Univapay Integration Guide: Comprehensive Implementation Documentation

Univapay provides a robust payment platform specializing in Asian markets with comprehensive support for both one-time payments and recurring subscriptions. GitHub **The platform does not offer an official Python SDK**, but Django REST Framework integration is achievable through direct API calls using JWT authentication. Recent 2025 updates show active development across all SDKs and payment widgets. itbusinesstoday GitHub

Frontend widget implementation delivers seamless checkout experiences

Univapay's frontend widget supports both HTML declarative setup and JavaScript programmatic implementation through a CDN-delivered solution. Univapay Univapay

Widget initialization and basic setup

The widget loads via CDN at `https://widget.univapay.com/client/checkout.js` and supports two implementation approaches: Univapay Univapay

HTML Declarative Setup:

```html
<script src="https://widget.univapay.com/client/checkout.js"></script>
<span
  data-app-id="<TOKEN>"
  data-checkout="payment"
  data-amount="1000"
  data-currency="jpy">
</span>
```

JavaScript Programmatic Setup:

```javascript
```

```
var checkout = UnivapayCheckout.create({
  appId: "<TOKEN>",
  checkout: "payment",
  amount: 1000,
  currency: "jpy",
  cvvAuthorize: true
});

document.querySelector("#payment-button").onclick = function() {
  checkout.open();
};
```

The programmatic approach offers greater flexibility for dynamic payment flows and custom UI integration, while the HTML approach provides faster implementation for standard checkout scenarios. Univapay

Configuration and customization capabilities

Core configuration parameters include checkout type ("payment", "token", or "subscription"), amount and currency settings, CVV authorization requirements, and token type specification ("one-time", "recurring", or "subscription").

Visual customization options encompass theme selection through the merchant console, individual color modifications, custom store logos, and customizable button text via the `data-txt` attribute. Store-level settings control payment method acceptance (credit cards, prepaid, debit, foreign cards), card brand restrictions, geographic IP-based restrictions, and recurring payment capabilities. Univapay

Event handling and framework integration

The widget provides comprehensive event handling through both window-level event listeners and callback functions: Univapay Univapay

```javascript
```

```
// Event listeners
window.addEventListener("univapay:success", (e) => {
  console.info("Success event", e);
});

window.addEventListener("univapay:error", (e) => {
  console.error("Error event", e);
});

// Callback approach
UnivapayCheckout.create({
  onSuccess: function(result) { console.log(result); },
  onError: function(error) { console.log(error); },
  onTokenCreated: function(result) { console.log(result); }
});
```

React integration follows standard hooks patterns using useEffect and useState for state management, while **Vue and Angular** implementations use similar lifecycle hooks. A complete Next.js proof-of-concept demonstrates server-side integration patterns. GitHub

Django backend integration requires custom API implementation

Critical finding: Univapay provides no official Python SDK. Integration must be accomplished through direct HTTP API calls using the requests library and JWT authentication. Univapaycast

Authentication and API client setup

Univapay uses a two-part JWT token system consisting of an application token and secret: Univapay Univapay

python

```python
# settings.py
UNIVAPAY_APP_TOKEN = 'your-app-token'
UNIVAPAY_APP_SECRET = 'your-app-secret'
UNIVAPAY_ENDPOINT = 'https://api.univapay.com'

# API client implementation
class UnivapayClient:
    def __init__(self):
        self.base_url = settings.UNIVAPAY_ENDPOINT
        self.headers = {
            'Authorization': f'Bearer {settings.UNIVAPAY_APP_SECRET}.{settings.UNIVAPAY_APP_TOKEN}',
            'Content-Type': 'application/json'
        }

    def make_request(self, method, endpoint, data=None):
        url = f"{self.base_url}/{endpoint}"
        response = requests.request(method, url, headers=self.headers, json=data)
        return response.json()
```

Database models and API endpoints

Recommended database structure includes Payment models with status tracking, ProviderPayment models for Univapay charge ID mapping, and WebhookEvent models for event processing: GitHub

```python
class Payment(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    status = models.CharField(max_length=20, choices=PAYMENT_STATUS_CHOICES)
    transaction_token_id = models.CharField(max_length=255, blank=True)

class ProviderPayment(models.Model):
    payment = models.OneToOneField(Payment, on_delete=models.CASCADE)
    provider_id = models.CharField(max_length=255)  # Univapay charge ID
    provider_data = models.JSONField(default=dict)
```

API endpoint implementation follows Django REST Framework patterns with proper error handling and transaction safety using Django's atomic decorators.

