Anonymous SecurePay API Documentation

Version: 1.0
Environment: Demo / Sandbox
Base URL: `https://api.anonymoussecurepay.example/v1`

1. Overview

Anonymous SecurePay is a payment gateway API designed to allow merchants and applications to initiate payments, verify transactions, receive payment status updates, and manage payment-related transaction data.

This documentation describes the Demo API integration for Anonymous SecurePay.

The API supports the following transaction flow:

Merchant Website → Anonymous SecurePay API → Payment Processing → Transaction Verification → Payment Confirmation → WooCommerce Order Update

A payment must not be treated as successful until Anonymous SecurePay returns a verified successful transaction status.

---

2. Authentication

All API requests should include an API key in the request headers.

Header

```
Authorization: Bearer YOUR_API_KEY
Content-Type: application/json
Accept: application/json
```

Example:

```
Authorization: Bearer asp_demo_xxxxxxxxxxxx
Content-Type: application/json
```

For security, API keys should never be exposed in frontend JavaScript or publicly accessible pages. All sensitive API requests should be performed server-side.

---

3. Create a Payment

Creates a new payment transaction.

Endpoint

```
POST /payments
```

Full Demo Endpoint

```
https://api.anonymoussecurepay.example/v1/payments
```

Request Headers

```
Authorization: Bearer YOUR_API_KEY
Content-Type: application/json
Accept: application/json
```

Request Parameters

Parameter	Type	Required	Description
amount	number	Yes	Payment amount
currency	string	Yes	Transaction currency, for example USD
reference	string	Yes	Unique merchant transaction reference
order_id	string	Yes	Merchant or WooCommerce order ID
customer_email	string	No	Customer email address
customer_name	string	No	Customer name
callback_url	string	Yes	URL for payment status notifications
return_url	string	No	URL where the customer is redirected after payment
metadata	object	No	Additional merchant information

Example Request

```
{
  "amount": 150.00,
  "currency": "USD",
```

```
  "reference": "ASP-ORDER-10025-ABC123",
  "order_id": "10025",
  "customer_email": "customer@example.com",
  "customer_name": "John Doe",
  "callback_url": "https://merchant.example.com/?wc-api=anonymous_securepay",
  "return_url": "https://merchant.example.com/checkout/order-received/10025/",
  "metadata": {
    "source": "woocommerce",
    "gateway": "anonymous_securepay"
  }
}
```

Example Successful Response

```
{
  "success": true,
  "status": "pending",
  "message": "Payment created successfully",
  "data": {
    "transaction_id": "ASP-TXN-8F72A91D",
    "reference": "ASP-ORDER-10025-ABC123",
    "amount": 150.00,
    "currency": "USD",
    "payment_url": "https://checkout.anonymoussecurepay.example/pay/ASP-
TXN-8F72A91D",
    "expires_at": "2026-09-01T20:00:00Z"
  }
}
```

A successful creation of a payment does **not** mean that the customer has paid. The transaction status should initially be treated as `pending` until verification confirms the payment.

---

4. Redirect Customer to Payment Page

After creating a transaction, redirect the customer to the `payment_url` returned by the API.

Example:

```
https://checkout.anonymoussecurepay.example/pay/ASP-TXN-8F72A91D
```

The customer completes the payment on the Anonymous SecurePay payment page.

After payment processing, Anonymous SecurePay may:

1. Send a server-to-server webhook to the merchant.
2. Redirect the customer to the configured return URL.
3. Allow the merchant to verify the transaction using the verification endpoint.

The redirect alone must not be used as proof of successful payment.

---

5. Verify a Transaction

The merchant should verify the transaction with Anonymous SecurePay before marking an order as paid.

Endpoint

```
GET /payments/{transaction_id}
```

Example

```
GET https://api.anonymoussecurepay.example/v1/payments/ASP-TXN-8F72A91D
```

Example Successful Response

```
{
  "success": true,
  "status": "successful",
  "message": "Transaction verified successfully",
  "data": {
    "transaction_id": "ASP-TXN-8F72A91D",
    "reference": "ASP-ORDER-10025-ABC123",
    "order_id": "10025",
    "amount": 150.00,
    "currency": "USD",
    "status": "successful",
    "paid_at": "2026-09-01T18:15:00Z"
  }
}
```

---

6. Payment Statuses

Anonymous SecurePay supports the following payment statuses.

