres disponibles** :\n- `direction` : `SENT` (émises) /
`RECEIVED` (reçues)\n- `status` : `DRAFT`, `SIGNED`, `ACCEPTED_TTN`, `REJECTED`\n-
`invoiceNumber` : recherche partielle\n- `startDate` / `endDate` : plage de
dates\n\n**Réponse** : `{ content: [...], totalElements, totalPages, number, size }`"
  },
  "response": [
    {
      "name": "200 OK",
      "originalRequest": {
        "method": "GET",
        "header": [],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices?page=0&pageSize=20&direction=SENT",
          "host": [
            "{{base_url}}/api/core-proxy/invoices?page=0&pageSize=20&direction=SENT"
          ]
        }
      },
      "status": "OK",
      "code": 200,
      "_postman_previewlanguage": "json",
      "header": [
        {
          "key": "Content-Type",
          "value": "application/json"
        }
      ],
      "cookie": [],
      "body": "{\n  \"content\": [\n    {\n      \"id\": \"inv_xxxxxxxxxxxxxxxx\",\n
\"invoiceNumber\": \"FACT-2026-000001\",\n      \"invoiceTypeCode\": \"I-11\",\n
\"invoiceDate\": \"2026-07-11\",\n      \"status\": \"ACCEPTED_TTN\",\n
\"senderName\": \"Ma Société SARL\",\n      \"receiverName\": \"Mon Client SARL\",\n
\"totalAmountExclTax\": 1000,\n      \"totalTaxAmount\": 195,\n
\"totalAmountInclTax\": 1195,\n      \"currencyCode\": \"TND\",\n
\"ttnGeneratedRef\": \"TTN-REF-000123\",\n      \"direction\": \"SENT\"\n    }\n  ],\n
\"page\": 0,\n  \"pageSize\": 20,\n  \"totalElements\": 1,\n  \"totalPages\": 1\n}"
    }
  ]
},
{
  "name": "Détail d'une facture",
  "request": {
    "method": "GET",
```

```
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer {{access_token}}"
        }
      ],
      "url": {
        "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}",
        "host": [
          "{{base_url}}"
        ],
        "path": [
          "api",
          "core-proxy",
          "invoices",
          "{{invoice_id}}"
        ]
      },
      "description": "Récupérer les détails complets d'une facture par son ID interne Fatoora.\n\n**Scope requis** : `invoice:read`\n\n**Réponse** : `{ invoice, teifJson, teifXml, ttnQrCodeBase64, localSignedXml, validationErrors, ttnAcknowledgments }` — l'objet facture est dans le champ `invoice`, avec le JSON et le XML TEIF générés inclus directement dans la réponse.\n\n**Statuts possibles** (`invoice.status`) :\n- `DRAFT` / `SUBMITTED` / `VALIDATED` : en amont de la signature\n- `SIGNED` / `SENT_TTN` : signée, en attente ou en cours d'envoi TTN\n- `ACCEPTED_TTN` : acceptée par TTN (Tunisia Trade Net)\n- `REJECTED_TTN` : rejetée\n\n**Note** : il n'existe pas d'endpoint dédié pour le statut seul — lisez le champ `status` de cette réponse."
    },
    "response": [
      {
        "name": "200 OK",
        "originalRequest": {
          "method": "GET",
          "header": [],
          "url": {
            "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}",
            "host": [
              "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}"
            ]
          }
        },
        "status": "OK",
        "code": 200,
        "_postman_previewlanguage": "json",
```

```
      "header": [
        {
          "key": "Content-Type",
          "value": "application/json"
        }
      ],
      "cookie": [],
      "body": "{\n  \"invoice\": {\n    \"id\": \"xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx\",\n    \"orgId\": \"{{org_id}}\",\n    \"invoiceNumber\": \"FACT-2026-
000001\",\n    \"invoiceTypeCode\": \"I-11\",\n    \"invoiceDate\": \"2026-07-12\",\n
\"status\": \"VALIDATED\",\n    \"senderName\": \"Ma Société SARL\",\n
\"receiverName\": \"Mon Client SARL\",\n    \"receiverIdentifierType\": \"I-01\",\n
\"receiverIdentifierValue\": \"1234567RAM000\",\n    \"totalAmountExclTax\": 1000,\n
\"totalTaxAmount\": 195,\n    \"totalAmountInclTax\": 1195,\n    \"currencyCode\":
\"TND\",\n    \"ttnGeneratedRef\": null,\n    \"ttnIdSaveEfact\": null,\n
\"createdAt\": \"2026-07-11T20:05:48.103042\",\n    \"submittedAt\": \"2026-07-
11T20:05:48.102170\",\n    \"signedAt\": null,\n    \"createdBy\": \"{{client_id}}\",\n
\"isQuotation\": false,\n    \"quotationStatus\": null,\n    \"duplicateCopy\": false,\n
\"importSource\": null,\n    \"importBlockedResign\": false,\n    \"direction\":
\"SENT\",\n    \"partnerAppId\": \"{{partner_app_id}}\"\n  },\n  \"teifJson\": \"
{\\\"body\\\":{\\\"bgm\\\":{...}},\\\"header\\\":{...},\\\"version\\\":\\\"1.8.8\\\"}\",\n
\"teifXml\": \"<TEIF controlingAgency=\\\"TTN\\\" version=\\\"1.8.8\\\">...</TEIF>\",\n
\"ttnQrCodeBase64\": null,\n  \"localSignedXml\": null,\n  \"validationErrors\": null,\n
\"ttnAcknowledgments\": null\n}"
    },
    {
      "name": "404 Not Found",
      "originalRequest": {
        "method": "GET",
        "header": [],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}",
          "host": [
            "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}"
          ]
        }
      },
      "status": "Not Found",
      "code": 404,
      "_postman_previewlanguage": "json",
      "header": [
        {
          "key": "Content-Type",
          "value": "application/json"
        }
      ],
      "cookie": [],
```