Webhook implementation and verification

Webhooks require CSRF exemption and signature verification: Unipaygateway DEV Community

python

```python
@method_decorator(csrf_exempt, name='dispatch')
class UnivapayWebhookAPIView(APIView):
    def post(self, request):
        if not self.verify_webhook_signature(request):
            return Response({'error': 'Invalid signature'}, status=400)

        event_data = request.data
        WebhookEvent.objects.create(
            event_type=event_data.get('event'),
            provider_id=event_data.get('id'),
            payload=event_data
        )

        return Response({'status': 'ok'})
```

A working **Flask proof-of-concept** exists (GitHub: NoManNayeem/UnivaPay-POC) demonstrating successful Python backend integration with user authentication, payment processing, subscription management, and comprehensive webhook handling. GitHub

Payment types support enables flexible billing models

Univapay's unified platform handles both one-time transactions and recurring subscriptions through differentiated token types and processing flows.

One-time payment implementation

Transaction tokens for one-time payments remain valid for 5 minutes after creation and support multiple payment methods including credit cards, QR codes (Alipay, WeChat Pay), and online payments. GitHub The platform provides **authorization and capture functionality** for delayed charge processing and **concurrent token creation** for high-volume scenarios with built-in rate limiting.

Recurring subscription management

Subscription tokens enable indefinite recurring billing with configurable intervals and amounts. The platform supports **different initial and recurring amounts**, flexible billing cycles (monthly, quarterly, annually), and **automatic retry logic** with 3 attempts over 1-week intervals for failed payments.

Advanced subscription features include bulk CSV billing for existing customers, Univapaycast subscription modification capabilities, comprehensive status management (active, suspended, canceled), and customer identification through multiple methods (Customer ID, email, reference ID). Univapaycast

Mixed payment scenario handling

The same Univapay integration supports **both payment types simultaneously** through unified API endpoints, shared customer profiles, and consolidated reporting. This enables complex business models like initial setup fees (one-time) combined with recurring service charges, usage-based billing adjustments, and comprehensive customer lifecycle management.

Latest API versions reflect active 2025 development

Univapay's infrastructure demonstrates **active maintenance and development** with recent 2025 updates across all components.

Current API structure and versions

The **primary API endpoint** `https://api.univapay.com` follows RESTful architecture with JSON responses. **Authentication uses JWT-based tokens** with both store-type and merchant-type application tokens supported. Univapay GitHub The platform provides **immediate test environment access** with comprehensive test card numbers.

SDK versions and recent updates

Node.js SDK version 4.0.121 was published recently with TypeScript support and tree-shaking via `univapay-node-es`. npm npmjs **Java SDK version 0.2.48** includes builder patterns and pagination support. GitHub GitHub **PHP SDK** received updates on August 22, 2025, and the **WooCommerce plugin** was updated September 7, 2025. GitHub

Supported payment methods and regions

The platform **specializes in Asian markets** with comprehensive coverage including Japan (PayPay, d払い, Merpay, Rakuten Pay), China (Alipay, WeChat Pay), South Korea (KakaoPay), and **20+ mobile wallets across ASEAN** countries. UnivaPay +3 Multi-currency credit card processing and dynamic currency conversion support international transactions.

