BY+ · CONSOLE NETBYUS · PLATEFORME INFLUENCER

# Connecter influencer-api à BY+ par des clés

Version du 23 octobre 2026. La plateforme des ambassadeurs et des partenaires ( `influencer-api` ) existe et reste indépendante. On ne l'intègre pas à BY+ : on refait, sur la nouvelle API ( `NewByPlus/core-api` ), la connexion par clés qui la relie aujourd'hui à l'ancienne core-api. Public : l'équipe technique NetByUs.

**En bref**

- **Deux plateformes séparées.** Chacune garde sa base, ses comptes et son métier. Aucune donnée n'est copiée : tout passe par des appels HTTP, protégés par des clés.
- **Trois liens**, comme dans l'ancienne core-api :
  1. BY+ → influencer : la console suit et administre influencer par un relais, avec un jeton de service ( `X-Service-Token` ) ;
  2. influencer → BY+ : influencer demande ses licences, lit les offres et transmet ses tickets, avec la clé de son application dans BY+ ;
  3. BY+ → influencer : les webhooks du support (tickets confiés aux influenceurs), signés par un secret.
- **Côté influencer, presque rien ne change** : l'adresse de BY+, les clés, la façon de signer, et quelques routes renommées.

## 1. Ce qui existe aujourd'hui avec l'ancienne core-api

| Sens | Clé | Usage |
|---|---|---|
| core-api → influencer | Jeton partagé `X-Service-Token` ( `INFLUENCER_SERVICE_TOKEN` chez core-api, `CORE_API_SERVICE_TOKEN` chez influencer) | Le module `app/v1/Influencer` relaie `/{client}/influencer/{chemin}` vers `influencer-api/api/v1/service/{chemin}` , après avoir vérifié la permission de l'utilisateur ( `influencer_ambassador_*` , `influencer_partner_*` , `influencer_application_*` ). Le back-office pilote ainsi ambassadeurs, partenaires, portefeuilles, dépôts, KYB, commissions, catalogue et rapports. |
| influencer → core-api | Couple de clés d'une application externe : `X-Api-Key` (pk) + `X-Api-Secret` (sk) | `LicenceApiService` et `HelpdeskService` : générer et gérer les licences ( `distributor/licenses` ), rattacher les contacts, lire le catalogue, déclarer les distributeurs d'une application, transférer un ticket. Le middleware `ExternalAppAuthenticate` vérifie le couple et la liste des routes permises. |
| core-api → influencer | Secret HMAC `X-Webhook-Signature` ( `CORE_API_WEBHOOK_SECRET` ) | Webhooks `/api/v1/webhook/helpdesk/ticket/created` et `…/replied` : un ticket d'une application est confié aux influenceurs qui la distribuent. |

## 2. Lien 1 — BY+ suit et administre influencer

On refait le relais de l'ancienne core-api dans la console : un petit domaine `Platform\Influencer` dans `packages/byplus-console` .

| Élément | Ce qu'il fait |
|---|---|
| Route `/api/v1/influencer/{chemin}` **à créer** | Toutes méthodes, réservée aux agents connectés à la console. Elle transmet la requête (paramètres, corps, fichiers) à `INFLUENCER_API_URL/{chemin}` avec l'en-tête `X-Service-Token`, puis rend la réponse telle quelle (statut et JSON). |
| Droits **à créer** | Le premier segment du chemin et la méthode donnent le droit demandé, comme avant : `influencer.ambassadors.view\|manage`, `influencer.partners.view\|manage`, `influencer.applications.view\|manage`, `influencer.finance.manage` (dépôts, portefeuilles, commissions), `influencer.settings.manage` (devises, pays, rôles, utilisateurs). Un chemin inconnu est refusé. |
| Journal | Chaque écriture (POST, PATCH, PUT, DELETE) est tracée dans le journal de la console : agent, chemin, statut de la réponse. |
| Écrans **à créer** | Un menu « Influenceurs » dans la console (core-app) : tableau de bord, ambassadeurs, partenaires, dépôts, commissions, KYB, catalogue, rapports. Les écrans appellent le relais ; ils ne parlent jamais directement à influencer. |

> **Une faille de l'ancienne core-api à ne pas reproduire**
>
> Son relais ne relie à une permission que `ambassadors`, `partners`, `applications`, `dashboard`, `currencies`, `roles`, `users` et `countries`. Pour tout autre chemin (`deposits`, `commissions`, `reports`, `partner-tiers`, `plans`…), aucun droit n'est vérifié : un utilisateur connecté au back-office peut valider un dépôt ou payer une commission. Dans BY+, chaque chemin a son droit et un chemin inconnu est refusé.