```
            "body": "{\n  \"error\": \"InvoiceNotFoundException\",\n  \"message\": \"Invoice not
        found.\"\n}"
          }
        ]
      }
    ]
  },
  {
    "name": "📋 Factures — XML TEIF",
    "item": [
      {
        "name": "Soumettre une facture (XML TEIF)",
        "event": [
          {
            "listen": "test",
            "script": {
              "type": "text/javascript",
              "exec": [
                "if (pm.response.code === 201 || pm.response.code === 200) {",
                "  const json = pm.response.json();",
                "  if (json.id) {",
                "    pm.collectionVariables.set('invoice_id', json.id);",
                "    console.log('✅ invoice_id sauvegardé :', json.id);",
                "  }",
                "} else {",
                "  console.error('❌ Facture non créée, invoice_id inchangé:',
        pm.response.text());",
                "}"
              ]
            }
          }
        ],
        "request": {
          "method": "POST",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/xml"
            },
            {
              "key": "Authorization",
```

```
            "value": "Bearer {{access_token}}"
          }
        ],
        "body": {
          "mode": "raw",
          "raw": "<TEIF controlingAgency=\"TTN\" version=\"1.8.8\">\n<InvoiceHeader>\n
<MessageSenderIdentifier type=\"I-01\">{{seller_tax_id}}
</MessageSenderIdentifier>\n<MessageRecieverIdentifier type=\"I-01\">{{buyer_tax_id}}
</MessageRecieverIdentifier>\n</InvoiceHeader>\n<InvoiceBody>\n<Bgm>\n<DocumentIdentifier>
{{invoice_number}}</DocumentIdentifier>\n<DocumentType code=\"I-
11\">Facture</DocumentType>\n</Bgm>\n<Dtm>\n<DateText format=\"ddMMyy\" functionCode=\"I-
31\">{{invoice_date_ddMMyy}}</DateText>\n</Dtm>\n<PartnerSection>\n<PartnerDetails
functionCode=\"I-63\">\n<Nad>\n<PartnerIdentifier type=\"I-01\">{{seller_tax_id}}
</PartnerIdentifier>\n<PartnerName nameType=\"Physical\">{{seller_name}}
</PartnerName>\n<PartnerAdresses lang=\"fr\">\n<AdressDescription>{{seller_address_desc}}
</AdressDescription>\n<Street>{{seller_street}}</Street>\n<CityName>{{seller_city}}
</CityName>\n<PostalCode>{{seller_postal_code}}</PostalCode>\n<Country
codeList=\"ISO_3166-1\">TN</Country>\n</PartnerAdresses>\n</Nad>\n<CtaSection>\n<Contact
functionCode=\"I-91\">\n<ContactIdentifier>{{seller_phone}}
</ContactIdentifier>\n<ContactName>{{seller_contact_name}}
</ContactName>\n</Contact>\n<Communication>\n<ComMeansType>I-
102</ComMeansType>\n<ComAdress>{{seller_phone}}
</ComAdress>\n</Communication>\n</CtaSection>\n<CtaSection>\n<Contact functionCode=\"I-
91\">\n<ContactIdentifier>{{seller_contact_id}}</ContactIdentifier>\n<ContactName>
{{seller_contact_name}}</ContactName>\n</Contact>\n<Communication>\n<ComMeansType>I-
101</ComMeansType>\n<ComAdress>{{seller_email}}
</ComAdress>\n</Communication>\n</CtaSection>\n</PartnerDetails>\n<PartnerDetails
functionCode=\"I-64\">\n<Nad>\n<PartnerIdentifier type=\"I-01\">{{buyer_tax_id}}
</PartnerIdentifier>\n<PartnerName nameType=\"Physical\">{{buyer_name}}
</PartnerName>\n<PartnerAdresses lang=\"fr\">\n<AdressDescription>{{buyer_address_desc}}
</AdressDescription>\n<Street>{{buyer_street}}</Street>\n<CityName>{{buyer_city}}
</CityName>\n<PostalCode>{{buyer_postal_code}}</PostalCode>\n
<Country codeList=\"ISO_3166-
