DOCUMENTATION DÉVELOPPEUR · PAYT

Guide d'intégration de l'API Payt

Alimenter Payt en débiteurs et factures via l'API REST, pour déclencher le suivi de relance automatisé.

Périmètre : ingestion de données (sens entrant, API REST) · **Version :** 1.1 · **Date :** juin 2026
Base URL production : `https://api.paytsoftware.com` · **Base URL démo :** `https://demo-api.paytsoftware.com`

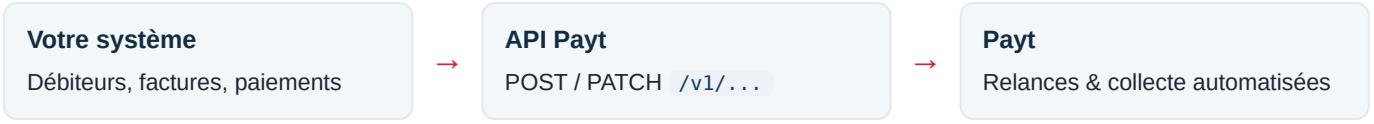
Sommaire

Ce guide se concentre exclusivement sur l'écriture de données *vers* Payt par l'API REST. La récupération des changements depuis Payt (synchronisation par `updated_after` ou webhooks temps réel) ainsi que l'import par fichiers (CSV/XML) font l'objet de documents distincts.

1. Objet et périmètre du guide

Ce guide explique comment votre système (ERP, logiciel de facturation, outil métier) transmet ses débiteurs et ses factures à Payt via l'API REST. Une fois les données reçues, Payt prend en charge automatiquement les rappels, relances et la collecte.

Le flux en une image



Ce que couvre ce guide

- Authentification et permissions nécessaires à l'écriture.
- Création / mise à jour des référentiels (taux de TVA, conditions de paiement), des débiteurs, des contacts et des factures.
- Enregistrement des paiements et blocage/reprise du suivi d'une facture.
- La lecture correcte des réponses et la gestion des erreurs.

Hors périmètre

La synchronisation *sortante* (récupérer paiements et statuts depuis Payt) et les webhooks temps réel ne sont pas traités ici. L'import par fichiers (CSV/XML) non plus : ce guide décrit la voie API REST, qui en est une alternative — **les deux voies ne se cumulent pas sur une même administration** (voir §8).

2. Prérequis et conventions

Prérequis

- Le **module API** doit être activé sur l'administration cible (à demander à votre interlocuteur Payt).
- Un identifiant d'administration (`administration_id`) : il est requis dans chaque appel.
- Une application OAuth2 ou un token statique (voir §4).

Environnements

Démo et production sont deux environnements **totalement séparés** : ni les tokens, ni les applications, ni les données ne sont partagés. Pour passer de l'un à l'autre, il suffit de changer les sous-domaines.

Usage	Application web	Base URL API
Production	`app.paytsoftware.com`	`api.paytsoftware.com`
Démo / tests	`demo.paytsoftware.com`	`demo-api.paytsoftware.com`

Un compte démo se crée via l'inscription classique sur `demo.paytsoftware.com`. Développez et validez toujours en démo avant de basculer en production.

Conventions transverses

Règle	Détail
Transport	HTTPS obligatoire, TLS 1.2 ou supérieur.
Montants	Toujours des **chaînes** à deux décimales, ex. `"1200.00"` (jamais un nombre).
Dates & heures	UTC, format ISO 8601, ex. `"2026-06-01"` ou `"2026-06-01T08:00:00Z"` .
Lot par requête	Les endpoints d'écriture acceptent **plusieurs records** dans un même appel (tableau `debtors` , `invoices` …). Privilégiez le batch.
Champ `fields`	Sur les endpoints de lecture, `fields` doit être passé comme un **tableau**, pas une chaîne (sinon erreur 422).

3. Hiérarchie et objets de données Payt

Avant d'écrire, il faut savoir où atterrissent les données. La doc Payt décrit, dans son chapitre *Models*, les objets dont les statuts et comportements méritent une explication. Voici la hiérarchie et les objets que vous manipulerez.

La hiérarchie

Company
Entité de tête : plan de facturation + comptes utilisateurs. Contient une ou plusieurs administrations.

