001K

001k API/v1 Documentation

Comprehensive guide for integration with our API

Author: 001k Bot Team
Generated:

Table of contents

1. 001k API/v1 Documentation

Welcome to the 001k API/v1 Documentation. This API provides a secure and reliable way to interact with our platform programmatically, allowing you to integrate our services into your applications.

1.1 Overview

Our API follows RESTful principles and uses HMAC-SHA256 authentication to ensure secure communication. All requests and responses are in JSON format.

1.2 Authentication

The API uses a custom authentication scheme based on HMAC-SHA256 signatures. See the Authentication section for details on how to authenticate your requests.

1.3 Rate Limiting

Each endpoint has its own rate limit, typically ranging from 30 to 600 requests per minute. The rate limit is specified in each endpoint's documentation.

1.4 Common Features

- **Custom IDs**: Most operations support a `custom_id` parameter that allows you to assign your own unique identifier to track operations.
- **Callbacks**: Many operations support callback URLs that will be called when the operation's status changes.
- **Idempotency**: Operations are designed to be idempotent when using custom IDs, meaning that sending the same request multiple times will not result in duplicate operations.

1.5 Available Endpoints

The API provides the following main categories of endpoints:

- Balances - View your account balances
- Currencies - Get information about available currencies
- Deposits - Generate deposit addresses and track deposits
- Withdrawals - Create and track withdrawal requests
- Internal Transfers - Transfer funds between users
- Swaps - Exchange one currency for another
- Limit Orders - Place and manage limit orders
- AML Checks - Perform anti-money laundering checks on addresses

1.6 Implementation Examples

To help you get started, we provide implementation examples in several programming languages:

- Python
- PHP
- JavaScript

1.7 Server Status

You can check the API server status using the `/status` endpoint, which does not require authentication.

```
GET /status
```

Response:

```
{
  "timestamp": 1676541877000
}
```

The timestamp is in milliseconds since the Unix epoch.

March 21, 2025

March 19, 2025

2. API Authentication

The API uses a custom authentication scheme based on HMAC-SHA256 signatures to ensure secure communication.

2.1 API Keys

To access the API, you need an API key pair consisting of:

- **Access Key**: Used to identify you
- **Secret Key**: Used to sign requests (never share this)

Contact support to obtain your API keys.

2.2 API Key Permissions

API keys have granular permissions that determine which endpoints you can access:

- `allow_balance` - Access to balance information
- `allow_deposit` - Access to deposit functions
- `allow_withdraw` - Access to withdrawal functions
- `allow_swap` - Access to swap functions
- `allow_transfer` - Access to transfer functions
- `allow_aml_check` - Access to AML check functions
- `allow_limit_order` - Access to limit order functions

2.3 IP Whitelisting

For enhanced security, your API key can be restricted to specific IP addresses. Only requests from whitelisted IPs will be accepted. CIDR notation is supported for IP ranges.

2.4 Required Headers

Every authenticated request must include the following headers:

- `X-Access-Key`: Your API access key
- `X-Timestamp`: Current timestamp in milliseconds since Unix epoch
- `X-Signature`: Request signature

2.5 Creating the Signature

The signature is created by:

1. Concatenating your access key, request path, timestamp, and request body
2. Signing this string with your secret key using HMAC-SHA256
3. Converting the result to a hexadecimal string

Signature Algorithm

```
message = access_key + request_path + timestamp + request_body
signature = HMAC-SHA256(secret_key, message)
```

Important Notes

- The timestamp must be within 5 seconds of the server time
- The request path is the full path component of the URL (e.g., `/api/v1/balance`)
- For GET requests, the request body is an empty string
- For POST requests, the request body is the JSON string (not the parsed object, see code examples)

2.6 Authentication Errors

Common authentication errors include:

- `access_key.missed` : The X-Access-Key header is missing
- `timestamp.missed` : The X-Timestamp header is missing
- `signature.missed` : The X-Signature header is missing
- `timestamp.invalid` : The timestamp is invalid or too far from server time
- `access_key.invalid` : The provided access key does not exist
- `access_key.inactive` : The API key is disabled
- `access_key.ip_whitelist` : The client IP is not in the whitelist
- `user.inactive` : The user account is inactive
- `signature.invalid` : The signature does not match

2.7 API Key Security Best Practices

1. **Never share your keys** - It should be known only to your server
2. **Use IP whitelisting** - Restrict API access to trusted IPs or CIRDs
3. **Use HTTPS** - Always make requests over HTTPS, never HTTP
4. **Rotate keys regularly** - Periodically request new API keys
5. **Monitor usage** - Regularly review your API key usage for unauthorized access

March 21, 2025

March 19, 2025

3. Endpoints

3.1 Balance Endpoints

This section covers API endpoints related to account balances.

Get All Balances

Returns the balance information for all currencies in your account.

```
GET /balance
```

Required Permissions

- `allow_balance`

Rate Limit

30 requests per minute

Response

A list of balance objects, one for each currency:

```
[
  {
    "currency": "BTC",
    "available": "0.05000000",
    "frozen": "0.00000000",
    "updated_at": "2023-01-15T14:30:15Z"
  },
  {
    "currency": "ETH",
    "available": "1.25000000",
    "frozen": "0.50000000",
    "updated_at": "2023-01-15T14:30:15Z"
  }
]
```

Response Fields

Field	Type	Description
`currency`	string	Currency code
`available`	string	Available balance that can be used for operations
`frozen`	string	Balance that is temporarily frozen (e.g., in pending operations)
`updated_at`	string	Last time the balance was updated (ISO 8601 format)

Notes

- Balances are returned for all currencies, including those with zero balance
- The `available` and `frozen` fields are strings to preserve precision for decimal values

Usage Example

Request

```
GET /api/v1/balance
X-Access-Key: your_access_key
X-Timestamp: 1676541877000
X-Signature: calculated_signature
```

Response

```
[
  {
    "currency": "BTC",
    "available": "0.05000000",
    "frozen": "0.00000000",
    "updated_at": "2023-01-15T14:30:15Z"
  },
  {
    "currency": "ETH",
    "available": "1.25000000",
```

```
    "frozen": "0.50000000",
    "updated_at": "2023-01-15T14:30:15Z"
  },
  {
    "currency": "USDT",
    "available": "1000.00000000",
    "frozen": "250.00000000",
    "updated_at": "2023-01-15T14:30:15Z"
  }
]
```


Common Use Cases

- **Check available funds**: Before initiating operations like withdrawals or swaps
- **Monitor account activity**: Track changes in balances over time
- **Reconcile transactions**: Verify that operations have correctly affected your balances

March 19, 2025

March 19, 2025

3.2 Currency Endpoints

This section covers API endpoints related to currency information.

Get All Currencies

Returns information about all available currencies and their associated deposit and withdrawal methods.

```
GET /currency
```

Authentication

This endpoint does not require authentication.