1\">TN</Country>\n</PartnerAdresses>\n</Nad>\n</PartnerDetails>\n</PartnerSection>\n<LinSe
ction>\n<Lin>\n<ItemIdentifier>{{line1_item_id}}</ItemIdentifier>\n<LinImd
lang=\"fr\">\n<ItemCode>{{line1_item_code}}</ItemCode>\n<ItemDescription>
{{line1_description}}</ItemDescription>\n</LinImd>\n<LinQty>\n<Quantity measurementUnit=\"
{{line1_unit}}\">{{line1_qty}}</Quantity>\n</LinQty>\n<LinTax>\n<TaxTypeName code=\"I-
1602\">TVA</TaxTypeName>\n<TaxCategory>S</TaxCategory>\n<TaxDetails>\n<TaxRate>
{{total_tva_pct}}</TaxRate>\n</TaxDetails>\n</LinTax>\n<LinMoa>\n<MoaDetails>\n<Moa
amountTypeCode=\"I-188\" currencyCodeList=\"ISO_4217\">\n<Amount
currencyIdentifier=\"TND\">{{line1_ht}}
</Amount>\n</Moa>\n</MoaDetails>\n<MoaDetails>\n<Moa amountTypeCode=\"I-171\"
currencyCodeList=\"ISO_4217\">\n<Amount currencyIdentifier=\"TND\">{{line1_ht}}
</Amount>\n</Moa>\n</MoaDetails>\n</LinMoa>\n</Lin>\n</LinSection>\n<InvoiceMoa>\n<AmountD
etails>\n<Moa amountTypeCode=\"I-172\" currencyCodeList=\"ISO_4217\">\n<Amount
currencyIdentifier=\"TND\">{{total_ht}}
</Amount>\n</Moa>\n</AmountDetails>\n<AmountDetails>\n<Moa amountTypeCode=\"I-181\"
currencyCodeList=\"ISO_4217\">\n<Amount currencyIdentifier=\"TND\">{{total_taxes}}
</Amount>\n</Moa>\n</AmountDetails>\n<AmountDetails>\n<Moa amountTypeCode=\"I-180\"
currencyCodeList=\"ISO_4217\">\n<Amount currencyIdentifier=\"TND\">{{total_ttc}}
</Amount>\n</Moa>\n</AmountDetails>\n<AmountDetails>\n<Moa amountTypeCode=\"I-177\"
currencyCodeList=\"ISO_4217\">\n<Amount currencyIdentifier=\"TND\">{{total_ht}}
</Amount>\n</Moa>\n</AmountDetails>\n<AmountDetails>\n<Moa amountTypeCode=\"I-176\"
currencyCodeList=\"ISO_4217\">\n<Amount currencyIdentifier=\"TND\">{{total_ht}}
</Amount>\n</Moa>\n</AmountDetails>\n</InvoiceMoa>\n<InvoiceTax>\n<InvoiceTaxDetails>\n<Ta
x>\n<TaxTypeName code=\"I-
1602\">TVA</TaxTypeName>\n<TaxCategory>S</TaxCategory>\n<TaxDetails>\n<TaxRate>
{{total_tva_pct}}</TaxRate>\n</TaxDetails>\n</Tax>\n<AmountDetails>\n<Moa
amountTypeCode=\"I-178\" currencyCodeList=\"ISO_4217\">\n<Amount
currencyIdentifier=\"TND\">{{total_tva}}
</Amount>\n</Moa>\n</AmountDetails>\n</InvoiceTaxDetails>\n<InvoiceTaxDetails>\n<Tax>\n<Ta
xTypeName code=\"I-
1601\">Timbre</TaxTypeName>\n<TaxCategory>O</TaxCategory>\n<TaxDetails>\n<TaxRate>0</TaxRa
te>\n</TaxDetails>\n</Tax>\n<AmountDetails>\n<Moa amountTypeCode=\"I-178\"
currencyCodeList=\"ISO_4217\">\n<Amount currencyIdentifier=\"TND\">{{total_timbre}}
</Amount>\n</Moa>\n</AmountDetails>\n</InvoiceTaxDetails>\n</InvoiceTax>\n</InvoiceBody>\n
</TEIF>",
        "options": {
```

```
            "raw": {
              "language": "xml"
            }
          }
        },
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices/submit-xml",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "core-proxy",
            "invoices",
            "submit-xml"
          ]
        }
      },
      "response": [
        {
          "name": "201 Created",
          "originalRequest": {
            "method": "POST",
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/invoices/submit-xml",
              "host": [
                "{{base_url}}/api/core-proxy/invoices/submit-xml"
              ]
            }
          },
          "status": "Created",
          "code": 201,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
```