Status	Description
pending	Payment has been created but is not yet confirmed
processing	Payment is currently being processed
successful	Payment has been successfully verified
failed	Payment failed
cancelled	Payment was cancelled
expired	Payment session expired
refunded	Payment was refunded

Important Payment Rule

A WooCommerce order should only be marked as paid when the API verification response confirms:

```
{
  "success": true,
  "status": "successful"
}
```

or when the verified transaction data contains:

```
{
  "data": {
    "status": "successful"
  }
}
```

---

7. Verify Payment by Reference

Merchants may also verify a transaction using their original payment reference.

Endpoint

```
GET /payments/verify/{reference}
```

Example

```
GET https://api.anonymoussecurepay.example/v1/payments/verify/ASP-ORDER-10025-ABC123
```

Example Response

```
{
  "success": true,
  "status": "successful",
  "data": {
    "transaction_id": "ASP-TXN-8F72A91D",
    "reference": "ASP-ORDER-10025-ABC123",
    "amount": 150.00,
    "currency": "USD",
    "status": "successful"
  }
}
```

The merchant should validate that:

1. The transaction reference matches the expected order reference.
2. The amount matches the expected order amount.
3. The currency matches the expected currency.
4. The transaction status is `successful`.
5. The transaction has not previously been used for another order.

---

8. Webhook Notifications

Anonymous SecurePay sends a webhook notification when the status of a transaction changes.

The merchant provides the webhook URL using the `callback_url` parameter during payment creation.

Example Webhook

```
{
  "event": "payment.successful",
  "transaction_id": "ASP-TXN-8F72A91D",
  "reference": "ASP-ORDER-10025-ABC123",
  "status": "successful",
  "amount": 150.00,
  "currency": "USD",
```

```
  "paid_at": "2026-09-01T18:15:00Z"
}
```

Other possible events include:

```
payment.pending
payment.processing
payment.successful
payment.failed
payment.cancelled
payment.expired
payment.refunded
```

---

9. Webhook Security

Webhook requests should include a signature.

Example Header

```
X-AnonymousSecurePay-Signature: SIGNATURE_VALUE
```

The merchant should calculate the expected signature using the webhook secret and compare it with the signature received in the request.

Example:

```
HMAC-SHA256(
    raw_request_body,
    WEBHOOK_SECRET
)
```

The webhook should be rejected if the signature does not match.

Important

Even after receiving a valid webhook, the recommended payment flow is:

```
Receive Webhook
        ↓
Validate Webhook Signature
```

```
        ↓
Locate Order Using Reference
        ↓
Verify Transaction with API
        ↓
Validate Amount and Currency
        ↓
Check Transaction Has Not Been Processed Before
        ↓
Mark WooCommerce Order as Paid
```

---

10. WooCommerce Order Processing

When a verified payment is successful, the WooCommerce integration should save the transaction ID:

```
$order->set_transaction_id($transaction_id);

$order->update_meta_data(
    '_gateway_transaction_id',
    $transaction_id
);
```

The order can then be marked as paid:

```
$order->payment_complete($transaction_id);
```

WooCommerce will apply the appropriate paid order status based on the order configuration and product type.

The integration should also save a payment processing flag to prevent duplicate processing:

```
$order->update_meta_data(
    '_gateway_payment_processed',
    'yes'
);
```

Example metadata:

```
_gateway_transaction_id
_gateway_payment_processed
```

```
_gateway_payment_processed_at
_gateway_payment_attempts
_gateway_last_payment_attempt
_gateway_last_failed_at
_gateway_cancelled_at
```

---

11. Duplicate Transaction Protection

Each Anonymous SecurePay transaction ID should only be associated with one WooCommerce order.

Before processing a payment, the integration should check whether the transaction has already been processed.

Example transaction metadata:

```
ASP-TXN-8F72A91D
```

If the transaction ID has already been associated with a completed payment, the new request should be rejected.

Recommended response:

```
{
  "success": false,
  "message": "This transaction has already been processed."
}
```

---

12. Payment Verification Logic

The following validation should be performed before confirming payment:

```
1. Find WooCommerce order.
2. Confirm order exists.
3. Confirm transaction ID exists.
4. Verify transaction with Anonymous SecurePay API.
5. Confirm API response is successful.
6. Confirm transaction status is successful.
7. Confirm reference matches order.
```

```
8. Confirm amount matches order total.
9. Confirm currency matches order currency.
10. Check transaction has not already been processed.
11. Save transaction ID.
12. Mark order as paid.
13. Save processing timestamp.
14. Add an order note.
```

Example successful order note:

```
Anonymous SecurePay payment verified successfully.
Transaction ID: ASP-TXN-8F72A91D
```

---

13. Error Responses

All errors should return JSON.

