REALNODE PROTOCOL

Guide d'Intégration B2B — React & Next.js (NPM)

Manuel technique de déploiement pour environnements JavaScript modernes.

Architecture Composants (React)
Ce document détaille l'intégration du protocole RealNode via notre package officiel NPM. Conçu pour React et Next.js, ce package vous fournit des composants prêts à l'emploi (boutons, badges, gestionnaire d'appareils) pour sécuriser votre tunnel de vente sans avoir à créer d'interfaces complexes.

1 Étape 0 : Création du compte et récupération des clés API

La toute première étape consiste à créer votre compte sur le portail RealNode et à récupérer vos clés d'accès. Ces clés identifient votre site auprès de notre infrastructure.

1. **Inscription :** Rendez-vous sur le portail administrateur (`https://app.realnode.emkaylabs.tech/signup`) pour créer votre compte entreprise. Le formulaire demande votre nom d'entreprise, votre adresse e-mail professionnelle et un mot de passe administrateur.
2. **Récupération des clés :** Le portail génère une paire de clés unique pour votre site :
   — **Clé Publique** (format : `pk_live_...`) : à utiliser dans votre code frontend (React). Elle peut figurer dans votre code côté navigateur sans risque.
   — **Clé Secrète** (format : `sk_live_...`) : à utiliser exclusivement sur votre serveur backend pour valider les transactions. Ne la partagez jamais et ne l'incluez jamais dans votre code frontend.

2 Étape 1 : Installation du package NPM

Ouvrez le terminal à la racine de votre projet React ou Next.js et exécutez la commande suivante :

```
npm install @emkaylabs/realnode-sdk
```

Que fait cette commande ? Elle télécharge et installe dans votre projet le package `@emkaylabs/realnode-s`
Ce package contient :

— Le client HTTP qui communique avec l'API RealNode (`/verify`, `/consume`, `/check-session`, etc.).

— Les composants React prêts à l'emploi (`RealNodeProvider`, `RealNodeCheckoutButton`, `RealNodeDeviceMa` `RealNodeQuotaBadge`).

— Les utilitaires de gestion des sessions et du stockage local (device token).

Une fois installé, le package sera disponible dans votre dossier `node_modules` et référencé dans votre `package.json`. Vous n'aurez plus besoin de réexécuter cette commande sauf pour mettre à jour le SDK.

3 Étape 2 : Configuration globale — le Provider

Le composant `RealNodeProvider` est le point central de l'intégration. Il initialise le moteur de sécurité et le rend accessible à tous les composants enfants de votre application. Il doit être placé **une seule fois**, au niveau le plus haut possible de votre arborescence.

Où placer ce code ? Dans votre fichier racine : `App.jsx` pour un projet Create React App, `_app.jsx` ou `layout.jsx` pour Next.js.

```
import { RealNodeProvider } from '@emkaylabs/realnode-sdk/react';

export default function App({ children }) {
  return (
    <RealNodeProvider apiKey="pk_live_VOTRE-CLE-PUBLIQUE">
      {children}
    </RealNodeProvider>
  );
}
```

Ce que fait le Provider au démarrage :

1. Il appelle automatiquement `GET /sdk/config?api_key=pk_live_...` sur nos serveurs.
2. Nos serveurs retournent la configuration correspondant à votre abonnement actif : le niveau de protection (`tier`) et les fonctionnalités activées (`features`).
3. En fonction de la réponse, le SDK active les bons modules sans que vous ayez à intervenir :
   - **Insight :** analyse comportementale passive uniquement, aucune interface visible pour l'utilisateur.
   - **Sentinel :** la matrice de capteurs comportementaux est activée en arrière-plan.
   - **Vault :** la matrice comportementale et la modal biométrique (FaceID, TouchID, Windows Hello) sont activées. La modal s'injecte automatiquement dans le DOM.
4. Il tente de restaurer silencieusement une session existante (si l'utilisateur a déjà vérifié son identité lors d'une visite précédente sur le même événement), afin de ne pas lui redemander sa biométrie inutilement.

Confidentialité de la clé publique
La clé publique (`pk_live_...`) peut figurer directement dans le code React, car elle ne donne accès à aucune action d'administration. Elle identifie simplement votre site auprès de notre API de vérification. La clé secrète (`sk_live_...`), en revanche, ne doit jamais quitter votre serveur backend.

4 Étape 3 : Protection du bouton d'achat

C'est la seule modification à apporter à vos pages de paiement. Vous remplacez votre bouton d'achat standard par le composant `RealNodeCheckoutButton`.

```
import { RealNodeCheckoutButton } from '@emkaylabs/realnode-sdk/react';

export default function PagePaiement() {

  const handleSuccess = (proof) => {
    // proof contient : { idh, device_hash, device_token, quantity, status
        }
    // Envoyez ces données à votre backend pour valider la commande.
    fetch('/api/votre-checkout', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
```

```
      body: JSON.stringify(proof)
    });
  };

  const handleDenied = () => {
    alert("Accès refusé : comportement suspect détecté.");
  };

  return (
    <RealNodeCheckoutButton
      quantity={2}
      onSuccess={handleSuccess}
      onDenied={handleDenied}
      className="votre-classe-css-existante"
    >
      Confirmer la commande
    </RealNodeCheckoutButton>
  );
}
```

Ce qui se passe lorsque l'utilisateur clique :

1. Le clic est intercepté. L'achat ne s'effectue pas immédiatement.
2. Selon votre abonnement détecté automatiquement par le Provider :
   - **Insight :** une vérification silencieuse est effectuée en arrière-plan. L'utilisateur ne voit rien.
   - **Sentinel :** le score comportemental est calculé et soumis à l'API. L'utilisateur ne voit rien.
   - **Vault :** la modal biométrique s'affiche. L'utilisateur vérifie son identité avec FaceID, TouchID ou Windows Hello. L'opération prend moins d'une seconde.
3. Si la vérification réussit, `onSuccess` est appelé avec une preuve cryptographique (`proof`).
4. Si la vérification échoue ou si un bot est détecté, `onDenied` est appelé et la transaction est bloquée.