```
      "cookie": [],
      "body": "{\n  \"id\": \"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\",\n  \"orgId\": \"
{{org_id}}\",\n  \"invoiceNumber\": \"FACT-2026-000001\",\n  \"invoiceTypeCode\": \"I-
11\",\n  \"invoiceDate\": \"2026-07-12\",\n  \"status\": \"VALIDATED\",\n  \"senderName\":
\"Ma Société SARL\",\n  \"receiverName\": \"Mon Client SARL\",\n
\"receiverIdentifierType\": \"I-01\",\n  \"receiverIdentifierValue\": \"1234567RAM000\",\n
\"totalAmountExclTax\": 1000,\n  \"totalTaxAmount\": 195,\n  \"totalAmountInclTax\":
1195,\n  \"currencyCode\": \"TND\",\n  \"ttnGeneratedRef\": null,\n  \"ttnIdSaveEfact\":
null,\n  \"createdAt\": \"2026-07-11T20:05:48.103042\",\n  \"submittedAt\": \"2026-07-
11T20:05:48.102170\",\n  \"signedAt\": null,\n  \"createdBy\": \"{{client_id}}\",\n
\"isQuotation\": false,\n  \"quotationStatus\": null,\n  \"duplicateCopy\": false,\n
\"importSource\": null,\n  \"importBlockedResign\": false,\n  \"direction\": \"SENT\",\n
\"partnerAppId\": \"{{partner_app_id}}\"\n}"

    }

  ]

},

{

  "name": "Valider un XML (XSD uniquement)",

  "request": {

    "method": "POST",

    "header": [

      {

        "key": "Content-Type",

        "value": "application/json"

      },

      {

        "key": "Authorization",

        "value": "Bearer {{access_token}}"

      }

    ],

    "body": {

      "mode": "raw",

      "raw": "{\n  \"xml\": \"<TEIF controlingAgency=\\\"TTN\\\" version=\\\"1.8.8\\\">
<InvoiceHeader><MessageSenderIdentifier type=\\\"I-
01\\\">1234567ABC000</MessageSenderIdentifier><MessageRecieverIdentifier type=\\\"I-
01\\\">1234567RAM000</MessageRecieverIdentifier></InvoiceHeader><InvoiceBody><!—— ...
corps complet ... ——></InvoiceBody></TEIF>\"\n}",

      "options": {

        "raw": {

          "language": "json"

        }

      }

    },

    "url": {

      "raw": "{{base_url}}/api/core-proxy/teif/validate",

      "host": [

        "{{base_url}}"
```

```
        ],
        "path": [
          "api",
          "core-proxy",
          "teif",
          "validate"
        ]
      },
      "description": "Valider un XML TEIF contre le schéma XSD sans créer de facture. Utile pour tester votre XML avant soumission.\n\n**Attention au chemin** : c'est bien `/api/core-proxy/teif/validate` (pas `/api/core-proxy/invoices/teif/validate-xml-xsd`, qui 404 — c'est le chemin interne de fatooraCore, jamais exposé tel quel par le BFF). Le champ du corps est `xml`, pas `xmlContent`."
    },
    "response": [
      {
        "name": "200 OK — valid",
        "originalRequest": {
          "method": "POST",
          "header": [],
          "url": {
            "raw": "{{base_url}}/api/core-proxy/teif/validate",
            "host": [
              "{{base_url}}/api/core-proxy/teif/validate"
            ]
          }
        },
        "status": "OK",
        "code": 200,
        "_postman_previewlanguage": "json",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/json"
          }
        ],
        "cookie": [],
        "body": "{\n  \"valid\": true,\n  \"issues\": []\n}"
      },
      {
        "name": "200 OK — invalid",
        "originalRequest": {
```

```
            "method": "POST",
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/teif/validate",
              "host": [
                "{{base_url}}/api/core-proxy/teif/validate"
              ]
            }
          },
          "status": "OK",
          "code": 200,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "{\n  \"valid\": false,\n  \"issues\": [\n    {\n      \"type\":
      \"XML_SCHEMA_VIOLATION\",\n      \"message\": \"schema rule (cvc-)complex-type.2.4.a:
      Invalid content was found starting with element 'NotValid'. One of '{InvoiceHeader}' is
      expected.\",\n      \"severity\": \"ERROR\",\n      \"xpath\": \"line:1\"\n    }\n  ]\n}"
        }
      ]
    }
  ]
},
{
  "name": "✍️ Signature et envoi TTN",
  "item": [
    {
      "name": "Signer et envoyer à TTN (SEAL)",
      "event": [
        {
          "listen": "test",
          "script": {
            "type": "text/javascript",
            "exec": [
              "const json = pm.response.json();",
              "if (json.jobId) {",
              "  pm.collectionVariables.set('job_id', json.jobId);",
```

```
              "  console.log('✅ job_id sauvegardé :', json.jobId);",
              "} else {",
              "  console.error('❌ Pas de jobId:', JSON.stringify(json));",
              "}"
            ]
          }
        }
      ],
      "request": {
        "method": "POST",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/json"
          },
          {
            "key": "Authorization",
            "value": "Bearer {{access_token}}"
          }
        ],
        "body": {
          "mode": "raw",
          "raw": "{\n  \"signatureType\": \"SEAL\"\n}",
          "options": {
            "raw": {
              "language": "json"
            }
          }
        },
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/sign-and-send",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "core-proxy",
            "invoices",
            "{{invoice_id}}",
            "sign-and-send"
```