→

Administration
L'entité opérationnelle qui gère factures et débiteurs. C'est l' `administration_id` présent dans chaque appel.

→

Objets métier
Débiteurs, contacts, factures (+ lignes, paiements), référencés sur des taux de TVA et conditions de paiement.

À retenir pour l'ingestion

Dans une intégration classique, **Company et Administration préexistent** (créées lors de l'onboarding Payt). Vous écrivez *dans* une administration. Leur création par API relève de l'intégration in-app (`sign_up`), hors périmètre de ce guide.

Company — statuts

La **company** est l'entité de tête. Toute nouvelle company démarre en période d'**essai** (30 jours après activation, sans facturation ; certaines fonctions limitées).

Statut	Signification
`unconfirmed`	Créée mais conditions Payt non encore acceptées. Connexion impossible. Supprimée après 30 jours si non confirmée.
`implementation`	Confirmée, mais aucune administration encore activée.
`active`	État normal — activée dès la première administration activée.
`deactivated`	Désactivée (essai/contrat expiré, impayés, risque…). Toutes les administrations sont désactivées, plus aucune communication débiteur.
`deleted`	Marquée pour suppression ; effacée définitivement après 30 jours.

Administration — statuts et pause

L'**administration** représente une entité (société, organisation, département) qui gère ses factures et débiteurs.

Statut	Signification
`implementation`	Créée mais pas encore activée.
`active`	Activée (checklist d'implémentation complétée) — la communication avec les débiteurs est active. État normal.
`deactivated`	Désactivée : tous les processus automatiques s'arrêtent, mais la communication individuelle avec un débiteur reste possible.
`deleted`	Marquée pour suppression ; effacée après 30 jours. Reste consultable via l'endpoint administrations.

Pause

Une administration peut être **mise en pause** (état temporaire) : soit totale, soit limitée aux relances (auquel cas les nouvelles factures partent quand même). Le statut reste `active` et la facturation continue.

Les objets que vous manipulez

Objet	Rôle	Sens
Débiteur	Vos clients.	**écriture**
Contact	Coordonnées (emails, téléphone) rattachées à un débiteur.	**écriture**
Facture	La créance ; porte lignes, paiements et document.	**écriture**
Taux de TVA · Condition de paiement	Référentiels que les factures référencent.	**écriture**
Fichier (File)	Document PDF/UBL attaché à une facture, via un flux d'upload dédié (voir §6).	**écriture**
Notification	Évènements émis par Payt (paiement, étape de relance, mise en recouvrement, solvabilité…).	**lecture**

Le détail des statuts de facture, des types de notifications et du flux fichiers est documenté dans le chapitre *Models* de la doc (Administration, Company, Invoice, File, Notification).

4. Authentification et permissions

Chaque requête porte un jeton dans l'en-tête `Authorization`. Deux modes :

Mode	Quand l'utiliser	Sécurité
OAuth2 Authorization Code (recommandé)	Intégration destinée à plusieurs clients Payt, ou dès que possible.	Élevée — l'utilisateur accorde des permissions et choisit ses administrations.
Token statique	Intégration interne simple, quand OAuth2 n'est pas envisageable.	Moindre — à réserver aux cas où OAuth2 est impossible.

Flux OAuth2 (résumé)

1. Créez une **application** dans Payt et sélectionnez les permissions (scopes) au plus juste.
2. Redirigez l'utilisateur vers le formulaire d'autorisation ; il accorde l'accès et choisit ses administrations.
3. Payt renvoie un `code` (valable **10 minutes**) sur votre `redirect_uri`.
4. Échangez ce `code` contre un jeton via `POST /oauth/token`.

Cycle de vie des jetons

Jeton	Validité	Note
Access token	2 heures	À renouveler via le refresh token.
Refresh token	90 jours	Renouvelé à chaque usage. Tant que l'app l'utilise au moins une fois tous les 90 jours, il n'expire jamais en pratique.