Rate Limit

10 requests per minute

Response

A list of currency objects with their available deposit and withdrawal methods:


```
[
  {
    "code": "BTC",
    "name": "Bitcoin",
    "precision": 8,
    "is_fiat": false,
    "is_stable": false,
    "updated_at": "2023-01-15T14:30:15Z",
    "deposit": [
      {
        "code": "BTC",
        "coin": "BTC",
        "network": "bitcoin",
        "contract": null,
        "is_active": true,
        "min_amount": "0.00010000",
        "max_amount": "100.00000000",
        "fee_percent": "0.00",
        "fee_absolute": "0.00000000",
        "required_confirmations": 2,
        "precision": 8
      }
    ],
    "withdraw": [
      {
        "code": "BTC",
        "coin": "BTC",
        "network": "bitcoin",
        "contract": null,
        "is_active": true,
        "min_amount": "0.00050000",
        "max_amount": "10.00000000",
        "fee_percent": "0.00",
        "fee_absolute": "0.00020000",
        "precision": 8,
        "is_tag_required": false,
        "is_beneficiary_required": false
      }
    ]
  },
  {
    "code": "USDT",
    "name": "Tether USD",
    "precision": 6,
    "is_fiat": false,
    "is_stable": true,
    "updated_at": "2023-01-15T14:30:15Z",
    "deposit": [
      {
        "code": "USDTERC20",
        "coin": "USDTERC20",
        "network": "ethereum",
        "contract": {
          "protocol": "ERC20",
          "address": "0xdAC17F958D2ee523a2206206994597C13D831ec7"
        },
        "is_active": true,
        "min_amount": "10.000000",
        "max_amount": "1000000.000000",
        "fee_percent": "0.00",
        "fee_absolute": "0.000000",
        "required_confirmations": 12,
        "precision": 6
      },
      {
        "code": "USDTTRC20",
        "coin": "USDTTRC20",
        "network": "tron",
        "contract": {
          "protocol": "TRC20",
          "address": "TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t"
        },
        "is_active": true,
        "min_amount": "10.000000",
```

```
      "max_amount": "1000000.000000",
      "fee_percent": "0.00",
      "fee_absolute": "0.000000",
      "required_confirmations": 20,
      "precision": 6
    }
  ],
  "withdraw": [
    {
      "code": "USDTERC20",
      "coin": "USDTERC20",
      "network": "ethereum",
      "contract": {
        "protocol": "ERC20",
        "address": "0xdAC17F958D2ee523a2206206994597C13D831ec7"
      },
      "is_active": true,
      "min_amount": "20.000000",
      "max_amount": "500000.000000",
      "fee_percent": "0.00",
      "fee_absolute": "15.000000",
      "precision": 6,
      "is_tag_required": false,
      "is_beneficiary_required": false
    },
    {
      "code": "USDTTRC20",
      "coin": "USDTTRC20",
      "network": "tron",
      "contract": {
        "protocol": "TRC20",
        "address": "TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t"
      },
      "is_active": true,
      "min_amount": "20.000000",
      "max_amount": "500000.000000",
      "fee_percent": "0.00",
      "fee_absolute": "1.000000",
      "precision": 6,
      "is_tag_required": false,
      "is_beneficiary_required": false
    }
  ]
 }
]
```


Response Fields

CURRENCY OBJECT

Field	Type	Description
`code`	string	Currency code (e.g., BTC, ETH, USDT)
`name`	string	Full name of the currency
`precision`	integer	Decimal precision for the currency
`is_fiat`	boolean	Whether the currency is a fiat currency
`is_stable`	boolean	Whether the currency is a stablecoin
`updated_at`	string	Last time the currency information was updated
`deposit`	array	List of available deposit methods
`withdraw`	array	List of available withdrawal methods

DEPOSIT METHOD OBJECT

Field	Type	Description
`code`	string	Unique code for the deposit method
`coin`	string	Coin symbol
`network`	string	Network code (e.g., bitcoin, ethereum, tron)
`contract`	object	Contract details (for token deposits)
`is_active`	boolean	Whether the deposit method is currently active
`min_amount`	string	Minimum deposit amount
`max_amount`	string	Maximum deposit amount
`fee_percent`	string	Percentage fee for deposits
`fee_absolute`	string	Fixed fee for deposits
`required_confirmations`	integer	Number of blockchain confirmations required
`precision`	integer	Decimal precision for this method

WITHDRAWAL METHOD OBJECT

Field	Type	Description
`code`	string	Unique code for the withdrawal method
`coin`	string	Coin symbol
`network`	string	Network code (e.g., bitcoin, ethereum, tron)
`contract`	object	Contract details (for token withdrawals)
`is_active`	boolean	Whether the withdrawal method is currently active
`min_amount`	string	Minimum withdrawal amount
`max_amount`	string	Maximum withdrawal amount
`fee_percent`	string	Percentage fee for withdrawals
`fee_absolute`	string	Fixed fee for withdrawals
`precision`	integer	Decimal precision for this method
`is_tag_required`	boolean	Whether a destination tag/memo is required
`is_beneficiary_required`	boolean	Whether beneficiary details are required

CONTRACT OBJECT

Field	Type	Description
`protocol`	string	Protocol type (e.g., ERC20, TRC20, BEP20)
`address`	string	Contract address on the blockchain

Warnings

- Deposit and withdrawal methods returning in this endpoint may not be accessible for all users. To get exact list of deposit and withdraw methods please use new endpoints in sections.

Common Use Cases

- **Get available currencies**: Show users what currencies are supported
- **Show deposit options**: Present users with available deposit methods and their limits
- **Show withdrawal options**: Present users with available withdrawal methods, fees, and limits
- **Validate operations**: Check if operations meet minimum and maximum amount requirements

March 19, 2025

March 19, 2025

3.3 Deposit Endpoints

This section covers API endpoints related to deposits. Deposit operations allow users to add cryptocurrencies and fiat currencies to their platform balance by receiving funds from external sources. Different cryptocurrency networks and fiat payment methods have specific conditions, fees, and reference requirements.

Get Personal Deposit Methods

Returns deposit methods available to the authenticated user, considering their segments. It also represents the user's deposit limits and fees.

```
GET /deposit-methods
```

Required Permissions

- `allow_deposit`

Rate Limit

30 requests per minute

Query Parameters

Parameter	Type	Required	Description
`currency`	string	No	Filter by currency code
`is_enabled`	boolean	No	Filter by enabled status

Response


```
[
  {
    "code": "btc_native",
    "name": "Bitcoin",
    "method_type": "crypto",
    "currency": {
      "code": "BTC",
      "name": "Bitcoin",
      "precision": 8,
      "is_fiat": false
    },
    "coin": {
      "symbol": "BTC",
      "name": "Bitcoin",
      "network": {
        "code": "BITCOIN",
        "name": "Bitcoin Network",
        "is_mainnet": true
      }
    },
    "network": {
      "code": "BITCOIN",
      "name": "Bitcoin Network",
      "is_mainnet": true
    },
    "min_amount": "0.00010000",
    "max_amount": "10.00000000",
    "fee_percent": "0.00",
    "fee_absolute": "0.00000000",
    "required_confirmations": 2,
    "is_visible": true,
    "is_enabled": true,
    "order_number": 1,
    "updated_at": "2023-01-15T10:00:00Z",
    "spend_rate": "1.00000000",
    "receive_rate": "1.00000000",
    "personal_fee": {
      "percent": "0.00",
      "absolute": "0.00000000"
    },
    "deposit_address": {
      "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
      "tag": null
    }
  },
  {
    "code": "eth_erc20",
    "name": "Ethereum",
    "method_type": "crypto",
    "currency": {
      "code": "ETH",
      "name": "Ethereum",
      "precision": 18,
      "is_fiat": false
    },
    "coin": {
      "symbol": "ETH",
      "name": "Ethereum",
      "network": {
        "code": "ETHEREUM",
        "name": "Ethereum Network",
        "is_mainnet": true
      }
```

```
    },
    "network": {
      "code": "ETHEREUM",
      "name": "Ethereum Network",
      "is_mainnet": true
    },
    "min_amount": "0.00100000",
    "max_amount": "100.00000000",
    "fee_percent": "0.00",
    "fee_absolute": "0.00000000",
    "required_confirmations": 12,
    "is_visible": true,
    "is_enabled": true,
    "order_number": 2,
    "updated_at": "2023-01-15T10:00:00Z",
    "spend_rate": "1.00000000",
    "receive_rate": "1.00000000",
    "personal_fee": {
      "percent": "0.00",
      "absolute": "0.00000000"
    },
    "deposit_address": {
      "address": "0x742d35Cc6634C0532925a3b844Bc454e4438f44e",
      "tag": null
    }
  },
  {
    "code": "xlm_native",
    "name": "Stellar Lumens",
    "method_type": "crypto",
    "currency": {
      "code": "XLM",
      "name": "Stellar Lumens",
      "precision": 7,
      "is_fiat": false
    },
    "coin": {
      "symbol": "XLM",
      "name": "Stellar Lumens",
      "network": {
        "code": "STELLAR",
        "name": "Stellar Network",
        "is_mainnet": true
      }
    },
    "network": {
      "code": "STELLAR",
      "name": "Stellar Network",
      "is_mainnet": true
    },
    "min_amount": "10.0000000",
    "max_amount": "10000.0000000",
    "fee_percent": "0.00",
    "fee_absolute": "0.00000000",
    "required_confirmations": 1,
    "is_visible": true,
    "is_enabled": true,
    "order_number": 3,
    "updated_at": "2023-01-15T10:00:00Z",
    "spend_rate": "1.00000000",
    "receive_rate": "1.00000000",
    "personal_fee": {
      "percent": "0.00",
      "absolute": "0.00000000"
    },
    "deposit_address": {
      "address": "GA7YNBW5CBTJZ3XJN66AZSCT44OWIIF6USOBHABGKH4RZVMOPVUHOSDU",
      "tag": "123456"
    }
  }
]
```