```
      ]
    },
    "description": "Déclencher la signature automatique SEAL et l'envoi à TTN (Tunisia Trade Net) en une seule opération.\n\n**Scopes requis** : `invoice:write` + `seal:sign`\n\n**Pré-requis** : L'organisation cliente doit avoir accordé le scope `seal:sign` ET avoir une configuration SEAL active.\n\n**Vérifiez** `allowedScopes` de l'étape 0 avant d'appeler cet endpoint.\n\nPour DigiGO, utilisez `\"signatureType\": \"DIGIGO\"` avec le scope `digigo:sign`.\n\nLe script de test sauvegarde automatiquement le `jobId` dans la variable `job_id` — utilisé ensuite pour interroger `GET /api/core-proxy/jobs/{{job_id}}`."
  },
  "response": [
    {
      "name": "200 OK — job created (SEAL)",
      "originalRequest": {
        "method": "POST",
        "header": [],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/sign-and-send",
          "host": [
            "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/sign-and-send"
          ]
        }
      },
      "status": "OK",
      "code": 200,
      "_postman_previewlanguage": "json",
      "header": [
        {
          "key": "Content-Type",
          "value": "application/json"
        }
      ],
      "cookie": [],
      "body": "{\n  \"jobId\": \"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\",\n  \"invoiceId\": \"{{invoice_id}}\",\n  \"jobType\": \"SIGN_AND_SUBMIT_SEAL\",\n  \"status\": \"PENDING\",\n  \"success\": true,\n  \"message\": \"Invoice queued for signature and TTN submission\"\n}"
    },
    {
      "name": "402 Payment Required — no signature credits",
      "originalRequest": {
        "method": "POST",
        "header": [],
        "url": {
```

```
            "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/sign-and-send",
            "host": [
              "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/sign-and-send"
            ]
          }
        },
        "status": "Payment Required",
        "code": 402,
        "_postman_previewlanguage": "json",
        "header": [
          {
            "key": "Content-Type",
            "value": "application/json"
          }
        ],
        "cookie": [],
        "body": "{\n  \"error\": \"NO_SIGNATURE_CREDITS\",\n  \"message\": \"No signing quota or signature-pack credits available. Buy a pack or subscribe/upgrade to sign and submit to TTN.\",\n  \"packsRemaining\": 0\n}"
      }
    ]
  },
  {
    "name": "Suivre le job de signature (poll)",
    "request": {
      "method": "GET",
      "header": [
        {
          "key": "Authorization",
          "value": "Bearer {{access_token}}"
        }
      ],
      "url": {
        "raw": "{{base_url}}/api/core-proxy/jobs/{{job_id}}",
        "host": [
          "{{base_url}}"
        ],
        "path": [
          "api",
          "core-proxy",
          "jobs",
```

```
            "{{job_id}}"
          ]
        },
        "description": "Interroger le statut du job créé par `sign-and-send` (asynchrone). `status` passe de `PENDING` à `COMPLETED` ou `FAILED`. En cas d'échec, `outputData` détaille l'étape fautive (ex. certificat SEAL/DigiGO non configuré).\n\n**Scope requis** : `invoice:read`"
      },
      "response": [
        {
          "name": "200 OK — completed",
          "originalRequest": {
            "method": "GET",
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/jobs/{{job_id}}",
              "host": [
                "{{base_url}}/api/core-proxy/jobs/{{job_id}}"
              ]
            }
          },
          "status": "OK",
          "code": 200,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "{\n  \"jobId\": \"{{job_id}}\",\n  \"invoiceId\": \"{{invoice_id}}\",\n  \"jobType\": \"SIGN_AND_SUBMIT_SEAL\",\n  \"status\": \"COMPLETED\",\n  \"progress\": 100,\n  \"currentStep\": \"TTN_SUBMISSION\",\n  \"errorMessage\": null,\n  \"errorData\": null,\n  \"outputData\": {\n    \"step0CertVerification\": {\n      \"valid\": true,\n      \"success\": true\n    }\n  },\n  \"signatureResult\": {\n    \"signed\": true\n  },\n  \"ttnSubmissionResult\": {\n    \"accepted\": true,\n    \"ttnGeneratedRef\": \"TTN-REF-000123\"\n  },\n  \"ttnFinalResult\": {\n    \"status\": \"ACCEPTED_TTN\"\n  },\n  \"createdAt\": \"2026-07-11T20:07:14.706Z\",\n  \"startedAt\": \"2026-07-11T20:07:14.836Z\",\n  \"completedAt\": \"2026-07-11T20:07:16.943Z\",\n  \"retryCount\": 0\n}"
        },
        {
          "name": "200 OK — failed (missing signing credential)",
          "originalRequest": {
            "method": "GET",
```