En-tête de requête

```
GET https://api.paytsoftware.com/v1/invoices?administration_id=81
Authorization: Bearer IBzLDErQvt9gOmSLarUtDy06emduHZmKEG2OSPdHpJ8
```

Permissions nécessaires à l'ingestion

Les permissions sont des **scopes OAuth2**. Pour écrire des données, votre application a besoin des scopes d'écriture correspondants (par ex. `debtors:write`, `invoices:write`, et selon les ressources écrites `contacts:write`, etc.).

> **Gérer les 403**
>
> Si un scope manque, l'API renvoie `403 forbidden` en indiquant **le nom du scope requis** dans le message. Un utilisateur peut aussi retirer un scope à tout moment : votre client doit savoir intercepter un 403 et informer que l'action est indisponible tant que la permission n'est pas réaccordée.

```
{
  "code": "forbidden",
  "message": "Missing permission: invoices:write"
}
```

5. Ordre d'écriture des ressources

Toutes les écritures sont des **upsert** (create-or-update) basés sur vos identifiants externes : Payt crée le record s'il n'existe pas, le met à jour sinon. Rejouer la même requête ne crée pas de doublon — l'intégration est donc **idempotente** par construction.

Clés d'upsert par ressource

Ressource	Clé(s) d'identification externe
Taux de TVA	`code`
Conditions de paiement	`code`
Débiteur	`debtor_number` et/ou `debtor_identifier`
Contact	`contact_identifier`
Facture	`invoice_number` et/ou `invoice_identifier`
Paiement	`origin_identifier` (unique dans l'administration)

Le flux recommandé par Payt

La doc (*Models* > *Invoice*) décrit l'ordre recommandé pour créer des factures : on crée d'abord les contacts, puis les débiteurs, on téléverse les documents, et enfin on crée les factures.

1 · Contacts
Coordonnées (emails)

→

2 · Débiteurs
Les clients

→

3 · Fichiers
Upload des PDF (§6)

→

4 · Factures
(+ lignes, paiements)

À ce flux s'ajoutent les **référentiels** en amont : un `vat_rate_code` utilisé sur une ligne de facture **doit être présent** dans l'administration (exigence explicite de la doc). Par cohérence, créez de même vos conditions de paiement avant de les référencer.

Précision de périmètre

La doc présente cet ordre comme une **recommandation**. Le caractère strictement bloquant de l'absence d'un débiteur ou d'une condition de paiement (erreur vs création/liaison à la volée) n'est pas explicité ; en pratique, suivez le flux ci-dessus, qui évite toute ambiguïté.

6. Procédure pas à pas

Toutes les requêtes ci-dessous sont des `POST` (ou `PATCH`) vers `https://api.paytsoftware.com` avec l'en-tête `Authorization: Bearer <access_token>` et `Content-Type: application/json`. Les exemples utilisent l'administration `81`.

Étape 0 — Référentiels (une fois)

Taux de TVA — `POST /v1/vat_rates`

```
{
  "administration_id": "81",
  "vat_rates": [
    { "code": "STD20", "rate": "20.0", "name": "TVA 20%", "category": "regular" },
    { "code": "RED10", "rate": "10.0", "name": "TVA 10%", "category": "regular" }
  ]
}
```

`category` ∈ { regular, zero, none, exempt, reversed, export, export_within_eea }.

Conditions de paiement — `POST /v1/payment_conditions`

```
{
  "administration_id": "81",
  "payment_conditions": [
    { "code": "30J", "name": "30 jours net" }
  ]
}
```

Étape 1 — Débiteurs — `POST /v1/debtors`

Champs **requis** : `debtor_number`, `name`, `debtor_identifier`.

```
{
  "administration_id": "81",
  "debtors": [
    {
      "debtor_number": "CLI-1001",
      "name": "Boulangerie Martin SARL",
      "debtor_identifier": "erp-uuid-8f3a",
      "is_company": true,
      "country_code": "FR",
      "language_code": "fr",
      "vat_number": "FR40303265045",
      "contact_identifier": "contact-1001"
    }
  ]
}
```

Champ	Rôle
`debtor_number` *	Code lisible identifiant le débiteur.
`debtor_identifier` *	Identifiant externe stable (idéalement l'ID base de votre ERP).
`contact_identifier`	Relie le débiteur à un contact (voir étape 1bis) pour les emails.
`country_code` / `language_code`	ISO 3166 alpha-2 / code langue — pilotent la langue des relances.
`is_company` , `vat_number` , `coc_number`	Nature et identifiants légaux du débiteur.

