REALNODE PROTOKOLL

B2B Integrationshandbuch — React & Next.js (NPM)

Technisches Bereitstellungshandbuch für moderne JavaScript-Umgebungen.

Komponentenarchitektur (React)
Dieses Dokument detailliert die Integration des RealNode-Protokolls über unser offizielles NPM-Paket. Speziell für React und Next.js entwickelt, bietet dieses Paket sofort einsatzbereite Komponenten (Checkout-Buttons, Quoten-Badges, Geräte-Manager), um Ihren Transaktionstrichter zu sichern, ohne dass komplexe Schnittstellen entworfen werden müssen.

1 Schritt 0: Kontoerstellung und Abruf der API-Schlüssel

Der allererste Schritt besteht darin, ein Konto im RealNode-Portal zu erstellen und Ihre Zugriffsschlüssel abzurufen. Diese Schlüssel identifizieren Ihre Anwendung sicher gegenüber unserer Infrastruktur.

1. **Registrierung:** Besuchen Sie das Admin-Portal (`https://app.realnode.emkaylabs.tech/signup`), um Ihr Unternehmenskonto zu erstellen. Sie müssen Ihren Firmennamen, eine professionelle E-Mail-Adresse und ein Administrator-Passwort angeben.
2. **Schlüssel abrufen:** Das Portal generiert ein einzigartiges Schlüsselpaar für Ihre Umgebung:
   - **Public Key** (Format: `pk_live_...`): Zur Verwendung in Ihrem React-Frontend-Code. Es ist absolut sicher, diesen Schlüssel im Browser preiszugeben.
   - **Secret Key** (Format: `sk_live_...`): Ausschließlich auf Ihrem Backend-Server zur kryptografischen Validierung von Transaktionen zu verwenden. Teilen Sie diesen Schlüssel niemals und binden Sie ihn nicht in Ihren Frontend-Code ein.

2 Schritt 1: Installation des NPM-Pakets

Öffnen Sie Ihr Terminal im Stammverzeichnis Ihres React- oder Next.js-Projekts und führen Sie den folgenden Befehl aus:

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

Was bewirkt dieser Befehl? Er lädt das Paket `@emkaylabs/realnode-sdk` herunter und installiert es in Ihrem Projekt. Dieses Paket enthält:

- Den Core-HTTP-Client, der mit der RealNode-API kommuniziert (`/verify`, `/consume`, `/check-session`, usw.).
- Produktionsreife React-Komponenten (`RealNodeProvider`, `RealNodeCheckoutButton`, `RealNodeDeviceMa` `RealNodeQuotaBadge`).
- Dienstprogramme für das Sitzungsmanagement und die sichere lokale Speicherung (Lebenszyklus des Geräte-Tokens).

Nach der Installation ist das Paket in Ihrem `node_modules`-Verzeichnis verfügbar und in Ihrer `package.json` referenziert.

3 Schritt 2: Globale Konfiguration — Der Provider

Die `RealNodeProvider`-Komponente fungiert als zentraler Knotenpunkt. Sie initialisiert die Sicherheits-Engine und stellt den SDK-Kontext für alle verschachtelten Komponenten bereit. Sie muss **genau einmal** gemountet werden, so hoch wie möglich in Ihrem Komponentenbaum.

Wo ist dieser Code zu platzieren? In Ihrer Root-Einstiegsdatei: `App.jsx` für Create React App oder `_app.jsx` / `layout.jsx` für Next.js.