Response Fields

Field	Type	Description
`code`	string	Unique code for the deposit method
`name`	string	Human-readable name of the deposit method
`method_type`	string	Type of payment method (e.g., "crypto", "bank", "card")
`currency`	object	Currency information object
`coin`	object	Cryptocurrency coin information (for crypto methods)
`network`	object	Blockchain network information (for crypto methods)
`min_amount`	string	Minimum deposit amount allowed
`max_amount`	string	Maximum deposit amount allowed (or null if no limit)
`fee_percent`	string	Percentage fee for deposits using this method
`fee_absolute`	string	Fixed fee amount for deposits using this method
`required_confirmations`	integer	Number of blockchain confirmations required (for crypto)
`is_visible`	boolean	Whether the method is visible to the user
`is_enabled`	boolean	Whether the method is currently enabled for use
`order_number`	integer	Sorting order for display
`updated_at`	string	Last update timestamp (ISO 8601 format)
`spend_rate`	string	Exchange rate for the spending currency
`receive_rate`	string	Exchange rate for the receiving currency
`personal_fee`	object	User-specific fee information
`deposit_address`	object	User's deposit address for this method (if already generated)

Generate Deposit Address

Generates a new deposit address for a specific currency and method.

```
POST /deposit/generate_address
```

Required Permissions

- `allow_deposit`

Rate Limit

600 requests per minute

Request Parameters

Parameter	Type	Required	Description
`currency`	string	Yes	Currency code for the deposit
`method`	string	No	Specific deposit method code. If not provided, the default method for the currency will be used. See the Get Personal Deposit Methods endpoint for available methods
`callback_url`	string	No	URL that will be called with status updates
`custom_id`	string	No	Custom identifier for tracking this deposit address

Example Request

```
{
  "currency": "BTC",
  "method": "BTC",
  "callback_url": "https://example.com/callbacks/deposits",
  "custom_id": "my-deposit-address-123"
}
```

Response


```
{
  "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
  "tag": null,
  "network": "bitcoin",
  "currency": "BTC"
}
```


For currencies that require a tag/memo (like XRP, XLM, EOS):


```
{
  "address": "rLW9gnQo7BQhU6igk5keqYnH3TVrCxGRzm",
  "tag": "123456",
  "network": "ripple",
  "currency": "XRP"
}
```


Response Fields

Field	Type	Description
`address`	string	Deposit address
`tag`	string	Destination tag/memo (if required by the currency)
`network`	string	Network code
`currency`	string	Currency code

List Deposits

Returns a list of deposits.

```
GET /deposit
```

Required Permissions

- `allow_deposit`

Rate Limit

600 requests per minute

Query Parameters

Parameter	Type	Required	Description
`limit`	integer	No	Number of results to return (default: 100, max: 200)
`offset`	integer	No	Number of results to skip (for pagination)
`status`	string	No	Filter by status (see Deposit Status Values)
`currency`	string	No	Filter by currency
`custom_id`	string	No	Filter by custom ID
`start_at`	string	No	Filter by start datetime on field `created_at` (inclusive, ISO 8601 format)
`end_at`	string	No	Filter by end datetime on field `created_at` (inclusive, ISO 8601 format)

Response

The API uses a standard offset-based pagination system with `limit` and `offset` parameters. The response is a direct array of deposit objects without pagination metadata wrapper.


```
[
  {
    "code": "DEP123456",
    "custom_id": "my-deposit-123",
    "callback_url": "https://example.com/callbacks/deposits",
    "created_at": "2023-01-15T14:30:15Z",
    "updated_at": "2023-01-15T14:35:20Z",
    "done_at": "2023-01-15T14:35:20Z",
    "method": "BTC",
    "network": "bitcoin",
    "contract": null,
    "status": "success",
    "reason_code": null,
    "reason_message": null,
    "currency": "BTC",
    "coin": "BTC",
```

```
    "amount": "0.05000000",
    "fee_amount": "0.00000000",
    "actual_confirmations": 3,
    "required_confirmations": 2,
    "txid": "7b5968e39e0d4c980a234931b8a2fd21e804eb7e9f7eb5af75b3d89e2972ad3d",
    "attributes": {
      "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
    }
  }
  // More deposits...
]
```

Response Fields

Field	Type	Description
`code`	string	Unique deposit code
`custom_id`	string	Custom identifier provided when generating the address
`callback_url`	string	URL that will be called with status updates
`created_at`	string	Creation timestamp (ISO 8601 format)
`updated_at`	string	Last update timestamp (ISO 8601 format)
`done_at`	string	Completion timestamp (ISO 8601 format)
`method`	string	Deposit method code
`network`	string	Network code
`contract`	object	Contract details (for token deposits)
`status`	string	Current status of the deposit
`reason_code`	string	Error code (if the deposit failed)
`reason_message`	string	Error message (if the deposit failed)
`currency`	string	Currency code
`coin`	string	Coin symbol
`amount`	string	Deposit amount
`fee_amount`	string	Fee amount
`actual_confirmations`	integer	Current number of blockchain confirmations
`required_confirmations`	integer	Number of confirmations required for completion
`txid`	string	Transaction ID on the blockchain
`attributes`	object	Additional attributes (like address and tag)

Pagination

The API uses simple offset-based pagination with `limit` and `offset` parameters. Offset is limited for maximum 10000, if you set more - error will be returned. The default limit is 100, and the maximum limit is 200. The response is a direct list of deposit objects without pagination metadata wrapper. If you need to fetch more than 200 deposits, you should use the `start_at` and `end_at` parameters to fetch deposits in smaller date ranges. List is sorted by creation date in descending order.

Deposit Status Values

Status	Description
`pending`	Deposit is detected, in confirmation procedure
`quarantine`	Deposit suspended for quarantine procedure
`suspended`	System or manager suspended deposit
`success`	Deposit successfully completed
`failed`	Deposit was rejected or failed

Confirmation Status

Note that not only `actual_confirmations` need to be higher than `required_confirmations` to mark the deposit as `success`. But we also do additional checks on the deposit such as AML screening. If the deposit is marked as `pending` or `quarantine`, it means that the deposit is still in the process of being confirmed.

Get Deposit Details

Returns details of a specific deposit.

```
GET /deposit/{code}
```

Required Permissions

- `allow_deposit`

Rate Limit

600 requests per minute

Path Parameters

Parameter	Type	Required	Description
`code`	string	Yes	Unique deposit code

Response

Same as a single deposit object from the list endpoint.

Callback Notifications

If a `callback_url` was provided when generating the deposit address, the system will send POST requests to this URL when the deposit status changes.

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

Callback Security

Currently, callbacks are **not signed**. The signature mechanism for callbacks is planned for a future update. You should validate callback data by making an authenticated API request to verify the deposit status.