```
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/jobs/{{job_id}}",
              "host": [
                "{{base_url}}/api/core-proxy/jobs/{{job_id}}"
              ]
            }
          },
          "status": "OK",
          "code": 200,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "{\n  \"jobId\": \"{{job_id}}\",\n  \"invoiceId\": \"{{invoice_id}}\",\n
      \"jobType\": \"SIGN_AND_SUBMIT_DIGIGO\",\n  \"status\": \"FAILED\",\n  \"progress\": 0,\n
      \"currentStep\": \"CERT_VERIFICATION\",\n  \"errorMessage\": null,\n  \"errorData\":
      null,\n  \"outputData\": {\n    \"step0CertVerification\": {\n      \"error\":
      \"Credential ID DigiGO (email) non configuré pour cette organisation\",\n      \"valid\":
      false,\n      \"success\": false,\n      \"errorCode\":
      \"MISSING_DIGIGO_CREDENTIAL_ID\",\n      \"certificates\": []\n    }\n  },\n
      \"signatureResult\": null,\n  \"ttnSubmissionResult\": null,\n  \"ttnFinalResult\":
      null,\n  \"createdAt\": \"2026-07-11T20:07:14.706Z\",\n  \"startedAt\": \"2026-07-
      11T20:07:14.836Z\",\n  \"completedAt\": \"2026-07-11T20:07:14.943Z\",\n  \"retryCount\":
      0\n}"
        }
      ]
    }
  ]
},
{
  "name": "📎 Artefacts (XML / PDF)",
  "item": [
    {
      "name": "Télécharger le PDF de la facture",
      "request": {
        "method": "GET",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer {{access_token}}"
```

```
        }
      ],
      "url": {
        "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/pdf",
        "host": [
          "{{base_url}}"
        ],
        "path": [
          "api",
          "core-proxy",
          "invoices",
          "{{invoice_id}}",
          "pdf"
        ],
        "query": [
          {
            "key": "pdfTemplate",
            "value": "",
            "description": "Optionnel — sinon le modèle par défaut de l'organisation cliente est utilisé",
            "disabled": true
          },
          {
            "key": "lang",
            "value": "",
            "description": "Optionnel — langue du document (fr/en/ar)",
            "disabled": true
          }
        ]
      },
      "description": "Réponse binaire (`application/pdf`), quel que soit le statut de la facture (brouillon inclus).\n\n**Scope requis** : `invoice:read`."
    },
    "response": []
  },
  {
    "name": "Récupérer le XML TEIF généré",
    "request": {
      "method": "GET",
      "header": [
        {
```

```
            "key": "Authorization",
            "value": "Bearer {{access_token}}"
          }
        ],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/generated-xml",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "core-proxy",
            "invoices",
            "{{invoice_id}}",
            "generated-xml"
          ]
        },
        "description": "⚠️ **Au-delà du statut SIGNED, cet endpoint régénère un XML NON SIGNÉ à la volée** (il perd la signature et le QR code TTN). Pour l'artefact final réellement signé/horodaté TTN, lisez plutôt le champ `teifXml` retourné par « Détail d'une facture » (`GET /api/core-proxy/invoices/{{invoice_id}}`), pas cet endpoint.\n\n**Scope requis** : `invoice:read`."
      },
      "response": [
        {
          "name": "200 OK",
          "originalRequest": {
            "method": "GET",
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/generated-xml",
              "host": [
                "{{base_url}}/api/core-proxy/invoices/{{invoice_id}}/generated-xml"
              ]
            }
          },
          "status": "OK",
          "code": 200,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
```

```
                "value": "application/json"
              }
            ],
            "cookie": [],
            "body": "{\n  \"note\": \"corps brut application/xml, pas du JSON\"\n}"
          }
        ]
      }
    ]
  },
  {
    "name": "🔔 Webhooks",
    "item": [
      {
        "name": "Créer un webhook",
        "event": [
          {
            "listen": "test",
            "script": {
              "type": "text/javascript",
              "exec": [
                "const json = pm.response.json();",
                "if (json.id) {",
                "  pm.collectionVariables.set('webhook_id', json.id);",
                "  if (json.secret) { pm.collectionVariables.set('webhook_secret', json.secret);
    }",
                "  console.log('✅ webhook_id sauvegardé :', json.id);",
                "  console.log('🔑 webhook_secret (à conserver, non réaffiché) :',
    json.secret);",
                "} else {",
                "  console.error('❌ Webhook non créé :', JSON.stringify(json));",
                "}"
              ]
            }
          }
        ],
        "request": {
          "method": "POST",
          "header": [
            {
              "key": "Content-Type",
```

