VCS One Commerce

Integration Guide

Step-by-Step Setup & Integration Instructions

Getting Started

Prerequisites and Requirements

Before starting integration, ensure you have the necessary infrastructure and access.

- **Development Environment:** Node.js 18+ or Python 3.11+, Git, modern code editor
- **API Access:** VCS One Commerce account with admin privileges and API key generation
- **Backend Services:** PostgreSQL 13+ database, Redis 6+ for caching and sessions
- **SSL Certificates:** Valid SSL certificate for HTTPS webhook endpoints
- **Payment Testing:** Stripe/PayPal sandbox accounts for payment integration testing
- **Domain:** Configured domain or subdomain for storefront deployment

Account Setup

Create your VCS One Commerce account and configure initial settings.

- Sign up at https://vcsmy.com with business email and organization details
- Verify email address and complete business profile
- Configure initial tenant settings: default currency, timezone, legal entity
- Set up admin user credentials and enable multi-factor authentication
- Access admin dashboard at https://vcsmy.com/commerce/admin

API Key Generation & Development Environment Setup

Generate API credentials and set up your development environment.

- **API Keys:** Navigate to Settings → API Keys → Create New Key
- **Key Types:** Read-only, Read-Write, Admin keys with IP whitelisting
- **Secret Storage:** Store API keys securely in environment variables, never commit to version control
- **Key Rotation:** Establish rotation schedule (recommended: every 90 days)
- **Testing Sandbox:** Enable test mode for safe experimentation without production impact

Development Environment Setup

- Install SDK: npm install @vcs-commerce/sdk or pip install vcs-commerce-sdk
- Configure .env file with API_KEY, API_URL, and database connection strings
- Run SDK initialization script to create workspace and test connection
- Clone example repository: git clone github.com/vcs-platforms/commerce-examples
- Start local development server with hot reload enabled

Testing Sandbox Access

Use sandbox environment for integration testing without affecting production data.

- Enable sandbox mode in API configuration: api.useSandbox(true)
- Use sandbox API endpoint: api-sandbox.vcsmy.com
- Test with sample data: products, customers, payment methods provided in SDK
- Validate API responses and error handling using test scenarios
- Review sandbox logs and debugging output for troubleshooting

Quick Start Integration

5-Minute Setup Walkthrough

Get up and running with a basic integration in 5 minutes.

Step 1: Initialize Project

```
npm init -y npm install @vcs-commerce/sdk
```

Step 2: Configure SDK

```
import { VCSCommerce } from '@vcs-commerce/sdk'; const commerce = new VCSCommerce({ apiKey:
process.env.VCS_API_KEY, environment: 'sandbox' });
```

Step 3: Fetch Products

```
const products = await commerce.products.list({ limit: 20, isActive: true });
```

Basic Product Catalog Integration

Connect your product catalog to VCS One Commerce.

- **Product Import:** Bulk import products via CSV or JSON API endpoints
- **Category Setup:** Create category hierarchy and assign products to categories
- **Variant Configuration:** Define product variants (size, color, material)
- **Inventory Sync:** Synchronize stock levels from your inventory system
- **Image Upload:** Upload product images via media API with CDN hosting

Simple Checkout Flow & First API Calls

Implement a basic checkout process.

Add to Cart

```
const cart = await commerce.cart.create({ items: [{ productId: 'prod_123', variantId: 'var_456',
quantity: 2 }] });
```

Calculate Tax & Shipping

```
const quote = await commerce.checkout.calculateTotal(cart.id, { shippingMethod: 'standard',
destination: { country: 'US', state: 'CA' } });
```

Process Payment

```
const order = await commerce.orders.create({ cartId: cart.id, payment: { method: 'stripe',
paymentIntentId: paymentIntent.id } });
```

Testing Checklist

- ✓ API authentication successful
- ✓ Product catalog loads correctly
- ✓ Cart creation and updates work
- ✓ Tax calculation accurate
- ✓ Shipping costs calculated
- ✓ Payment processing functional
- ✓ Order confirmation delivered
- ✓ Webhook events received

Advanced Integrations

Payment Gateway Integration

Connect your preferred payment processors for secure transaction handling.

Stripe Integration

- **Create Stripe Account:** Sign up at stripe.com and obtain API keys
- **Install SDK:** npm install stripe
- **Configure Webhooks:** Register webhook endpoint in Stripe dashboard
- **Test Payments:** Use test cards (4242 4242 4242 4242) for sandbox testing

Stripe Setup Example

```
import Stripe from 'stripe'; const stripe = new Stripe(process.env.STRIPE_SECRET_KEY); const
paymentIntent = await stripe.paymentIntents.create({ amount: totalAmount, currency: 'usd', metadata:
{ orderId: order.id } });
```

Webhook Configuration

- Register webhook URL in Stripe dashboard: https://yourdomain.com/webhooks/stripe
- Verify webhook signature to ensure authenticity
- Handle payment_intent.succeeded and payment_intent.failed events
- Update order status in VCS One Commerce based on payment status

PayPal Integration

- Create PayPal Business account and obtain client ID and secret
- Install SDK: npm install @paypal/checkout-server-sdk
- Configure PayPal REST API credentials
- Set up return/cancel URLs for checkout flow

Testing Procedures

- Use PayPal sandbox accounts for development testing
- Test successful payment flow and cancellation scenarios
- Validate webhook processing and event handling
- Perform end-to-end checkout with real payment methods
- Monitor error logs and implement retry logic for failed transactions

Shipping Integration

Integrate with shipping carriers for rate calculation and label generation.