Callback Payload

The callback payload contains the same data as the deposit details response, with essential fields for status updates:


```
{
  "code": "DEP123456",
  "custom_id": "my-deposit-123",
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:35:20Z",
  "done_at": "2023-01-15T14:35:20Z",
  "status": "success",
  "reason_code": null,
  "reason_message": null,
  "currency": "BTC",
  "amount": "0.05000000",
  "fee_amount": "0.00000000",
  "txid": "7b5968e39e0d4c980a234931b8a2fd21e804eb7e9f7eb5af75b3d89e2972ad3d",
  "attributes": {
    "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
  }
}
```


Securing Callbacks

Since callbacks are not currently signed, you should implement these security measures:

1. **Whitelist IPs**: Accept callbacks only from the service's IP addresses. Contact support to get the list of server IPs.
2. **Verify Data**: Always make an authenticated API request to `/deposit/{code}` to verify the status and details of any deposit after receiving a callback.
3. **Use HTTPS**: Ensure your callback URL uses HTTPS to encrypt the data in transit.

Common Use Cases

- **Generating deposit addresses**: Create addresses for user deposits
- **Monitoring deposits**: Track the status of incoming deposits
- **Reconciling deposits**: Verify that deposits have been correctly credited to your account

March 21, 2025

March 19, 2025

3.4 Withdrawal Endpoints

This section covers API endpoints related to withdrawals. Withdrawal operations allow users to send funds from their platform balance to external addresses or accounts. Depending on the cryptocurrency or fiat method, withdrawals may require additional attributes like tags, memos, or bank details, and have specific network fees and processing conditions.

Get Personal Withdrawal Methods

Returns withdrawal methods available to the authenticated user, considering their segments. It also reports the user's specific limits and fees for each method.

```
GET /withdrawal-methods
```

Required Permissions

- `allow_withdraw`

Rate Limit

30 requests per minute

Response

Similar to the withdrawal methods in the currencies endpoint, but filtered by user-specific rules.

Create Withdrawal

Initiates a withdrawal request.

```
POST /withdraw
```

Required Permissions

- `allow_withdraw`

Rate Limit

600 requests per minute

Request Parameters

Parameter	Type	Required	Description
`currency`	string	Yes	Currency code for the withdrawal
`method`	string	No	Specific withdrawal method code. If not provided, the default method for the currency will be used. See available methods in your account settings
`amount`	string	Yes	Amount to withdraw
`address`	string	Yes	Destination address
`tag`	string	No	Destination tag/memo (required for some currencies)
`beneficiary`	object	No	Beneficiary details (required for fiat withdrawals)
`callback_url`	string	No	URL that will be called with status updates
`custom_id`	string	No	Custom identifier for tracking this withdrawal
`receive_mode`	string	No	If set to "exact", the recipient will receive exactly the specified amount (fees will be added on top)
`address_book_code`	string	No	Address book entry code to use for withdrawal

Example Request for Crypto Withdrawal

```
{
  "currency": "BTC",
  "method": "BTC",
  "amount": "0.05000000",
  "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
  "callback_url": "https://example.com/callbacks/withdrawals",
  "custom_id": "my-withdrawal-123",
```

```
  "receive_mode": true
}
```


Example Request for Fiat Withdrawal


```
{
  "currency": "EUR",
  "method": "sepa",
  "amount": "1000.00",
  "address": "DE89370400440532013000",
  "beneficiary": {
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@example.com",
    "iban": "DE89370400440532013000"
  },
  "callback_url": "https://example.com/callbacks/withdrawals",
  "custom_id": "my-withdrawal-456",
  "receive_mode": true
}
```


Response


```
{
  "code": "WDR123456",
  "custom_id": "my-withdrawal-123",
  "callback_url": "https://example.com/callbacks/withdrawals",
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:30:15Z",
  "closed_at": null,
  "currency": "BTC",
  "coin": "BTC",
  "method": "BTC",
  "network": "bitcoin",
  "contract": null,
  "status": "pending",
  "reason_code": null,
  "reason_message": null,
  "amount": "0.05000000",
  "fee_amount": "0.00020000",
  "paid_amount": "0.00000000",
  "paid_fee_amount": "0.00000000",
  "txid": null,
  "attributes": {
    "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
  },
  "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
}
```

Response Fields

Field	Type	Description
`code`	string	Unique withdrawal code
`custom_id`	string	Custom identifier provided in the request
`callback_url`	string	URL that will be called with status updates
`created_at`	string	Creation timestamp (ISO 8601 format)
`updated_at`	string	Last update timestamp (ISO 8601 format)
`closed_at`	string	Completion timestamp (ISO 8601 format)
`currency`	string	Currency code
`coin`	string	Coin symbol
`method`	string	Withdrawal method code
`network`	string	Network code
`contract`	object	Contract details (for token withdrawals)
`status`	string	Current status of the withdrawal
`reason_code`	string	Error code (if the withdrawal failed)
`reason_message`	string	Error message (if the withdrawal failed)
`amount`	string	Withdrawal amount requested
`fee_amount`	string	Fee amount reserved
`paid_amount`	string	Withdrawal amount paid
`paid_fee_amount`	string	Fee amount paid
`txid`	string	Transaction ID on the blockchain (once available)
`attributes`	object	Additional attributes (like address and tag)
`address`	string	Addresses of crypto wallet
`tag`	string	Additional tag of crypto wallet
`address_book_code`	string	Identifier of address book that reference's attributes

Beneficiary Object (for fiat withdrawals)

Field	Type	Description
`first_name`	string	First name of the beneficiary
`last_name`	string	Last name of the beneficiary
`email`	string	Email of the beneficiary
`phone`	string	Phone number of the beneficiary
`iban`	string	IBAN of the beneficiary
`tin`	string	Tax identification number of the beneficiary
`edrpou`	string	EDRPOU code (for Ukrainian businesses)

All fields are optional, but some may be required depending on the withdrawal method.

List Withdrawals

Returns a list of withdrawals.

```
GET /withdraw
```

Required Permissions

- `allow_withdraw`

Rate Limit

600 requests per minute

Query Parameters

Parameter	Type	Required	Description
`status`	string	No	Filter by status (see Withdrawal Status Values)
`currency`	string	No	Filter by currency
`custom_id`	string	No	Filter by custom ID
`start_at`	string	No	Filter by start datetime on field `created_at` (inclusive, ISO 8601 format)
`end_at`	string	No	Filter by end datetime on field `created_at` (inclusive, ISO 8601 format)
`limit`	integer	No	Number of results to return (default: 100, max: 200)
`offset`	integer	No	Number of results to skip (for pagination)

Response

The API uses simple offset-based pagination with `limit` and `offset` parameters. Offset is limited for maximum 10000, if you set more - error will be returned. The default limit is 100, and the maximum limit is 200. The response is a direct list of withdrawal objects without pagination metadata wrapper. If you need to fetch more than 200 withdrawals, you should use the `start_at` and `end_at` parameters to fetch withdrawals in smaller date ranges. List is sorted by creation date in descending order.


```
[
  {
    "code": "WDR123456",
    "custom_id": "my-withdrawal-123",
    "callback_url": "https://example.com/callbacks/withdrawals",
    "created_at": "2023-01-15T14:30:15Z",
    "updated_at": "2023-01-15T14:30:15Z",
    "closed_at": null,
    "currency": "BTC",
    "coin": "BTC",
    "method": "BTC",
    "network": "bitcoin",
    "contract": null,
    "status": "pending",
    "reason_code": null,
    "reason_message": null,
    "amount": "0.05000000",
    "fee_amount": "0.00020000",
    "paid_amount": "0.00000000",
    "paid_fee_amount": "0.00000000",
    "txid": null,
    "attributes": {
      "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
    },
    "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
  }
  // More withdrawals...
]
```


Response Fields

Same as the creation withdrawal response.