```
            "value": "application/json"
          },
          {
            "key": "Authorization",
            "value": "Bearer {{access_token}}"
          }
        ],
        "body": {
          "mode": "raw",
          "raw": "{\n  \"url\": \"https://votre-erp.example.com/webhooks/fatoora\",\n
\"events\": [\n    \"invoice.validated\",\n    \"invoice.signed\",\n
\"invoice.accepted_ttn\",\n    \"invoice.rejected_ttn\"\n  ]\n}",
          "options": {
            "raw": {
              "language": "json"
            }
          }
        },
        "url": {
          "raw": "{{base_url}}/api/core-proxy/webhooks",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "core-proxy",
            "webhooks"
          ]
        },
        "description": "Crée un abonnement webhook pour l'organisation cliente ciblée par le
token org-scoped (`{{access_token}}` obtenu à l'étape 1 — aucun header `X-Org-ID` à gérer
manuellement, il est déduit automatiquement du token).\n\n**Scope requis** :
`invoice:write`\n\n**Pré-requis** : votre application partenaire doit être autorisée pour
cette organisation (sinon `401 PARTNER_NOT_AUTHORIZED`).\n\nLe `secret` est retourné **une
seule fois** dans cette réponse — sauvegardez-le, il sert à vérifier la signature `X-HMAC-
Signature` des évènements reçus.\n\n**Types d'évènements par défaut** (si `events` est
omis) : `invoice.created`, `invoice.validated`, `invoice.signed`, `invoice.submitted_ttn`,
`invoice.accepted_ttn`, `invoice.rejected_ttn`, `webhook.test`. Ajoutez explicitement
`invoice.validation_failed` si vous en avez besoin (non inclus par défaut)."
      },
      "response": [
        {
          "name": "201 Created",
          "originalRequest": {
            "method": "POST",
            "header": [],
```

```
            "url": {
              "raw": "{{base_url}}/api/core-proxy/webhooks",
              "host": [
                "{{base_url}}/api/core-proxy/webhooks"
              ]
            }
          },
          "status": "Created",
          "code": 201,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "{\n  \"id\": \"wh_xxxxxxxxxxxxxxxx\",\n  \"appId\": \"{{client_id}}\",\n
\"url\": \"https://votre-erp.example.com/webhooks/fatoora\",\n  \"events\": [\n
\"invoice.validated\",\n    \"invoice.signed\",\n    \"invoice.accepted_ttn\",\n
\"invoice.rejected_ttn\"\n  ],\n  \"status\": \"ACTIVE\",\n  \"secret\":
\"whsec_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\",\n  \"expiresAt\": \"2026-08-14T00:00:00\",\n
\"createdAt\": \"2026-07-15T00:00:00\"\n}"
        },
        {
          "name": "401 Unauthorized — partenaire non autorisé",
          "originalRequest": {
            "method": "POST",
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/webhooks",
              "host": [
                "{{base_url}}/api/core-proxy/webhooks"
              ]
            }
          },
          "status": "Unauthorized",
          "code": 401,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
```

```
        }
      ],
      "cookie": [],
      "body": "{\n  \"errorCode\": \"PARTNER_NOT_AUTHORIZED\",\n  \"message\": \"Partner application is not authorized for this organization.\"\n}"
    }
  ]
},
{
  "name": "Lister les webhooks",
  "request": {
    "method": "GET",
    "header": [
      {
        "key": "Authorization",
        "value": "Bearer {{access_token}}"
      }
    ],
    "url": {
      "raw": "{{base_url}}/api/core-proxy/webhooks",
      "host": [
        "{{base_url}}"
      ],
      "path": [
        "api",
        "core-proxy",
        "webhooks"
      ]
    },
    "description": "Liste les webhooks que **votre application partenaire** a créés pour cette organisation cliente (pas ceux créés par la clé API propre de l'organisation, ni par un autre partenaire).\n\n**Scope requis** : `invoice:read`"
  },
  "response": [
    {
      "name": "200 OK",
      "originalRequest": {
        "method": "GET",
        "header": [],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/webhooks",
          "host": [
```

```
                "{{base_url}}/api/core-proxy/webhooks"
              ]
            }
          },
          "status": "OK",
          "code": 200,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "[\n  {\n    \"id\": \"{{webhook_id}}\",\n    \"appId\": \"
{{client_id}}\",\n    \"url\": \"https://votre-erp.example.com/webhooks/fatoora\",\n
\"events\": [\n      \"invoice.validated\",\n      \"invoice.signed\"\n    ],\n
\"status\": \"ACTIVE\",\n    \"expiresAt\": \"2026-08-14T00:00:00\",\n    \"createdAt\":
\"2026-07-15T00:00:00\"\n  }\n]"
        }
      ]
    },
    {
      "name": "Envoyer un évènement de test",
      "request": {
        "method": "POST",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer {{access_token}}"
          }
        ],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/webhooks/{{webhook_id}}/test",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "core-proxy",
            "webhooks",
            "{{webhook_id}}",
```

```
            "test"
          ]
        },
        "description": "Déclenche un évènement `webhook.test` livré immédiatement à l'URL configurée — utile pour valider votre récepteur (vérification de signature comprise) avant de traiter de vrais évènements de facture.\n\n**Scope requis** : `invoice:write`"
      },
      "response": [
        {
          "name": "200 OK",
          "originalRequest": {
            "method": "POST",
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/webhooks/{{webhook_id}}/test",
              "host": [
                "{{base_url}}/api/core-proxy/webhooks/{{webhook_id}}/test"
              ]
            }
          },
          "status": "OK",
          "code": 200,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
          "body": "{\n  \"status\": \"queued\",\n  \"subscriptionId\": \"{{webhook_id}}\",\n  \"eventType\": \"webhook.test\"\n}"
        }
      ]
    },
    {
      "name": "Supprimer un webhook",
      "request": {
        "method": "DELETE",
        "header": [
          {
            "key": "Authorization",
```

