ChatGPT

Guide d'intégration de Touchpoint by InTouch (paiement Orange Money & Wave) sous Laravel

Pour accepter des paiements Mobile Money via **Touchpoint by InTouch** dans Laravel, il faut utiliser l'API REST fournie par InTouch Group. Voici les étapes clés, avec exemples de code PHP/Laravel. Les informations essentielles (URL API, clés, formats) doivent être confirmées sur le portail développeur officiel d'InTouch (developers.intouchgroup.net) après ouverture de compte.

1. Création du compte commerçant et obtention des clés API

1. **Inscription et KYC** – Créez un compte commerçant chez InTouch, fournissez les documents KYC requis (ex. NINEA, pièce d'identité, justificatif de domicile, etc.) [1] . Après validation (signature de contrat), InTouch active votre compte.
2. **Accès au portail développeur** – Une fois le compte validé, accédez au portail développeur d'InTouch (developers.intouchgroup.net). Vous y trouverez la documentation et une section "Clés API" (ou "Keys") pour générer vos identifiants d'API (clé ou token secret) [2] . Ces clés sont utilisées pour authentifier vos appels API. Stockez-les dans votre fichier `.env` (ex. `INTOUCH_API_KEY`), **uniquement côté serveur**.

   Exemple : supposons que le portail vous ait donné `INTOUCH_API_KEY=abc123…`. Ne l'exposez **jamais** au client. Conservez-le dans `.env` et accédez-y via `config/services.php` ou directement `env('INTOUCH_API_KEY')` en Laravel.

2. URL et structure de l'API Touchpoint

L'API Touchpoint d'InTouch est une API REST. InTouch fournira une *base URL* (par exemple `https://api.touchpoint.intouchgroup.net/v1`) pour les environnements test et production. Tous les endpoints sont relatifs à cette base. Par exemple, un endpoint générique pourrait être :

```
POST /payment/request-payment
```

Comme illustration, l'API PayTech (similaire) utilise `POST /payment/request-payment` pour initier un paiement [3] . Dans le cas d'InTouch, ce type d'endpoint accepterait un JSON avec les détails de la transaction (montant, référence, devise) et la méthode de paiement ciblée. L'API retourne typiquement un identifiant de transaction et l'URL de redirection pour finaliser le paiement.

> **Référence** : InTouch propose une documentation en ligne de son API (portail développeur) [4] . Les appels se font en HTTPS (TLS) uniquement.

3. Initiation d’un paiement Orange Money

Sur votre serveur Laravel, préparez une requête HTTP (POST) vers l’API d’InTouch pour lancer le paiement. Exemple avec la façade `Http` de Laravel :

```
use Illuminate\Support\Facades\Http;

public function payOrangeMoney(Request $request)
{
    // Données de la transaction
    $amount = 15000; // en XOF
    $orderId = 'CMD-' . time();

    // Préparation du payload
    $payload = [
        'item_name'    => 'Achat de service',      // description
        'item_price'   => $amount,                 // montant
        'currency'     => 'XOF',
        'ref_command'  => $orderId,                // référence unique
        'command_name' => 'Facture n°'.$orderId,
        'success_url'  => route('payment.success'), // URL de retour en cas de
succès
        'cancel_url'   => route('payment.cancel'),  // URL si le client annule
        'ipn_url'      => route('intouch.webhook'), // URL de callback pour
notification
        'target_payment'=> 'Orange Money',         // méthode ciblée
        // autres champs éventuels...
    ];

    // Appel API à Touchpoint
    $response = Http::withToken(config('services.intouch.api_key'))
                    ->post('https://api.touchpoint.intouchgroup.net/v1/payment/
request-payment', $payload);

    if (!$response->successful()) {
        abort(500, "Erreur Intouch: " . $response->body());
    }

    $data = $response->json();
    // On attend un champ 'redirect_url' pour finaliser le paiement
    if (isset($data['redirect_url'])) {
        // Rediriger l'utilisateur vers la page de paiement InTouch/Orange Money
        return redirect()->away($data['redirect_url']);
    }

    // Gérer les cas d'erreur ou réponse inattendue...
```

```
    return abort(500, "Réponse API inattendue.");
}
```

Dans cet exemple : - **Méthode :** appel côté serveur (backend) via `Http::post`. La clé API est transmise dans l'entête `Authorization: Bearer <clé>` [5] .
- **Payload :** contient montant, monnaie, référence client, URLs de retour et de callback, et le paramètre `"target_payment": "Orange Money"` pour spécifier le paiement Orange Money [6] .
- **Réponse :** l'API retourne typiquement un objet JSON. Par exemple, un champ `"redirect_url"` pour envoyer l'utilisateur vers l'interface de paiement [7] . On utilise `redirect()->away($url)` en Laravel pour quitter le site et finaliser le paiement sur le site d'Orange/Touchpoint.