Étape 1bis — Contacts (recommandé) — `POST /v1/contacts`

Sans adresse email, pas de relance par email. Fournissez les contacts et leurs adresses, reliés au débiteur via `contact_identifier`.

Le flux recommandé par Payt (§5) crée les contacts *avant* les débiteurs qui les référencent ; comme tout est upsert, l'essentiel est qu'ils soient présents avant la facture.

```
{
  "administration_id": "81",
  "contacts": [
    {
      "contact_identifier": "contact-1001",
      "firstname": "Claire",
      "lastname": "Martin",
      "invoice_email_address": "compta@boulangerie-martin.fr",
      "reminder_email_address": "compta@boulangerie-martin.fr"
    }
  ]
}
```

Adresses dédiées possibles : `invoice_email_address` , `reminder_email_address` (+ variantes `_cc_` / `_bcc_`), `default_email_address` , `sms_phone_number` , etc.

Étape 1ter — Upload des documents (optionnel)

Pour joindre le PDF (ou l'UBL) d'une facture, Payt utilise un flux en **deux temps**, à réaliser **avant** la création de la facture.

① Déclarer les fichiers — `POST /v1/files`

Par fichier : `byte_size` , `checksum` (SHA3-512 encodé en base64) et `content_type` — tous **requis**.

```
{
  "administration_id": "81",
  "files": [
    {
      "byte_size": 43616,
      "checksum": "Dr/LPU7TQVK4pw/8wXK8pEvW0c/pCJHtB/BbWK0Csfb2...",
      "content_type": "application/pdf"
    }
  ]
}
```

② Récupérer les URL présignées

Payt renvoie, **uniquement pour les fichiers pas encore présents**, un objet `{ checksum, url, expires_at }`. Les fichiers déjà connus ne sont pas renvoyés (rien à ré-uploader).

```
[
  {
    "checksum": "Dr/LPU7TQVK4pw/8wXK8pEvW0c/...",
    "url": "https://upload.paytsoftware.com/...",
    "expires_at": "2026-06-24T12:00:00Z"   // ~4 h
  }
]
```

③ Téléverser le binaire (PUT)

Envoyez le contenu du fichier sur l'URL fournie. Payt vérifie la correspondance `byte_size` + `content_type` + `checksum`.

```
PUT https://upload.paytsoftware.com/...
Content-Type: application/pdf

<contenu binaire du PDF>
```

④ Référencer le fichier sur la facture

À la création/màj de la facture, renseignez `document` avec le **même** checksum :

```
"document": {
  "checksum": "Dr/LPU7TQVK4pw/8wXK8pEvW0c/...",
  "filename": "facture-2026-000142.pdf"
}
```

Pour détacher un document : `document.destroy: true` (combiné à `checksum` pour cibler un fichier précis). Le champ `ubl_document` fonctionne à l'identique pour un fichier UBL.

Étape 2 — Factures — `POST /v1/invoices`

Champs **requis** : `currency_code`, `invoice_number`, `invoice_date`, `due_date`, `amount_total`, `amount_open`, `book_amount_total`, `book_amount_open`. Le lien au débiteur se fait via `debtor_number` ou `debtor_identifier`.

```
{
  "administration_id": "81",
  "invoices": [
    {
      "invoice_number": "2026-000142",
      "invoice_identifier": "erp-inv-7741",
      "debtor_number": "CLI-1001",
      "currency_code": "EUR",
      "invoice_date": "2026-06-01",
      "due_date": "2026-07-01",
      "amount_total": "1200.00",
      "amount_open": "1200.00",
      "book_amount_total": "1200.00",
      "book_amount_open": "1200.00",
      "payment_condition": "30J",
      "payment_method": "bank_transfer",
      "sent_at": "2026-06-01T08:00:00Z",
      "invoice_lines": [
        {
          "description": "Prestation de conseil",
          "vat_rate_code": "STD20",
          "total_excl_tax_amount": "1000.00",
          "vat_amount": "200.00",
          "total_incl_tax_amount": "1200.00",
          "line_order": 1
        }
      ]
    }
  ]
}
```

Montants : devise facture vs devise administration

Champ	Signification
`amount_total` / `amount_open`	Total / restant dû, dans la **devise de la facture** (`currency_code`).
`book_amount_total` / `book_amount_open`	Mêmes valeurs dans la **devise comptable** de l'administration.