DHL Integration

- Register for DHL API access and obtain API credentials
- Install DHL Express SDK or use REST API directly
- Configure service codes (Express, Economy) and delivery options
- Implement rate quotation API for real-time shipping costs

Shippo Integration

- Sign up for Shippo account and get API key
- Install Shippo SDK: npm install shippo
- Configure carrier accounts (USPS, FedEx, UPS) in Shippo dashboard
- Use multi-carrier rate shopping for cost optimization

Tracking Integration & Label Generation

- Set up webhook notifications for shipment status updates
- Poll carrier tracking APIs for real-time shipment status
- Display tracking information in customer order portal
- Generate shipping labels via API with print-ready PDF
- Automated customs documentation for international shipments

Headless Storefront Setup

Next.js Integration Example

Build a modern storefront using Next.js with server-side rendering.

Project Setup

```
npx create-next-app@latest commerce-storefront cd commerce-storefront npm install @vcs-commerce/sdk
npm install @vcs-commerce/ui-components
```

Product Listing Page

```
import { getProducts } from '@/lib/commerce'; export async function getServerSideProps() { const
products = await getProducts({ limit: 20 }); return { props: { products } }; } export default
function Products({ products }) { return ( {products.map(product => ( )} ); }
```

React Components Setup

- Install UI library: npm install @vcs-commerce/ui-components tailwindcss
- Import pre-built components: ProductCard, Cart, Checkout forms
- Customize styling with Tailwind CSS classes or theme configuration
- Implement responsive design with mobile-first approach
- Add loading states and error boundaries for better UX

Shopping Cart Implementation

- Create cart context for state management across components
- Implement add/remove/update quantity functionality
- Persist cart in localStorage for guest users
- Sync cart with customer account for logged-in users
- Display real-time cart totals with tax and shipping

Multi-Region Configuration

Adding New Regions

Expand your commerce operations to new geographic markets.

- **Region Selection:** Add regions in Settings → Regions → New Region
- **Geographic Coverage:** Specify countries, states/provinces for this region
- **Local Presence:** Configure warehouses, fulfillment centers, return addresses
- **Shipping Zones:** Define shipping zones with carrier-specific rates
- **Tax Configuration:** Set up local tax rates, exemptions, and compliance rules

Currency Setup & Tax Configuration

Configure multi-currency support and tax compliance.

- **Currency Setup:** Enable currencies: USD, EUR, GBP, JPY with conversion rates
- **Exchange Rates:** Configure auto-sync with forex providers (XE, OANDA)
- **Rounding Rules:** Define currency-specific rounding (e.g., JPY: no decimals)
- **Tax Configuration:** Upload tax tables for VAT, GST, sales tax by jurisdiction
- **Incoterms:** Set default incoterms (DDP, DAP, CIF) for international orders
- **Compliance:** Enable automatic tax calculation with GeoIP detection

Localization Settings & AI Localization Tuning

Adapt your store for local markets.

- **Languages:** Enable languages: English, Chinese, Japanese, Spanish with RTL support
- **Content Translation:** Upload translations or use AI-powered translation
- **Cultural Adaptation:** Adjust imagery, colors, layouts for cultural preferences
- **Payment Methods:** Enable local payment methods (Alipay, GrabPay, Klarna)
- **AI Localization Tuning:** Train AI models on regional preferences and behavior
- **A/B Testing:** Test localization effectiveness with conversion tracking

Vendor Marketplace Integration

Vendor Onboarding API

Automate vendor registration and verification.

- **Registration Endpoint:** POST /api/v1/vendors/register
- **Business Information:** Company name, tax ID, business license upload
- **KYC/KYB Verification:** Automated identity and business verification
- **Bank Account Setup:** Payment method configuration for payouts
- **Product Catalog Import:** Bulk product upload via CSV or API
- **Onboarding Status:** Track progress and requirements in vendor dashboard

SLA Management & Payout Configuration

Configure vendor performance expectations and payment processing.

- **Delivery SLAs:** Define shipping time commitments by product category and region
- **Quality Standards:** Set return rate thresholds, review rating requirements
- **Automated Alerts:** Notify vendors when SLA violations are at risk
- **Payout Schedule:** Configure weekly, bi-weekly, or monthly payout frequencies
- **Commission Structure:** Define tiered commission rates based on sales volume
- **Multi-Currency Payouts:** Support multiple payment methods and currencies

Multi-Vendor Product Catalog

- **Product Attribution:** Link products to vendor accounts with ownership rights
- **Duplicate Management:** Handle same products from multiple vendors with best price
- **Vendor Storefronts:** Enable vendor-specific landing pages and product collections
- **Inventory Pooling:** Aggregated inventory display across multiple vendors
- **Fulfillment Routing:** Automated routing to vendor closest to customer

Ready to Build?

Support Resources & Community

- **Documentation:**
  https://vcsmy.com/docs
- **API Reference:**
  https://vcsmy.com/api
- **Code Examples:**
  https://vcsmy.com/examples
- **Community Forum:**
  https://vcsmy.com/community
- **Email Support:**
  support@vcsmy.com

Value Creating Solutions Sdn Bhd