> **À noter :** Orange Money Web Payment demandera à l'utilisateur de générer un code OTP via le service USSD Orange Money pour valider le paiement [8] . Mais cette logique est gérée par la plateforme de paiement après la redirection.

4. Initiation d'un paiement Wave

La procédure est similaire pour Wave. Avec Touchpoint, on change la méthode ciblée :

```
$payload['target_payment'] = 'Wave'; // au lieu de Orange Money
$response = Http::withToken(config('services.intouch.api_key'))
                ->post('https://api.touchpoint.intouchgroup.net/v1/payment/
request-payment', $payload);
// Suite analogue : récupérer et suivre $response['redirect_url']
```

On peut également indiquer une `ipn_url` pour être notifié du résultat. En général, si une seule méthode est spécifiée, Touchpoint peut permettre de pré-remplir et d'automatiser l'envoi (auto-submit) de l'utilisateur vers la page Wave [9] .

> **Remarque:** InTouch intègre probablement en interne l'API Wave. Par ailleurs, l'API officielle Wave permettrait, en l'absence de Touchpoint, de créer directement une session de paiement via `POST https://api.wave.com/v1/checkout/sessions` avec `{amount, currency, success_url, error_url}` [10] . Mais avec Touchpoint, on délègue ces appels à InTouch.

5. Gestion du callback (webhook)

Pour suivre le statut du paiement, exposez une route dans Laravel qui reçoit la notification d'InTouch (souvent via `ipn_url` ou `callback_url`). Par exemple :

```
// routes/web.php
Route::post('/webhook/intouch', [PaymentController::class,
'handleIntouchWebhook'])->name('intouch.webhook');
```

Dans le contrôleur :

```
public function handleIntouchWebhook(Request $request)
{
    // Exemple : récupérer les données JSON envoyées par InTouch
    $payload = $request->all();
    // Exemple : { "ref_command": "...", "status": "SUCCESS", "transaction_id":
"...", ... }

    // **Sécurité :** vérifier que la requête vient bien d'InTouch.
    // Par exemple, comparer une signature HMAC dans l'entête à un secret connu
[11].
    // (Ici, on suppose une stratégie "Shared Secret" : InTouch fournirait un
secret webhooks à stocker.)
    $signature = $request->header('Authorization');
    if ($signature !== 'Bearer ' . config('services.intouch.webhook_secret')) {
        return response('Unauthorized', 401);
    }

    // Mettre à jour la commande selon le statut
    $ref = $payload['ref_command'] ?? null;
    $status = $payload['status'] ?? null;
    if ($ref && $status) {
        // Rechercher la commande en base et marquer payée le cas échéant
        $order = Order::where('reference', $ref)->first();
        if ($order) {
            $order->status = ($status === 'SUCCESS') ? 'paid' : 'failed';
            $order->save();
        }
    }

    return response('OK', 200);
}
```

Quelques points clés :
- **Vérification de source :** Il est crucial de vérifier l'origine du callback. Par analogie avec l'API Wave, InTouch peut utiliser un header d'authentification (secret partagé ou signature) [11]. Vous devez comparer cette valeur avec la vôtre pour vous assurer que la requête est authentique.
- **Réponse HTTP :** Retourner un code 200 (`response('OK')`) pour indiquer la bonne réception avant de lancer votre logique métier.
- **Idempotence :** Les webhooks peuvent être renvoyés en cas d'erreur. Votre gestion doit être idempotente.
- **Modèle MVC :** Dans Laravel, ce code tourne côté serveur (backend). On ne touche pas au client.