Le champ `sent_at` : à manier avec soin

Si `sent_at` est fourni, Payt considère que **la facture a déjà été envoyée par vos soins** et ne l'enverra pas au débiteur. Omettez-le pour laisser Payt envoyer la facture. Il n'est modifiable que tant que la facture n'a pas été envoyée.

Listes « tout ou rien »

Pour `invoice_lines` et `payments` : si vous en fournissez, vous devez fournir **la totalité** des éléments de la facture (Payt remplace l'ensemble). De même pour `labels` : les libellés absents de la requête sont retirés.

Étape 3 — Mises à jour ultérieures

Mettre à jour une facture / enregistrer un paiement — `PATCH /v1/invoices`

Seul `invoice_number` est requis ; ne transmettez que les champs modifiés. Pour solder une facture, ajustez l'`amount_open` et fournissez le paiement correspondant (par ordre chronologique, le plus ancien d'abord) :

```
{
  "administration_id": "81",
  "invoices": [
    {
      "invoice_number": "2026-000142",
      "amount_open": "0.00",
      "book_amount_open": "0.00",
      "payments": [
        {
          "amount": "1200.00",
          "origin_identifier": "bank-tx-99812",
          "payment_date": "2026-06-20",
          "transaction_type": "payment",
          "cost_type": "principal"
        }
      ]
    }
  ]
}
```

Si vous changez l'`amount_open` sans fournir de `payments`, Payt crée automatiquement un paiement pour l'écart. `transaction_type` ∈ { credit, payment, reversal, revaluation, settlement, write_off }. `cost_type` ∈ { principal, administration_costs, interest_costs, debt_collection_costs }.

Bloquer / reprendre le suivi d'une facture

Pour suspendre les relances (litige, geste commercial) puis les reprendre :

```
PATCH /v1/invoices/block     # administration_id + invoice_numbers[]
PATCH /v1/invoices/resume    # reprend uniquement ce qui a été bloqué via l'API
```

7. Lecture des réponses (succès partiels)

Les endpoints d'écriture acceptant plusieurs records, la réponse indique combien ont été traités et lesquels ont échoué. **Point essentiel à comprendre.**

Réponse type

```
{
  "success": true,
  "count": 9,
  "errors": {
    "not_found": [ "2026-000999" ]
  }
}
```

Champ	Signification
`success`	La requête a été traitée (≠ « tous les records ont réussi »).
`count`	Nombre de records traités avec succès (sans erreur).
`errors`	Records non traités, avec le motif. Objet **vide** = tout est passé.
`warnings`	Records traités partiellement, avec messages (présent seulement s'il y en a).

À implémenter absolument

Une réponse renvoie `success: true` **même si certains records ont échoué**. Ne vous fiez jamais au seul `success` : inspectez systématiquement `errors`. Un objet `errors` vide signifie que tout a été traité.

Erreurs fatales

Code	Cause	Conséquence
`422`	Requête inexploitable (corps mal formé, paramètre requis manquant, `fields` en chaîne au lieu de tableau).	**Aucun** record traité.
`401`	Non authentifié (token absent/expiré).	Requête rejetée.
`403`	Scope manquant ou accès non autorisé (le message nomme le scope requis).	Requête rejetée.

8. Limites, erreurs et pièges fréquents

Limites de débit

Limite	Valeur	Dépassement
API REST	10 requêtes / seconde / access token	Throttling — espacez ou regroupez en lots.

Comme les endpoints acceptent plusieurs records par appel, regroupez vos débiteurs et vos factures dans des requêtes en lot plutôt que de multiplier les appels unitaires : c'est plus rapide et cela ménage la limite de débit.