Security implementation follows enterprise standards

Univapay maintains **PCI DSS compliance** and Privacy Mark certification UnivaPay univapay with comprehensive security features for both frontend and backend integration. UnivaPay

Authentication methods and security practices

JWT-based authentication uses application tokens with secrets, where store tokens include **domain validation** for browser security. Authorization headers follow the format `Bearer {secret}.{jwt}` for authenticated requests. Univapay GitHub **Critical security requirement**: secrets must never be exposed in frontend code and are designed exclusively for backend integration. Univapay

Webhook security and verification

Webhooks implement **HMAC-SHA256 signature verification** with X-Signature headers for authentication. The platform provides **automatic retry mechanisms** with defined schedules for failed deliveries and supports both authenticated and non-authenticated webhook URLs. Unipaygateway

Fraud prevention and compliance features

EMV 3-D Secure 2.0 is mandatory for all credit card transactions, providing risk-based authentication and merchant liability protection. UnivaPay univapay The platform includes **built-in fraud detection**, transaction monitoring, and IP address tracking capabilities.

GDPR compliance requires proper consent management, data minimization practices, and implementation of privacy rights including data access, erasure, and portability. The tokenization approach **reduces PCI compliance scope** for merchants by avoiding direct card data storage. UnivaPay univapay

Code examples demonstrate practical implementation patterns

Widget initialization with error handling

javascript

```
// Complete widget setup with comprehensive event handling
const checkout = UnivapayCheckout.create({
  appId: process.env.NEXT_PUBLIC_UNIVAPAY_APP_ID,
  checkout: "token",
  amount: 1000,
  currency: "jpy",
  opened: function() { console.log("Widget Opened"); },
  onSuccess: function(result) {
    // Handle successful payment
    window.location.href = `/success/${result.id}`;
  },
  onError: function(error) {
    // Display user-friendly error message
    showErrorMessage("Payment failed. Please try again.");
  },
  onTokenCreated: function(result) {
    // Send token to backend for charge processing
    submitPaymentToken(result.id);
  }
});
```

Django payment processing endpoint

python

```
class CreateChargeAPIView(APIView):
    def post(self, request):
        transaction_token_id = request.data.get('transaction_token_id')
        amount = request.data.get('amount')

        try:
            with transaction.atomic():
                # Create payment record
                payment = Payment.objects.create(
                    user=request.user,
                    amount=amount,
                    transaction_token_id=transaction_token_id
                )

                # Call Univapay API
                client = UnivapayClient()
                charge_response = client.make_request('POST', 'charges', {
                    'transaction_token_id': transaction_token_id,
                    'amount': amount,
                    'currency': 'JPY'
                })

                # Store provider response
                ProviderPayment.objects.create(
                    payment=payment,
                    provider_id=charge_response['id'],
                    provider_data=charge_response
                )

                return Response({'payment_id': payment.id, 'status': 'created'})

        except Exception as e:
            return Response({'error': str(e)}, status=400)
```

Subscription management implementation

```
javascript
```

```
// Subscription creation with lifecycle handling
const subscriptionCheckout = UnivapayCheckout.create({
  appId: APP_TOKEN,
  checkout: "subscription",
  amount: 2000,  // Initial payment
  currency: "jpy",
  onSubscriptionCreated: function(result) {
    // Store subscription ID for future management
    saveSubscriptionId(result.subscription_id);

    // Set up subscription monitoring
    monitorSubscriptionStatus(result.subscription_id);
  }
});
```

Implementation recommendations prioritize security and reliability

Start with the **Flask proof-of-concept patterns** as a foundation, implementing proper error handling with Django's transaction.atomic for payment operations. Use **Celery for asynchronous webhook processing** and status polling to avoid blocking request-response cycles. GitHub

Testing strategy should utilize Univapay's comprehensive test environment with provided test cards, implement webhook endpoint testing using ngrok for local development, and create both unit tests with mocked API responses and integration tests against the sandbox environment.

Production deployment requires proper secret management through environment variables, comprehensive logging and monitoring for payment flows, implementation of idempotency keys for charge creation, and establishment of proper webhook endpoints with signature verification.

The absence of an official Python SDK necessitates **direct API implementation**, but the comprehensive documentation and working proof-of-concept provide sufficient guidance for robust Django REST Framework integration supporting both one-time payments and recurring subscriptions on a unified platform.