6. Bonnes pratiques de sécurité

- **HTTPS :** Toutes les communications doivent se faire via HTTPS. Ni le client ni vous ne devez envoyer de données sensibles sur HTTP non sécurisé [5].
- **Clés secrètes** – Gardez votre clé API strictement côté serveur [12]. Ne l'incluez jamais dans du code JavaScript ou dans des pages publiques. Chargez-la depuis `.env`.
- **Validation des signatures :** Vérifiez systématiquement les signatures ou tokens inclus dans les callbacks pour prévenir toute requête frauduleuse [11]. Par exemple, Wave envoie un header `Authorization: Bearer <secret>`, qui doit être comparé au secret enregistré [11]. InTouch pourrait proposer un mécanisme similaire.
- **Éviter le logging sensible :** Ne loguez ni les clés, ni les en-têtes de webhook (certains services soulignent qu'un header exposé dans les logs peut compromettre le secret) [11].
- **Permissions minimales :** Si InTouch offre plusieurs clés (production, sandbox, etc.), utilisez celle qui a le minimum de droits nécessaires et révoquez les clés obsolètes.
- **Time-out et retry :** Gérez les erreurs réseau : mettez en place des retry/timeout sur vos appels API. Ne jamais bloquer l'utilisateur si l'API réagit lentement.

7. Intégration dans Laravel (packages, middleware…)

- **Middleware/Routes** : Aucune extension spécifique n'est imposée. Créez vos propres contrôleurs et routes comme ci-dessus. Vous pouvez utiliser un middleware `VerifyCsrfToken` désactivé pour l'URL de callback (`except` dans `VerifyCsrfToken`) afin d'accepter les POST externes.
- **Code d'exemple** : Utilisez `Illuminate\Support\Facades\Http` ou `GuzzleHttp\Client` pour les appels API. Pour le callback, créez une route `POST` simple.
- **Fichiers de config** : Ajouter les identifiants dans `config/services.php` et `.env` (ex. `INTOUCH_API_KEY`, `INTOUCH_WEBHOOK_SECRET`). Exemple :

```
// config/services.php
'intouch' => [
    'api_key' => env('INTOUCH_API_KEY'),
    'webhook_secret' => env('INTOUCH_WEBHOOK_SECRET'),
    'base_uri' => env('INTOUCH_BASE_URI'), // ex. https://
api.touchpoint.intouchgroup.net/v1
],
```

- **Packages tiers** : Il n'existe pas (à ma connaissance) de package Laravel officiel pour Touchpoint. Cependant, pour Orange Money pur (sans Intouch) on trouve des packages Laravel (ex. *ibracilinks/orange-money*) qui illustrent la structure (configuration, service provider) [13] [14]. Vous pouvez vous en inspirer pour organiser votre code.
- **Front-end** : La plupart du flux se déroule côté serveur. En front-end, vous pouvez simplement déclencher les actions serveur (clic sur bouton "Payer avec Orange Money") et rediriger selon la réponse. Aucune intégration JavaScript particulière n'est nécessaire puisque la page de paiement est externe.
- **Tests** : InTouch devrait fournir un environnement sandbox. Testez d'abord en sandbox. Vérifiez les réponses de l'API et simulez les webhooks (ils proposent souvent un outil de test ou des exemples de payload).

Sources officielles

Ce guide s'appuie sur la documentation InTouch et fournisseurs associés. Par exemple, InTouch confirme proposer des APIs REST « accessibles et faciles à intégrer » sur son portail développeur [4]. Les exemples de structure de requêtes se basent sur la documentation PayTech (Orange Money / Wave) [3] [6] qui suit les mêmes principes. Les bonnes pratiques de sécurité (utilisation du Bearer token, HTTPS, validation des webhooks) sont soulignées par les docs officielles Wave [12] [5] et Orange. Respectez toujours les recommandations d'InTouch et adaptez les exemples ci-dessus aux spécifications exactes fournies par votre contrat Intouch.

---

[1] [3] [6] [7] [9] PayTech API - Documentation
https://docs.intech.sn/doc_paytech.php

[2] API - InTouch
https://help.intouch.cloud/article/c2r366oa2k-api

[4] Documentation API | Simplifiez vos paiements et services digitaux
https://www.intouchgroup.net/solutions/api-intouch

[5] [10] [12] Wave Business APIs
https://docs.wave.com/business

[8] Orange Money Web Payment / M Payment (1.0) API – Overview – Orange Developer
https://developer.orange.com/apis/om-webpay

[11] Wave Payout API
https://docs.wave.com/webhook

[13] [14] GitHub - Ibracilinks/OrangeMoney: A laravel package for Orange Money Web Payment API.
https://github.com/Ibracilinks/OrangeMoney