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

export default function App({ children }) {
  return (
    <RealNodeProvider apiKey="pk_live_IHR_PUBLIC_KEY">
      {children}
    </RealNodeProvider>
  );
}
```

Interner Initialisierungsablauf:

1. Der Provider sendet automatisch eine `GET /sdk/config?api_key=pk_live_...`-Anfrage an unsere Edge-Server.
2. Die Server geben die aktive Konfiguration zurück, die Ihrem Abonnementplan zugeordnet ist: die Schutzstufe und die aktivierten Funktionen.
3. Basierend auf der Nutzlast (Payload) bindet das SDK dynamisch die entsprechenden Module ein, ohne dass Codeänderungen Ihrerseits erforderlich sind:
   - **Insight:** Nur passive Verhaltensanalyse. Keine sichtbare UI-Interferenz.
   - **Sentinel:** Die vollständige kinetische Verhaltensmatrix wird im Hintergrund aktiviert.
   - **Vault:** Die kinetische Matrix und das biometrische Modal (FaceID, TouchID, Windows Hello) werden aktiviert. Das Modal wird automatisch in das DOM eingefügt.
4. Das SDK versucht im Hintergrund, eine bestehende Sitzung wiederherzustellen (falls der Benutzer während desselben Ereignisfensters bereits eine Verifizierung abgeschlossen hat), um redundante biometrische Aufforderungen zu vermeiden.

Sicherheit des Public Key
Ihr öffentlicher Schlüssel (`pk_live_...`) kann sicher in Ihrem öffentlichen React-Bundle verbleiben. Er gewährt keinerlei Zugriff auf administrative Aktionen und dient ausschließlich dazu, Ihr Eigentum gegenüber unserer Verifizierungs-API zu identifizieren. Im Gegensatz dazu muss Ihr geheimer Schlüssel (`sk_live_...`) vollständig auf Ihrer Backend-Infrastruktur isoliert bleiben.

4 Schritt 3: Absicherung des Checkout-Buttons

Dies ist die einzige notwendige Änderung innerhalb Ihres Checkout-Ablaufs. Ersetzen Sie einfach Ihren Standard-HTML-Kaufbutton durch die `RealNodeCheckoutButton`-Komponente.

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

export default function CheckoutPage() {

  const handleSuccess = (proof) => {
    // proof enthaelt: { idh, device_hash, device_token, quantity, status
       }
```

```
    // UEbertragen Sie diesen Payload an Ihr Backend, um die Bestellung
       abzuschliessen.
    fetch('/api/ihr-checkout-endpunkt', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(proof)
    });
  };

  const handleDenied = () => {
    alert("Zugriff verweigert: Verd chtige Automatisierung erkannt.");
  };

  return (
    <RealNodeCheckoutButton
      quantity={2}
      onSuccess={handleSuccess}
      onDenied={handleDenied}
      className="ihre-vorhandene-css-klasse"
    >
      Bestellung Bestaetigen
    </RealNodeCheckoutButton>
  );
}
```

Ausführungsablauf nach Benutzerklick:

1. Das Klickereignis wird abgefangen. Die Transaktion wird vorübergehend angehalten.
2. Abhängig von dem durch den Provider erkannten aktiven Plan:
   - **Insight:** Eine stille Hintergrundüberprüfung wird ausgeführt. Der Benutzer bemerkt nichts.
   - **Sentinel:** Der Verhaltens-Score wird berechnet und verifiziert. Der Benutzer bemerkt nichts.
   - **Vault:** Das biometrische Modal rückt in den Fokus. Der Benutzer verifiziert seine Identität via FaceID, TouchID oder Passkey. Die Hardware-Operation wird in unter 1 Sekunde abgeschlossen.
3. Nach erfolgreicher Attestierung wird `onSuccess` ausgelöst und der kryptografische Beweis (`proof`) injiziert.
4. Wenn die Attestierung fehlschlägt oder eine Bot-Signatur erkannt wird, wird `onDenied` ausgelöst und die Transaktionssequenz abgebrochen.

Wichtig: Biometrie wird nur einmal pro Sitzung abgefragt.
Das SDK speichert den Attestierungsbeweis im Speicher. Wenn ein Benutzer eine biometrische Herausforderung erfolgreich bestanden hat (z. B. während eines vorherigen Kaufs), wird er bei nachfolgenden Transaktionen auf derselben Seite nicht erneut aufgefordert, vorausgesetzt, die Sitzung bleibt gültig.

5 Schritt 4: Benutzeroberflächen-Komponenten (Empfohlen)

RealNode bietet vorgefertigte, anpassbare Komponenten, die sich nahtlos in Ihre bestehende Benutzeroberfläche einfügen und keinerlei Wireframing oder CSS-Boilerplate erfordern.

5.1 4.1. Echtzeit-Quotenanzeige

Verfügbar in allen Plänen. Dieses Badge zeigt das verbleibende Ticketinventar an, das für die aktuelle Benutzersitzung verfügbar ist. Der numerische Wert wird automatisch über eine persistente Server-Sent Events (SSE)-Verbindung mit unseren Edge-Servern synchronisiert.