**Côté influencer : rien à changer.** Ses routes `/api/v1/service/*` et le middleware `auth.service` restent tels quels ; seul le jeton est régénéré (`php artisan service:token:generate`) et recopié dans la console.

> **Le jeton reste côté serveur**
>
> Le relais est le seul à connaître `X-Service-Token` : aucun navigateur ne le voit. On ajoute chez influencer une restriction à l'adresse IP du serveur de la console, et une limite de requêtes sur `/service/*`.

## 3. Lien 2 — influencer appelle BY+ avec ses clés

### Les clés

Dans BY+, une application externe est enregistrée dans la console (menu Applications) et reçoit une **instance** chez le client `netbyus`, puis une **clé**. C'est l'équivalent du couple pk/sk de l'ancienne core-api, en plus sûr : la clé privée ne voyage jamais, elle sert à *signer* chaque requête (Ed25519).

1. Console → Applications : créer l'application `influencer`, puis son instance pour `netbyus`.
2. Créer une clé « Serveur influencer-api » : la console montre la clé privée une seule fois. Elle va dans le `.env` d'influencer, avec l'identifiant de l'instance.
3. Révoquer une clé et en créer une autre se fait à tout moment ; deux clés peuvent coexister le temps de changer le `.env`.

| Ancienne core-api | BY+ |
|---|---|
| `X-Api-Key: pk_…` | `X-App-Key` : l'identifiant de l'instance |
| `X-Api-Secret: sk_…` | `X-App-Timestamp`, `X-App-Nonce`, `X-App-Signature` : la signature de la requête, faite avec la clé privée |

Une seule méthode change dans influencer : `LicenceApiService::http()` (et la même dans `HelpdeskService`) signe désormais la requête.

```
// influencer-api : app/Services/LicenceApiService.php
private function send(string $method, string $path, array $body = []): Response
{
    $json = $body === [] ? '' : json_encode($body, JSON_UNESCAPED_SLASHES |
JSON_UNESCAPED_UNICODE);
    $path = '/api/v1/store/influencer/'.ltrim($path, '/');
    $timestamp = time();
    $nonce = bin2hex(random_bytes(16));
    $message = implode("\n", [strtoupper($method), $path, $timestamp, $nonce, hash('sha256',
$json)]);
    $signature = base64_encode(sodium_crypto_sign_detached($message,
base64_decode(config('services.byplus.private_key'))));

    return Http::baseUrl(config('services.byplus.url'))->timeout(30)->acceptJson()
        ->withHeaders([
            'X-App-Key' => config('services.byplus.instance_id'),
            'X-App-Timestamp' => (string) $timestamp,
            'X-App-Nonce' => $nonce,
            'X-App-Signature' => $signature,
        ])
        ->withBody($json, 'application/json')
        ->send($method, $path);
}
```

Le corps signé doit être exactement celui envoyé ; la console refuse une requête à plus de 5 minutes de son heure ou un nonce déjà vu. `php artisan byplus:support:sign-request` signe une requête d'essai pour comparer.

### Les routes de BY+ pour influencer à créer

Un groupe `/api/v1/store/influencer/*`, réservé à l'instance de l'application `influencer` (requête signée). Il reprend une à une les routes qu'influencer appelle aujourd'hui, avec les mêmes réponses autant que possible :

| Aujourd'hui (ancienne core-api) | Demain (BY+) | Notes |
|---|---|---|
| `POST distributor/licenses` | `POST licenses` | Émet la licence d'une offre (domaine Licensing). Rend `license_key` et `id`, comme avant. Le titulaire et l'ambassadeur ou le partenaire sont rangés dans la licence. |
| `licenses/{id}/suspend`, `activate`, `revoke` | `licenses/{id}/suspend`, `unsuspend`, `revoke` | Seulement sur les licences émises par influencer. |
| `licenses/send/{id}/by/mail` | `licenses/{id}/send` | Envoi au client final. |
| `public-licences/licenses/validate` | `licenses/validate` | Vérifier une clé. |
| `applications/collection`, offres, types de licence | `offers` | Les offres que l'on peut vendre, pour relier les plans d'influencer. |
| `store/contact`, `contact/by-mail` | `contacts`, `contacts/by-email` | Un contact dans l'espace `netbyus`, comme avant ; l'identifiant change. |
| `distributor/influencers/assign` / `remove` | `distributors` (PUT / DELETE) | Les influenceurs qui distribuent une application : à qui confier ses tickets. |

| Aujourd'hui (ancienne core-api) | Demain (BY+) | Notes |
|---|---|---|
| Transfert de ticket, `helpdesk/sync-response` | `/api/v1/store/support/app/*` existe | Ouvrir un ticket ; répondre et changer le statut d'un ticket confié (délégué). |

L'instance ne voit que son application, ses offres et les licences qu'elle a émises. Chaque appel est journalisé à son nom.