Pagination

The API uses simple offset-based pagination with `limit` and `offset` parameters. No additional pagination metadata is returned in the response or headers.

Withdrawal Status Values

Status	Description
`pending`	Withdrawal being processed
`suspended`	Withdrawal is suspended or paused
`failed`	Withdrawal was rejected or failed
`success`	Withdrawal successfully completed

Transaction hash

Withdrawal order may receive a transaction hash before order is success, while transaction is pending and not confirmed in blockchain. In some cases such transaction hash may be replaced, if for example transaction gas need to be bumped or transaction is replaced by another one. In such cases transaction hash will be updated in withdrawal order.

Get Withdrawal Details

Returns details of a specific withdrawal.

```
GET /withdraw/{code}
```

Required Permissions

- `allow_withdraw`

Rate Limit

600 requests per minute

Path Parameters

Parameter	Type	Required	Description
`code`	string	Yes	Unique withdrawal code

Response

Same as a single withdrawal object from the list endpoint.

Callback Notifications

If a `callback_url` was provided when creating the withdrawal, the system will send POST requests to this URL when the withdrawal status changes.

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

> ⚠ **Callback Security**

Currently, callbacks are **not signed**. The signature mechanism for callbacks is planned for a future update. You should validate callback data by making an authenticated API request to verify the withdrawal status.

Callback Payload

The callback payload contains the same data as the withdrawal details response, with essential fields for status updates:


```
{
  "code": "WDR123456",
  "custom_id": "my-withdrawal-123",
  "callback_url": "https://example.com/callbacks/withdrawals",
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:30:15Z",
  "closed_at": null,
  "currency": "BTC",
  "coin": "BTC",
  "method": "BTC",
  "network": "bitcoin",
  "contract": null,
  "status": "pending",
  "reason_code": null,
  "reason_message": null,
  "amount": "0.05000000",
  "fee_amount": "0.00020000",
  "paid_amount": "0.00000000",
  "paid_fee_amount": "0.00000000",
  "txid": null,
  "attributes": {
    "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
  },
  "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
}
```


Securing Callbacks

Since callbacks are not currently signed, you should implement these security measures:

1. **Whitelist IPs**: Accept callbacks only from the service's IP addresses. Contact support to get the list of server IPs.
2. **Verify Data**: Always make an authenticated API request to `/withdraw/{code}` to verify the status and details of any withdrawal after receiving a callback.
3. **Use HTTPS**: Ensure your callback URL uses HTTPS to encrypt the data in transit.

Common Use Cases

- **Initiating withdrawals**: Send funds to external addresses
- **Monitoring withdrawals**: Track the status of outgoing withdrawals
- **Reconciling withdrawals**: Verify that withdrawals have been correctly processed

March 21, 2025

March 19, 2025

3.5 Swap Endpoints

This section covers API endpoints related to currency swaps. Swap operations allow users to exchange one currency for another within the platform using system-calculated rates. The main advantage of swaps is that they execute almost instantly at the current market rate, without waiting for specific price conditions to be met. Swaps are ideal for users who want to convert funds immediately at the current market price.

Get Available Swap Directions

Returns the list of swap directions available to the authenticated user.

```
GET /swap-directions
```

Required Permissions

- `allow_swap`

Rate Limit

30 requests per minute

Response


```
[
  {
    "code": "BTC-USDT",
    "spend_currency": "BTC",
    "receive_currency": "USDT",
    "is_enabled": true,
    "spend_precision": 8,
    "receive_precision": 6,
    "fee_percent": "0.25",
    "min_spend_amount": "0.00100000",
    "max_spend_amount": "10.00000000",
    "min_receive_amount": "29.485250",
    "max_receive_amount": "294852.500000",
    "spend_rate_value": "29485.25",
    "receive_rate_value": "1.00000000"
  },
  {
    "code": "USDT-BTC",
    "spend_currency": "USDT",
    "receive_currency": "BTC",
    "is_enabled": true,
    "spend_precision": 6,
    "receive_precision": 8,
    "rate": "0.00003390",
    "fee_percent": "0.25",
    "min_spend_amount": "30.000000",
    "max_spend_amount": "300000.000000",
    "min_receive_amount": "0.00101700",
    "max_receive_amount": "10.17000000",
    "spend_rate_value": "29485.25",
    "receive_rate_value": "1.00000000"
  }
]
```

Response Fields

Field	Type	Description
`code`	string	Unique code for the swap direction
`spend_currency`	string	Currency code of the currency to spend
`receive_currency`	string	Currency code of the currency to receive
`is_enabled`	boolean	Whether the swap direction is currently enabled
`spend_precision`	integer	Decimal precision for the spend amount
`receive_precision`	integer	Decimal precision for the receive amount
`fee_percent`	string	Percentage fee for the swap
`min_spend_amount`	string	Minimum amount that can be spent
`max_spend_amount`	string	Maximum amount that can be spent
`min_receive_amount`	string	Minimum amount that can be received
`max_receive_amount`	string	Maximum amount that can be received
`spend_rate_value`	string	Amount of spend currency for exchange
`receive_rate_value`	string	Amount of receive currency for exchange

Create Swap

Initiates a swap between two currencies.

```
POST /swap/open
```

Required Permissions

- `allow_swap`

Rate Limit

300 requests per minute

Request Parameters

Parameter	Type	Required	Description
`direction`	string	Yes	Code of the swap direction (see Get Available Swap Directions)
`spend_amount`	string	No*	Amount to spend (*Either spend_amount or receive_amount must be provided)
`receive_amount`	string	No*	Amount to receive (*Either spend_amount or receive_amount must be provided)
`callback_url`	string	No	URL that will be called with status updates
`custom_id`	string	No	Custom identifier for tracking this swap
`time_to_live`	integer	No	Validity period in milliseconds (default: 30000)

Example Request

```
{
  "direction": "BTC-USDT",
  "spend_amount": "0.01000000",
  "callback_url": "https://example.com/callbacks/swaps",
  "custom_id": "my-swap-123",
  "time_to_live": 30000
}
```

Response

```
{
  "code": "SWP123456",
  "custom_id": "my-swap-123",
  "callback_url": "https://example.com/callbacks/swaps",
  "direction": "BTC-USDT",
  "spend_currency": "BTC",
  "receive_currency": "USDT",
  "spend_amount": "0.01000000",
  "receive_amount": "294.85250000",
```

```
  "fee_amount": "0.73713125",
  "fee_currency": "USDT",
  "rate": "29485.25",
  "status": "open",
  "reason_code": null,
  "reason_message": null,
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:30:15Z",
  "done_at": null,
  "expires_at": "2023-01-15T14:30:45Z"
}
```


Response Fields

Field	Type	Description
`code`	string	Unique swap code
`custom_id`	string	Custom identifier provided in the request
`callback_url`	string	URL that will be called with status updates
`direction`	string	Swap direction code
`spend_currency`	string	Currency code of the currency to spend
`receive_currency`	string	Currency code of the currency to receive
`spend_amount`	string	Amount to spend
`receive_amount`	string	Amount to receive
`spend_rate`	string	Amount to spend
`receive_rate`	string	Amount to receive
`fee_amount`	string	Fee amount
`fee_currency`	string	Currency code of the fee
`spend_fee_filled`	string	Filled fee for spend currency
`receive_fee_filled`	string	Filled fee for receive currency
`spend_filled`	string	Filled amount for spend currency
`receive_filled`	string	Filled amount for receive currency
`status`	string	Current status of the swap
`reason_code`	string	Error code (if the swap failed)
`reason_message`	string	Error message (if the swap failed)
`created_at`	string	Creation timestamp (ISO 8601 format)
`updated_at`	string	Last update timestamp (ISO 8601 format)
`done_at`	string	Completion timestamp (ISO 8601 format)
`valid_till`	string	Expiration timestamp (ISO 8601 format)

Cancel Swap

Cancels a pending swap.

```
POST /swap/cancel
```

Required Permissions

- `allow_swap`

Rate Limit

300 requests per minute

Request Parameters

Parameter	Type	Required	Description
`code`	string	Yes*	Unique order code
`custom_id`	string	Yes*	Custom identifier

At least one of the parameters `code` or `custom_id` must be provided. If both are provided, the `code` parameter will be used first.