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

// Platzieren Sie diese Komponente neben dem Checkout-Button.
// Rendert automatisch: "Verbleibende Tickets: 3 / 5"
<RealNodeQuotaBadge label="Verbleibende Tickets" />
```

5.2 4.2. Benutzer-Dashboard — Geräteverwaltung

Diese Komponente fügt ein vollständiges, interaktives Dashboard in den Bereich "Mein KontoÏhrer Benutzer ein und ermöglicht ihnen Folgendes:

- Prüfung der Liste der autorisierten Hardwaregeräte (Smartphones, Laptops), die mit ihrer Identität verknüpft sind.
- Fernwiderruf des Zugriffs für ein verlorenes oder gestohlenes Gerät.
- Trennung von Hardware-Bindungen vor dem Weiterverkauf eines Geräts.
- Sichere Übertragung des Kontozugriffs an einen anderen Benutzer.
- Wiederherstellung des Zugriffs mit einem von Ihrem Administratorenteam bereitgestellten Notfall-Master-Code.

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

export default function MyAccount() {
  return (
    <div>
      <h2>Sicherheitseinstellungen</h2>
      {/* Generiert automatisch die gesamte Verwaltungsschnittstelle. */}
      {/* Keinerlei zusaetzliche HTML- oder CSS-Logik erforderlich.    */}
      <RealNodeDeviceManager apiKey="pk_live_IHR_PUBLIC_KEY" />
    </div>
  );
}
```

6 Schritt 5: Abschließende Server-seitige Validierung (Backend)

Der vom Browser ausgegebene kryptografische Beweis (`proof`) muss von Ihrer Backend-Infrastruktur validiert werden, bevor die Bestellung ausgeführt wird. Dies ist ein obligatorischer Sicherheitskontrollpunkt: Ihr Server führt einen authentifizierten Aufruf an die RealNode-API mit Ihrem **Secret Key** durch, um zu garantieren, dass der Beweis nicht manipuliert wurde und die Inventarquote weiterhin gültig ist.

```
// Beispiel: Node.js / Express Backend
app.post('/api/ihr-checkout-endpunkt', 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_IHR_PUBLIC_KEY',
      'X-RealNode-Secret-Key': 'sk_live_IHR_SECRET_KEY'
    },
    body: JSON.stringify({
      quantity:     proof.quantity,
      idh:          proof.idh,
      device_hash:  proof.device_hash,
      device_token: proof.device_token,
      client_id:    'ihre-website-id',
      // event_id: 'KONZERT-2024-BERLIN' // Optional: fuer
         ereignisspezifische Quoten
    })
  });

  if (response.ok) {
    // RealNode hat die Attestierung erfolgreich validiert.
    await processPayment();
    res.status(200).json({ success: true });
  } else {
    // Quote ueberschritten oder ungueltige kryptografische Signatur.
    res.status(403).json({ error: "Transaktion von RealNode Engine
       abgelehnt." });
  }
});
```

Standard `/consume` API-Antworten:

HTTP-Code	Bedeutung	Empfohlene Aktion
200	Transaktion validiert, Quote abgezogen	Bestellung abschließen
403	Ungültige Signatur oder Bot erkannt	Transaktion ablehnen
429	Quote für diese Benutzer-IDH erschöpft	Cooldown-Nachricht anzeigen
401	Ungültige API-Schlüssel-Anmeldeinformationen	Überprüfen Sie Ihre Schlüssel im Dashboard

Dedizierter B2B-Support
Als RealNode-Unternehmenskunde erhalten Sie priorisierten Integrationssupport. Wenden Sie sich für architektonische Anfragen an unser Engineering-Team unter **security@emkaylabs.tech**. Wir garantieren eine technische Antwort innerhalb von 24 Geschäftsstunden.

7 Anhang: Erweiterte Konfigurationen

7.1 Ereignisspezifisches Quotenmanagement (Vault-Plan)

Für stark kontrollierte Drops (limitierte Tickets, exklusive Sneaker-Releases usw.) können Sie einen internen Ereignis-Identifikator an den `/consume`-Payload anhängen (siehe Schritt 5).

Wie funktioniert das?

- Im RealNode-Dashboard ist keine vorherige Konfiguration erforderlich.
- Hängen Sie einfach Ihre eigene beliebige Zeichenfolge an (z. B. `KONZERT-2024-BERLIN`).
- Die RealNode Engine isoliert und verfolgt Hardware-Quoten für diesen spezifischen Benutzer automatisch und strikt im Kontext dieses Ereignisses.