## 4. Lien 3 — les webhooks de BY+ vers influencer

La console sait déjà envoyer des webhooks signés à une instance. Dans l'instance `influencer`, on déclare l'adresse d'influencer et les événements :

| Événement | Adresse chez influencer |
|---|---|
| `helpdesk_ticket_created` | `/api/v1/webhook/helpdesk/ticket/created` (existe) |
| `ticket_replied` | `/api/v1/webhook/helpdesk/ticket/replied` (existe) |
| `licence_updated` (facultatif) | À ajouter si influencer doit suivre un changement fait depuis la console (licence suspendue par un agent). |

> **Une seule différence à corriger chez influencer**
>
> BY+ envoie `X-Webhook-Signature: sha256=<empreinte>` ; l'ancienne core-api envoyait l'empreinte seule. `App\Services\WebhookService::verifySignature()` doit retirer le préfixe `sha256=` avant de comparer. Le secret est celui du webhook de l'instance, copié dans `CORE_API_WEBHOOK_SECRET`.

## 5. Les variables

| Où | Variables |
|---|---|
| BY+ (`NewByPlus/core-api`) | `INFLUENCER_API_URL` (ex. `https://influencer.byplus.io/api/v1/service`), `INFLUENCER_SERVICE_TOKEN` |
| influencer-api | `BY_PLUS_API_URL` (l'adresse de BY+, ex. `https://byplus.io`) ; à la place de `BY_PLUS_PUBLIC_KEY` / `BY_PLUS_SECRET_KEY` : `BY_PLUS_INSTANCE_ID` et `BY_PLUS_PRIVATE_KEY` ; `CORE_API_SERVICE_TOKEN` et `CORE_API_WEBHOOK_SECRET` gardent leur nom. |

> **À corriger tout de suite**
>
> `influencer-api/.env.example` contient un vrai jeton dans `LICENCE_API_KEY`. Il faut le remplacer par une valeur d'exemple et révoquer ce jeton.

## 6. Ce qu'il faut développer

| Lot | Contenu | Où |
|---|---|---|
| 1 | Relais `/api/v1/influencer/{chemin}`, ses droits, son journal ; variables `INFLUENCER_*` | BY+ (console) |
| 2 | Écrans « Influenceurs » de la console | core-app |

| Lot | Contenu | Où |
|---|---|---|
| 3 | Application `influencer`, son instance, sa clé et ses webhooks (paramétrage, sans code) | BY+ (console) |
| 4 | Routes `/api/v1/store/influencer/*` (licences, offres, contacts, distributeurs) | BY+ (console) |
| 5 | Signature des requêtes, routes renommées, préfixe `sha256=` des webhooks, variables | influencer-api |

Les lots 1 et 2 suffisent pour suivre influencer depuis BY+. Les lots 3 à 5 remplacent l'ancienne core-api pour les licences, les contacts et le support. Tant qu'ils ne sont pas faits, influencer peut continuer à demander ses licences à l'ancienne core-api : les deux liens sont indépendants.

### Tests

- BY+ : le relais (droit manquant → 403, chemin inconnu refusé, jeton ajouté, réponse rendue telle quelle, fichiers transmis) et les routes signées (signature valide, refusée, rejouée ; licence d'une autre instance refusée).
- influencer : la signature des requêtes (contre un faux serveur) et la vérification des webhooks avec le préfixe `sha256=`.

---

Sources : ancienne `core-api` (`app/v1/Influencer`, `app/Http/Middleware/ExternalAppAuthenticate.php`, `app/v1/License/routes/tenant.php`, `app/v1/Webhook/Jobs/DispatchWebhookJob.php`) ; `influencer-api` (`app/Services/LicenceApiService.php`, `app/v1/Helpdesk`, `app/v1/Admin/routes/service.php`, `app/Services/WebhookService.php`) ; `NewByPlus/core-api` (`packages/byplus-console/src/Applications`, `Support`, `core/Integrations`).