Example Request

```
{
  "code": "SWP123456"
}
```

Response

Same as the swap object. Status may not be changed to `cancelled` instantly, as the system may need time to process the request. Instead `cancel_requested_at` will be set to the current time.

List Swaps

Returns a list of swaps.

```
GET /swap
```

Required Permissions

- `allow_swap`

Rate Limit

300 requests per minute

Query Parameters

Parameter	Type	Required	Description
`status`	string	No	Filter by status (see Swap Status Values)
`direction`	string	No	Filter by direction (see Get Available Swap Directions)
`custom_id`	string	No	Filter by custom ID
`start_at`	string	No	Filter by start datetime on field `created_at` (inclusive, ISO 8601 format)
`end_at`	string	No	Filter by end datetime on field `created_at` (inclusive, ISO 8601 format)
`limit`	integer	No	Number of results to return (default: 100, max: 200)
`offset`	integer	No	Number of results to skip (for pagination)

Response


```
[
  {
    "code": "SWP123456",
    "custom_id": "my-swap-123",
    "callback_url": "https://example.com/callbacks/swaps",
    "direction": "BTC-USDT",
    "spend_currency": "BTC",
    "receive_currency": "USDT",
    "spend_rate": "1.0000000",
    "receive_rate": "29485.25000000",
    "spend_amount": "0.01000000",
    "receive_amount": "294.85250000",
    "amount_parameter": "spend",
    "fee_amount": "0.73713125",
    "fee_currency": "USDT",
    "spend_filled": "0.01000000",
    "receive_filled": "294.85250000",
    "spend_fee_filled": "0.00000000",
    "receive_fee_filled": "0.73713125",
    "status": "success",
    "reason_code": null,
    "reason_message": null,
    "created_at": "2023-01-15T14:30:15Z",
    "updated_at": "2023-01-15T14:35:20Z",
    "done_at": "2023-01-15T14:35:20Z",
    "valid_till": "2023-01-15T14:30:45Z",
    "cancel_requested_at": null
  }
```

```
  // More swaps...
]
```

Response Fields

Same as the create swap response. The results are returned as a direct array without pagination metadata.

Pagination

The API uses simple offset-based pagination with `limit` and `offset` parameters. Offset is limited for maximum 10000, if you set more - error will be returned. The default limit is 100, and the maximum limit is 200. The response is a direct list of swap objects without pagination metadata wrapper. If you need to fetch more than 200 swaps, you should use the `start_at` and `end_at` parameters to fetch swaps in smaller date ranges. List is sorted by creation date in descending order.

Get Swap Details

Returns details of a specific swap.

```
GET /swap/{code}
```

Required Permissions

- `allow_swap`

Rate Limit

300 requests per minute

Path Parameters

Parameter	Type	Required	Description
`code`	string	Yes	Unique swap code

Response

Same as a single swap object from the list endpoint.

Swap Status Values

Status	Description
`open`	Swap is open and being processed
`success`	Swap has been successfully completed
`failed`	Swap has failed (see reason_code for details)

Callback Notifications

If a `callback_url` was provided when creating the swap, the system will send POST requests to this URL when the swap status changes.

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

Callback Security

Currently, callbacks are **not signed**. The signature mechanism for callbacks is planned for a future update. You should validate callback data by making an authenticated API request to verify the swap status.

Callback Payload

The callback payload contains the same data as the swap details response, with essential fields for status updates:

```
{
  "code": "SWP123456",
  "custom_id": "my-swap-123",
  "callback_url": "https://example.com/callbacks/swaps",
  "direction": "BTC-USDT",
  "spend_currency": "BTC",
  "receive_currency": "USDT",
  "spend_rate": "1.0000000",
```

```
  "receive_rate": "29485.25000000",
  "spend_amount": "0.01000000",
  "receive_amount": "294.85250000",
  "amount_parameter": "spend",
  "fee_amount": "0.73713125",
  "fee_currency": "USDT",
  "spend_filled": "0.01000000",
  "receive_filled": "294.85250000",
  "spend_fee_filled": "0.00000000",
  "receive_fee_filled": "0.73713125",
  "status": "success",
  "reason_code": null,
  "reason_message": null,
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:35:20Z",
  "done_at": "2023-01-15T14:35:20Z",
  "valid_till": "2023-01-15T14:30:45Z",
  "cancel_requested_at": null
}
```


Securing Callbacks

Since callbacks are not currently signed, you should implement these security measures:

1. **Whitelist IPs**: Accept callbacks only from the service's IP addresses. Contact support to get the list of server IPs.
2. **Verify Data**: Always make an authenticated API request to `/swap/{code}` to verify the status and details of any swap after receiving a callback.
3. **Use HTTPS**: Ensure your callback URL uses HTTPS to encrypt the data in transit.

Common Use Cases

- **Currency conversion**: Convert one currency to another
- **Price discovery**: Get current exchange rates
- **Trade execution**: Execute trades at current market rates

March 21, 2025

March 19, 2025

3.6 Internal Transfer Endpoints

This section covers API endpoints related to internal transfers between users. Internal transfer operations allow users to send funds to other users within the platform, as well as receive funds from other platform users. These transfers are executed instantly without blockchain or external banking fees, providing a cost-effective way to move funds between platform accounts.

Create Internal Transfer

Initiates a transfer from the authenticated user to another user.

```
POST /internal-transfer
```

Required Permissions

- `allow_transfer`

Rate Limit

600 requests per minute

Request Parameters

Parameter	Type	Required	Description
`currency_code`	string	Yes	Currency code to transfer
`amount`	string	Yes	Amount to transfer
`recipient`	string	Yes	Recipient's user ID or email
`callback_url`	string	No	URL that will be called with status updates
`custom_id`	string	No	Custom identifier for tracking this transfer
`address_book_code`	string	No	Address book entry code to use for the transfer
`note`	string	No	Note about the transfer (visible to recipient)

Example Request

```
{
  "currency_code": "BTC",
  "amount": "0.01000000",
  "recipient": "user123",
  "callback_url": "https://example.com/callbacks/transfers",
  "custom_id": "my-transfer-123",
  "note": "Payment for services"
}
```

Response

```
{
  "code": "TRF123456",
  "custom_id": "my-transfer-123",
  "callback_url": "https://example.com/callbacks/transfers",
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:30:15Z",
  "done_at": null,
  "timestamp": "2023-01-15T14:30:15Z",
  "status": "pending",
  "reason_code": null,
  "reason_message": null,
  "currency_code": "BTC",
  "amount": "0.01000000",
  "fee_amount": "0.00000000",
  "sender": "user456",
  "recipient": "user123",
  "note": "Payment for services"
}
```

Response Fields

Field	Type	Description
`code`	string	Unique transfer code
`custom_id`	string	Custom identifier provided in the request
`callback_url`	string	URL that will be called with status updates
`created_at`	string	Creation timestamp (ISO 8601 format)
`updated_at`	string	Last update timestamp (ISO 8601 format)
`done_at`	string	Completion timestamp (ISO 8601 format)
`timestamp`	string	Operation timestamp (ISO 8601 format)
`status`	string	Current status of the transfer
`reason_code`	string	Error code (if the transfer failed)
`reason_message`	string	Error message (if the transfer failed)
`currency_code`	string	Currency code
`amount`	string	Transfer amount
`fee_amount`	string	Fee amount
`sender`	string	Sender's user ID
`recipient`	string	Recipient's user ID
`note`	string	Note about the transfer