Important : la biométrie est demandée une seule fois par session.
Le SDK conserve la preuve de vérification en mémoire pour toute la durée de la session. Si l'utilisateur a déjà effectué une vérification biométrique (par exemple lors d'un premier achat), elle ne lui sera pas redemandée pour ses achats suivants sur la même page, sauf si sa session a expiré.

5 Étape 4 : Composants d'interface utilisateur (recommandé)

RealNode fournit des composants prêts à l'emploi qui s'intègrent à votre design existant. Ils ne nécessitent aucun travail de maquette ou de CSS.

5.1 4.1. Affichage des quotas en temps réel

Applicable à tous les plans. Affiche le nombre de billets restants pour l'utilisateur courant. La valeur est synchronisée automatiquement via une connexion temps réel (SSE) avec nos serveurs.

```
import { RealNodeQuotaBadge } from '@emkaylabs/realnode-sdk/react';

// Placez ce composant a cote du bouton d'achat.
// Rendu automatique : "Billets restants : 3 / 5"
<RealNodeQuotaBadge label="Billets restants" />
```

5.2 4.2. Espace client — gestion des appareils

Ce composant génère automatiquement une interface complète dans la page "Mon Compte" de vos utilisateurs, leur permettant de :

- Consulter la liste des appareils (téléphones, ordinateurs) liés à leur identité matérielle.
- Bloquer un appareil en cas de vol.
- Libérer un appareil en cas de revente.
- Transférer leur compte à un autre utilisateur.
- Restaurer leur accès via un code de récupération (Master Code) fourni par l'administrateur de votre site.

```
import { RealNodeDeviceManager } from '@emkaylabs/realnode-sdk/react';

export default function MonCompte() {
  return (
    <div>
      <h2>Parametres de Securite</h2>
      {/* Genere automatiquement toute l'interface de gestion. */}
      {/* Aucun HTML ni CSS supplementaire requis.            */}
      <RealNodeDeviceManager apiKey="pk_live_VOTRE-CLE-PUBLIQUE" />
    </div>
  );
}
```

6 Étape 5 : Validation finale côté serveur (backend)

La preuve générée côté navigateur (`proof`) doit être validée par votre serveur avant de procéder au paiement. C'est une étape de sécurité obligatoire : votre serveur contacte l'API RealNode avec votre **Clé Secrète** pour confirmer que la vérification est authentique et que le quota n'est pas dépassé.

```
// Exemple : backend Node.js / Express
app.post('/api/votre-checkout', async (req, res) => {
  const proof = req.body;
  // proof = { idh, device_hash, device_token, quantity, status }

  const response = await fetch('https://api.emkaylabs.tech/consume', {
    method: 'POST',
    headers: {
```

```
      'Content-Type': 'application/json',
      'X-RealNode-API-Key':    'pk_live_VOTRE-CLE-PUBLIQUE',
      'X-RealNode-Secret-Key': 'sk_live_VOTRE-CLE-SECRETE'
    },
    body: JSON.stringify({
      quantity:     proof.quantity,
      idh:          proof.idh,
      device_hash:  proof.device_hash,
      device_token: proof.device_token,
      client_id:    'votre-site-web',
      // event_id: 'CONCERT-2024-PARIS' // Optionnel : pour des quotas par
          evenement
    })
  });

  if (response.ok) {
    // RealNode a valide la transaction. Procedez au paiement.
    await processPayment();
    res.status(200).json({ success: true });
  } else {
    // Quota depasse ou verification invalide.
    res.status(403).json({ error: "Transaction refusee par RealNode." });
  }
});
```

Réponses de l'API `/consume` :

Code HTTP	Signification	Action recommandée
200	Transaction validée, quota déduit	Procéder au paiement
403	Vérification invalide ou bot détecté	Refuser la commande
429	Quota épuisé pour cet utilisateur	Afficher un message d'attente
401	Clé API incorrecte	Vérifier vos clés dans le tableau de bord

Support B2B dédié
En tant que client RealNode, vous bénéficiez d'un accompagnement technique prioritaire. Pour toute question sur votre intégration, contactez notre équipe à **security@emkaylabs.tech**. Nous répondons sous 24 heures ouvrables.

7 Annexe : Fonctionnalités Avancées

7.1 Gestion des Quotas par Événement (Abonnement Vault)

Pour les ventes à accès contrôlé (billets en édition limitée, drops exclusifs, etc.), vous pouvez transmettre l'identifiant de votre événement interne lors de l'appel `/consume` (voir Étape 5).

Comment ça fonctionne ?

— Vous n'avez rien à configurer au préalable dans le tableau de bord RealNode.

— Utilisez simplement vos propres identifiants (ex : `CONCERT-2024-PARIS`).

— RealNode l'utilisera pour isoler et comptabiliser les quotas de chaque utilisateur spécifiquement pour cet événement.