Pièges fréquents

À vérifier en priorité

- **Auto-import et API ne se cumulent pas** — si l'import par fichiers (auto-import) est activé sur l'administration, la création de factures via l'API est refusée. Réponse observée : `403` avec `{"code":"forbidden","message":"Not available with auto import turned on"}`. Choisissez **une seule** voie d'alimentation par administration.
  (Comportement non documenté dans la référence API, mais constaté en pratique.)
- **Montants en chaîne** — `"1200.00"` et non `1200.00`. Un nombre brut peut être rejeté.
- **Référentiel manquant** — un `vat_rate_code` ou un `payment_condition` absent de l'administration fait échouer la ligne / la facture. Posez les référentiels d'abord.
- `fields` **en tableau** sur les lectures, jamais en chaîne (sinon 422).
- **Restriction IP** — si vous restreignez par IP côté pare-feu, sachez que l'IP sortante peut varier ; en cas de 403 intermittents, vérifiez l'allowlist (page *IP addresses* de la doc).
- **Numéro de facture non global** — `invoice_number` est unique *par administration*, pas au-delà. Si vous gérez plusieurs administrations, conservez le couple administration + identifiant.

9. Checklist de mise en production

#	Vérification
1	Module API activé sur l'administration de production.
2	Intégration développée et validée d'abord sur l'environnement **démo**.
3	Scopes accordés au plus juste (lecture/écriture des ressources réellement utilisées).
4	Référentiels (taux de TVA, conditions de paiement) en place avant les factures.
5	Ordre d'écriture respecté : débiteurs (+ contacts) avant factures.
6	Lecture des réponses : `errors` inspecté systématiquement, pas seulement `success`.
7	Gestion des codes `401` / `403` / `422` et du throttling.
8	Idempotence vérifiée : rejouer une requête ne crée pas de doublon.
9	Une seule voie d'alimentation (API **ou** import fichiers) par administration.
10	Bascule des URLs `demo-api` → `api` et des jetons de production.

10. Annexes

Annexe A — Endpoints d'écriture utilisés

Ressource	Méthode & endpoint	Quand l'appeler	Clé d'upsert
Taux de TVA	`POST /v1/vat_rates`	Une fois (référentiel)	`code`
Conditions de paiement	`POST /v1/payment_conditions`	Une fois (référentiel)	`code`
Fichiers (upload)	`POST /v1/files` + PUT présigné	Avant la facture, si document PDF/UBL	`checksum` (SHA3-512 base64)
Débiteurs	`POST /v1/debtors`	À chaque création/màj client	`debtor_number` / `debtor_identifier`
Contacts	`POST /v1/contacts`	Avec les débiteurs (emails)	`contact_identifier`
Factures (upsert)	`POST /v1/invoices`	À chaque émission de facture	`invoice_number` / `invoice_identifier`
Factures (màj)	`PATCH /v1/invoices`	Paiements, corrections	`invoice_number`
Bloquer / reprendre	`PATCH /v1/invoices/block` · `/resume`	Litige, suspension	`invoice_numbers[]`

Annexe B — Champs requis par ressource

Ressource	Champs requis
Taux de TVA	`rate` · `name` (requis à la création) — `code` sert de clé d'identification
Conditions de paiement	`code` · `name`
Fichier (déclaration)	`byte_size` · `checksum` · `content_type`
Débiteur	`debtor_number` · `name` · `debtor_identifier`
Facture	`invoice_number` · `currency_code` · `invoice_date` · `due_date` · `amount_total` · `amount_open` · `book_amount_total` · `book_amount_open`
Ligne de facture	`description` · `vat_rate_code` · (un des deux montants HT/TTC)
Paiement	`amount` · `origin_identifier`

Annexe C — Références

Sujet	Page
Vue d'ensemble & prérequis	`docs.paytsoftware.com/api/overview`
Modèles & hiérarchie (Company, Administration…)	`docs.paytsoftware.com/models/overview`
Flux de création de factures	`docs.paytsoftware.com/models/invoice`
Upload de fichiers	`docs.paytsoftware.com/models/file`
Authentification (OAuth2)	`docs.paytsoftware.com/authentication/authorization`
Permissions / scopes	`docs.paytsoftware.com/authentication/permissions`
Lecture des réponses	`docs.paytsoftware.com/api/responses`
Référence des endpoints	`docs.paytsoftware.com/api-reference`
Environnement de test	`docs.paytsoftware.com/api/testing`