List Internal Transfers

Returns a list of internal transfers.

```
GET /internal-transfer
```

Required Permissions

- `allow_transfer`

Rate Limit

600 requests per minute

Query Parameters

Parameter	Type	Required	Description
`status`	string	No	Filter by status (see Transfer Status Values)
`currency_code`	string	No	Filter by currency
`custom_id`	string	No	Filter by custom ID
`start_at`	string	No	Filter by start datetime on field `timestamp` (inclusive, ISO 8601 format)
`end_at`	string	No	Filter by end datetime on field `timestamp` (inclusive, ISO 8601 format)
`limit`	integer	No	Number of results to return (default: 100, max: 200)
`offset`	integer	No	Number of results to skip (for pagination)

Response

The API uses a standard offset-based pagination system with `limit` and `offset` parameters. The response is a direct array of transfer objects without pagination metadata wrapper.


```
[
  {
    "code": "TRF123456",
    "custom_id": "my-transfer-123",
    "callback_url": "https://example.com/callbacks/transfers",
    "created_at": "2023-01-15T14:30:15Z",
    "updated_at": "2023-01-15T14:35:20Z",
```

```
    "done_at": "2023-01-15T14:35:20Z",
    "timestamp": "2023-01-15T14:35:20Z",
    "status": "success",
    "reason_code": null,
    "reason_message": null,
    "currency_code": "BTC",
    "amount": "0.01000000",
    "fee_amount": "0.00000000",
    "sender": "user456",
    "recipient": "user123",
    "note": "Payment for services"
  }
  // More transfers...
]
```

Response Fields

Same as the create transfer response.

Pagination

The API uses simple offset-based pagination with `limit` and `offset` parameters. Offset is limited for maximum 10000, if you set more - error will be returned. The default limit is 100, and the maximum limit is 200. The response is a direct list of transfer objects without pagination metadata wrapper. If you need to fetch more than 200 transfers, you should use the `start_at` and `end_at` parameters to fetch transfers in smaller date ranges. List is sorted by creation date in descending order.

Get Transfer Details

Returns details of a specific transfer.

```
GET /internal-transfer/{code}
```

Required Permissions

- `allow_transfer`

Rate Limit

600 requests per minute

Path Parameters

Parameter	Type	Required	Description
`code`	string	Yes	Unique transfer code

Response

Same as a single transfer object from the list endpoint.

Transfer Status Values

Status	Description
`pending`	Transfer is pending execution
`success`	Transfer has been successfully completed
`rejected`	Transfer has been rejected (see reason_code for details)

Callback Notifications

If a `callback_url` was provided when creating the transfer, the system will send POST requests to this URL when the transfer status changes.

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

Callback Security

Currently, callbacks are **not signed**. The signature mechanism for callbacks is planned for a future update. You should validate callback data by making an authenticated API request to verify the transfer status.

Callback Payload

The callback payload contains the same data as the transfer details response, with essential fields for status updates:

```
{
  "code": "TRF123456",
  "custom_id": "my-transfer-123",
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:35:20Z",
  "done_at": "2023-01-15T14:35:20Z",
  "status": "completed",
  "reason_code": null,
  "reason_message": null,
  "currency_code": "BTC",
  "amount": "0.01000000",
  "fee_amount": "0.00000000",
  "sender": "user456",
  "recipient": "user123",
  "note": "Payment for services"
}
```

Securing Callbacks

Since callbacks are not currently signed, you should implement these security measures:

1. **Whitelist IPs**: Accept callbacks only from the service's IP addresses. Contact support to get the list of server IPs.
2. **Verify Data**: Always make an authenticated API request to `/internal-transfer/{code}` to verify the status and details of any transfer after receiving a callback.
3. **Use HTTPS**: Ensure your callback URL uses HTTPS to encrypt the data in transit.

Common Use Cases

- **B2B payments**: Transfer funds between business partners
- **User payments**: Transfer funds between your users
- **Fund management**: Move funds between your own accounts

March 21, 2025

March 19, 2025

3.7 Limit Order Endpoints

This section covers API endpoints related to limit orders. Limit order operations allow users to exchange currencies within the platform at a specific price set by the user. Unlike swaps, limit orders are not executed immediately but remain active until either the specified price condition is met and the order executes, or until the user cancels the order. Limit orders are ideal for users who want to exchange at a specific target price rather than the current market rate.

> **Feature Activation Required**

To use limit order features, you need to request this feature to be enabled for your account by contacting support. Once enabled, you must also ensure that your API keys have the appropriate `allow_limit_order` permission. All API features require specific permissions which you can select when creating or updating API keys.

Get Available Trading Pairs

Returns the list of trading pairs available to the authenticated user.

```
GET /limit-pairs
```

Required Permissions

- `allow_limit_order`

Rate Limit

30 requests per minute

Response


```
[
  {
    "symbol": "BTCUSDT",
    "base_currency": "BTC",
    "quote_currency": "USDT",
    "price_precision": 2,
    "amount_precision": 6,
    "total_precision": 8,
    "min_base_amount": "0.000100",
    "max_base_amount": "10.000000",
    "min_quote_value": "10.00",
    "max_quote_value": "1000000.00",
    "is_enabled": true,
    "fee_percent": "0.25",
    "latest_price": "29485.25"
  },
  {
    "symbol": "ETHUSDT",
    "base_currency": "ETH",
    "quote_currency": "USDT",
    "price_precision": 2,
    "amount_precision": 5,
    "min_base_amount": "0.000100",
    "max_base_amount": "10.000000",
    "min_quote_value": "10.00",
    "max_quote_value": "1000000.00",
    "is_enabled": true,
    "fee_percent": "0.25",
    "latest_price": "1843.75"
  }
]
```

Response Fields

Field	Type	Description
`symbol`	string	Trading pair symbol (e.g., BTCUSDT)
`base_currency`	string	Base currency code
`quote_currency`	string	Quote currency code
`price_precision`	integer	Decimal precision for the price
`amount_precision`	integer	Decimal precision for the amount
`total_precision`	integer	Decimal precision for the quote amount (value)
`min_base_amount`	string	Minimum allowed amount
`max_base_amount`	string	Maximum allowed amount
`min_quote_value`	string	Minimum allowed value (amount * price)
`max_quote_value`	string	Maximum allowed value (amount * price)
`is_enabled`	bool	Trading is active
`fee_percent`	string	Trading fee percentage
`fee_deduction`	string	Algoruthm of fees deduction: `received` - from received currency; `quote` - always from quote currency (default)
`latest_price`	string	Current market price. It may be null when do data available

Create Limit Order

Places a new limit order.

```
POST /limit-order
```

Required Permissions

- `allow_limit_order`

Rate Limit

600 requests per minute

Request Parameters

Parameter	Type	Required	Description
`pair`	string	Yes	Trading pair symbol (e.g., BTCUSDT) (see Get Available Trading Pairs)
`side`	string	Yes	Order side: "buy" or "sell"
`limit_price`	string	No*	Limit price in quote currency
`spend_amount`	string	No*	Order amount that you going to spend for order
`receive_amount`	string	No*	Order amount that you want to receive for order (after fees)
`callback_url`	string	No	URL that will be called with status updates
`custom_id`	string	No	Custom identifier for tracking this order

Amounts and limit prices are optional, but two of three parameters must be provided: `limit_price`, `spend_amount`, or `receive_amount`. Choose combination most suitable for your use case. Also note that `spent_amount` and `receive_amount` express different currencies depending on order `side`. So is you set both amounts - limit price will be calculated automatically based on those amounts and fees.

Placed order will be in `open` status until it is filled or cancelled by user or system in rare cases.