Invalid Authentication

```json
{
  "success": false,
  "error": {
    "code": "invalid_api_key",
    "message": "Invalid or missing API credentials."
  }
}
```

Transaction Not Found

```json
{
  "success": false,
  "error": {
    "code": "transaction_not_found",
    "message": "The requested transaction could not be found."
  }
}
```

Invalid Amount

```
{
  "success": false,
  "error": {
    "code": "amount_mismatch",
    "message": "The transaction amount does not match the order amount."
  }
}
```

Duplicate Transaction

```
{
  "success": false,
  "error": {
    "code": "duplicate_transaction",
    "message": "This transaction has already been processed."
  }
}
```

Payment Not Complete

```
{
  "success": false,
  "error": {
    "code": "payment_not_completed",
    "message": "The payment has not been successfully completed."
  }
}
```

---

14. Recommended HTTP Status Codes

HTTP Code	Meaning
200	Request successful
201	Payment created successfully
400	Invalid request
401	Authentication failed
403	Request not authorized

HTTP Code	Meaning
404	Resource not found
409	Duplicate transaction
422	Validation failed
429	Too many requests
500	Internal server error
503	Service temporarily unavailable

---

15. Rate Limiting

For the demo environment, merchants should avoid sending excessive requests.

A recommended limit is:

```
60 requests per minute per API key
```

When the limit is exceeded, the API should return:

```
HTTP/1.1 429 Too Many Requests
```

Example response:

```
{
  "success": false,
  "error": {
    "code": "rate_limit_exceeded",
    "message": "Too many requests. Please try again later."
  }
}
```

---

16. Demo Test Transaction

Create Payment Request

```
{
  "amount": 100.00,
  "currency": "USD",
  "reference": "ASP-DEMO-ORDER-1001",
  "order_id": "1001",
  "callback_url": "https://merchant.example.com/?wc-api=anonymous_securepay"
}
```

Payment Creation Response

```
{
  "success": true,
  "status": "pending",
  "data": {
    "transaction_id": "ASP-DEMO-TXN-100001",
    "reference": "ASP-DEMO-ORDER-1001",
    "payment_url": "https://checkout.anonymoussecurepay.example/pay/ASP-DEMO-
TXN-100001"
  }
}
```

Successful Verification Response

```
{
  "success": true,
  "status": "successful",
  "data": {
    "transaction_id": "ASP-DEMO-TXN-100001",
    "reference": "ASP-DEMO-ORDER-1001",
    "amount": 100.00,
    "currency": "USD",
    "status": "successful"
  }
}
```

---

17. Security Requirements

Merchants integrating with Anonymous SecurePay should:

- Keep API keys secret.
- Never expose secret keys in JavaScript.
- Use HTTPS for all API requests.
- Verify webhook signatures.
- Verify transaction status server-side.
- Validate payment amount.
- Validate currency.
- Validate the merchant reference.
- Prevent duplicate transaction processing.
- Log API errors securely.
- Avoid storing unnecessary sensitive payment information.
- Never rely solely on the customer's return URL as payment confirmation.

---

18. Recommended Integration Flow

```
Customer places WooCommerce order
            ↓
WooCommerce creates order
            ↓
Merchant sends Create Payment request
            ↓
Anonymous SecurePay returns Transaction ID
            ↓
Customer is redirected to Payment URL
            ↓
Customer completes payment
            ↓
Anonymous SecurePay sends Webhook
            ↓
Merchant validates Webhook Signature
            ↓
Merchant calls Transaction Verification API
            ↓
Transaction verified as successful
            ↓
Amount, Currency and Reference validated
            ↓
Transaction duplicate check performed
            ↓
WooCommerce payment_complete() executed
```

```
            ↓
Transaction ID saved
            ↓
Customer and merchant receive order confirmation
```

---

19. Support

Gateway Name: Anonymous SecurePay
Environment: Demo / Sandbox
API Version: v1

For production use, replace all demo API endpoints, API keys, webhook secrets, and example transaction IDs with the actual production credentials and endpoints provided by the Anonymous SecurePay payment service.