```
            "value": "Bearer {{access_token}}"
          }
        ],
        "url": {
          "raw": "{{base_url}}/api/core-proxy/webhooks/{{webhook_id}}",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "api",
            "core-proxy",
            "webhooks",
            "{{webhook_id}}"
          ]
        },
        "description": "**Scope requis** : `invoice:write`"
      },
      "response": [
        {
          "name": "200 OK",
          "originalRequest": {
            "method": "DELETE",
            "header": [],
            "url": {
              "raw": "{{base_url}}/api/core-proxy/webhooks/{{webhook_id}}",
              "host": [
                "{{base_url}}/api/core-proxy/webhooks/{{webhook_id}}"
              ]
            }
          },
          "status": "OK",
          "code": 200,
          "_postman_previewlanguage": "json",
          "header": [
            {
              "key": "Content-Type",
              "value": "application/json"
            }
          ],
          "cookie": [],
```

```
            "body": "{\n  \"deleted\": true,\n  \"id\": \"{{webhook_id}}\"\n}"
          }
        ]
      }
    ]
  }
]
}
```

---

*This guide reflects the Fatoora Partner API surface as implemented in `fatooraBusiness` (routes: `oauth.routes.ts`, `partner-auth.routes.ts`, `partner-self-register.routes.ts`, `core-proxy.routes.ts`) and `fatooraCore` (`InvoiceController`, `QuotationController`, `WebhookController`), and has been **live-verified end to end** against a dev instance (fatooraLive :3600 / fatooraBusiness :4100, test partner account) on 2026-07-11: OAuth2 token exchange (app-only + org-scoped), org discovery, JSON and XML invoice submission, invoice detail retrieval, sign-and-send (SEAL and DigiGO), job polling, and the blocked `submit-ttn` path were all exercised against live responses. That pass found and corrected several real discrepancies — the `teif/validate` endpoint path/field names were wrong in both this guide and the app's own Postman collection (fixed in `fatooraLive/src/components/partner/credentials/PartnerApiGuide.tsx`), `GET /invoices/:id/status` was confirmed broken (always `500 INTERNAL_ERROR`, no matching backend mapping) and has since been removed outright rather than fixed — it now returns a clean `404`, `GET /invoices/:id` returns a wrapped object rather than a flat invoice, and submit/sign-and-send response shapes needed correction. See inline "verified live" / "confirmed live" notes throughout for details.*

Update pass (2026-07-15): added §7's "Retrieving generated artifacts (XML / PDF)" subsection and a new §9. Webhooks section — webhook management (`POST/GET/DELETE /api/core-proxy/webhooks*`) was unblocked for authorized partner apps at the fatooraCore level (`WebhookController` now uses the same `OrgIdResolver` + `PartnerAuthorizationService` pattern as every invoice endpoint, and webhook dispatch was made org-scoped rather than app-actor-scoped so subscriptions fire regardless of whether the org's own key or an authorized partner triggered the event) — previously it unconditionally returned `401 PARTNER_CONTEXT_NOT_ALLOWED` for any partner token. `event-delivery/mode` and pull-notification endpoints remain org-owned-key-only. This pass also fixed a real authorization gap in `InvoiceController.getGeneratedXml`, which — unlike its sibling read endpoints — never called `verifyPartnerAuthorization`, so any partner token with `invoice:read` could previously fetch the generated XML of an invoice belonging to an organization it was not authorized for; it now

performs the same check as `getInvoice` / `downloadPdf` and correctly returns `401` (rather than a `500`-with-XML-body) on failure. Sections 9-16 were renumbered accordingly (old §9-§16 → §10-§17).

Diagrams are embedded directly as base64 PNG data URIs (self-contained — no dependency on sibling files) but are still rendered from Mermaid sources in `partner-api-diagrams/src/*.mmd` using `@mermaid-js/mermaid-cli` (`mmdc`). To regenerate after editing a source file: `mmdc -i src/<name>.mmd -o <name>.png -b white -s 2 --width 1200 -p puppeteer-config.json` from the `partner-api-diagrams/` directory (the `-p` puppeteer config disables the Chromium sandbox, required in this environment), then re-run the base64-embedding step to refresh the data URI in this file.

The Postman collection JSON in §17 is not hand-written — it's the literal output of `buildPostmanCollection()` in `fatooraLive/src/components/partner/credentials/PartnerApiGuide.tsx`, extracted and executed with Node to guarantee fidelity with the code. The OpenAPI YAML in §17 is hand-authored from this guide's contents, since no partner-scoped OpenAPI source exists in the codebase yet.