Example Request

```
{
  "pair": "BTCUSDT",
  "side": "buy",
  "receive_amount": "0.01000000",
  "limit_price": "29000.00",
  "callback_url": "https://example.com/callbacks/orders",
```

```
  "custom_id": "my-order-123"
}
```

Response

```
{
  "code": "LMT123456",
  "custom_id": "my-order-123",
  "callback_url": "https://example.com/callbacks/orders",
  "pair": "BTCUSDT",
  "side": "buy",
  "receive_amount": "0.01000000",
  "limit_price": "29000.00",
  "spend_amount": "290.00",
  "base_amount": "0.01000000",
  "quote_amount": "290.00",
  "filled_base_amount": "0.00000000",
  "filled_quote_amount": "0.00",
  "filled_fee_amount": "0.00000000",
  "fee_currency": "USDT",
  "status": "open",
  "reason_code": null,
  "reason_message": null,
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:30:15Z",
  "done_at": null
}
```

Response Fields

Field	Type	Description
`code`	string	Unique order code
`custom_id`	string	Custom identifier provided in the request
`callback_url`	string	URL that will be called with status updates
`pair`	string	Trading pair symbol
`side`	string	Order side: "buy" or "sell"
`limit_price`	string	Limit price in quote currency
`spend_amount`	string	Order amount reserved to spend
`receive_amount`	string	Order amount expected after execution
`base_amount`	string	Base amount of order
`quote_amount`	string	Quote amount of order
`filled_base_amount`	string	Filled amount in base currency
`filled_quote_amount`	string	Filled value in quote currency
`filled_fee_amount`	string	Fee amount
`fee_currency`	string	Fee currency code
`status`	string	Current status of the order
`reason_code`	string	Error code (if the order failed)
`reason_message`	string	Error message (if the order failed)
`created_at`	string	Creation timestamp (ISO 8601 format)
`updated_at`	string	Last update timestamp (ISO 8601 format)
`done_at`	string	Completion timestamp (ISO 8601 format)

Cancel Limit Order

Cancels an open limit order.

```
POST /limit-order/cancel
```

Required Permissions

- `allow_limit_order`

Rate Limit

600 requests per minute

Request Parameters

Parameter	Type	Required	Description
`code`	string	Yes*	Unique order code
`custom_id`	string	Yes*	Custom identifier

At least one of the parameters `code` or `custom_id` must be provided. If both are provided, the `code` parameter will be used first.

Example Request

```
{
  "code": "LMT123456"
}
```

Response

Same as the order object, with status updated to `cancelled`.

List Limit Orders

Returns a list of limit orders.

```
GET /limit-order
```

Required Permissions

- `allow_limit_order`

Rate Limit

600 requests per minute

Query Parameters

Parameter	Type	Required	Description
`status`	string	No	Filter by status (see Order Status Values)
`pair`	string	No	Filter by trading pair (see Get Available Trading Pairs)
`side`	string	No	Filter by side (buy or sell)
`custom_id`	string	No	Filter by custom ID
`start_at`	string	No	Filter by start datetime on field `created_at` (inclusive, ISO 8601 format)
`end_at`	string	No	Filter by end datetime on field `created_at` (inclusive, ISO 8601 format)
`limit`	integer	No	Number of results to return (default: 100, max: 200)
`offset`	integer	No	Number of results to skip (for pagination)

Response

```
[
  {
    "code": "LMT123456",
    "custom_id": "my-order-123",
    "callback_url": "https://example.com/callbacks/orders",
    "pair": "BTCUSDT",
    "side": "buy",
    "receive_amount": "0.01000000",
    "limit_price": "29000.00",
    "spend_amount": "290.00",
    "base_amount": "0.01000000",
    "quote_amount": "290.00",
    "filled_base_amount": "0.00000000",
    "filled_quote_amount": "0.00",
    "filled_fee_amount": "0.00000000",
    "fee_currency": "USDT",
    "status": "open",
    "reason_code": null,
    "reason_message": null,
    "created_at": "2023-01-15T14:30:15Z",
```

```
    "updated_at": "2023-01-15T14:30:15Z",
    "done_at": null
  }
  // More orders...
]
```

Response Fields

Same as the creation order response. The results are returned as a direct array without pagination metadata.

Pagination

The API uses simple offset-based pagination with `limit` and `offset` parameters. Offset is limited for maximum 10000, if you set more - error will be returned. The default limit is 100, and the maximum limit is 200. The response is a direct list of limit order objects without pagination metadata wrapper. If you need to fetch more than 200 limit orders, you should use the `start_at` and `end_at` parameters to fetch limit orders in smaller date ranges. List is sorted by creation date in descending order.

Get Order Details

Returns details of a specific order.

```
GET /limit-order/{code}
```

Required Permissions

- `allow_limit_order`

Rate Limit

600 requests per minute

Path Parameters

Parameter	Type	Required	Description
`code`	string	Yes	Unique order code

Response

Same as a single order object from the list endpoint.

Order Status Values

Status	Description
`queued`	Order is queued, awaiting to be placed on exchange
`open`	Order is placed on the exchange
`cancelling`	Order cancellation has been requested
`success`	Order has been successfully completed, balance changed
`failed`	Order has failed or been rejected (see reason_code for details)

Callback Notifications

If a `callback_url` was provided when creating the order, the system will send POST requests to this URL when the order status changes.

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

Callback Security

Currently, callbacks are **not signed**. The signature mechanism for callbacks is planned for a future update. You should validate callback data by making an authenticated API request to verify the order status.

Callback Payload

The callback payload contains the same data as the order details response, with essential fields for status updates:


```
{
  "code": "LMT123456",
  "custom_id": "my-order-123",
  "callback_url": "https://example.com/callbacks/orders",
  "pair": "BTCUSDT",
  "side": "buy",
  "receive_amount": "0.01000000",
  "limit_price": "29000.00",
  "spend_amount": "290.00",
  "base_amount": "0.01000000",
  "quote_amount": "290.00",
  "filled_base_amount": "0.00000000",
  "filled_quote_amount": "0.00",
  "filled_fee_amount": "0.00000000",
  "fee_currency": "USDT",
  "status": "open",
  "reason_code": null,
  "reason_message": null,
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:30:15Z",
  "done_at": null
}
```


Securing Callbacks

Since callbacks are not currently signed, you should implement these security measures:

1. **Whitelist IPs**: Accept callbacks only from the service's IP addresses. Contact support to get the list of server IPs.
2. **Verify Data**: Always make an authenticated API request to `/limit-order/{code}` to verify the status and details of any order after receiving a callback.
3. **Use HTTPS**: Ensure your callback URL uses HTTPS to encrypt the data in transit.

Common Use Cases

- **Trading strategy execution**: Implement buy/sell signals from trading strategies
- **Arbitrage**: Place orders on multiple exchanges
- **Market making**: Provide liquidity by placing buy and sell orders
- **Dollar-cost averaging**: Place regular buy orders at predetermined prices

March 21, 2025

March 19, 2025

3.8 AML Check Endpoints

This section covers API endpoints related to Anti-Money Laundering (AML) checks. AML checks are designed for verifying the risk profile and compliance status of cryptocurrency addresses or transactions through paid third-party analysis services. These checks help meet regulatory requirements and mitigate fraud risk by analyzing blockchain data.

> **Feature Activation Required**

To use AML check features, you need to request this feature to be enabled for your account by contacting support. Once enabled, you must also ensure that your API keys have the appropriate `allow_aml_check` permission. All API features require specific permissions which you can select when creating or updating API keys.

Get Available AML Plans

Returns the list of available AML check plans. Each plan record represents the price for a single AML check. Currently, only the "check" type is supported.

When a check is initiated, the plan price is deducted from the user's balance in the specified currency. Users can buy packs of checks at a lower price per check, which is why plans are priced individually.

```
GET /aml/plan
```

Required Permissions

- `allow_aml_check`

Rate Limit

600 requests per minute

Response

```
[
  {
    "code": "amlcheck",
    "name": "AML Check",
    "description": "Standard AML check using AMLCHECK token",
    "check_type": "check",
    "currency": "AMLCHECK",
    "price": "1.00"
  },
  {
    "code": "usdt",
    "name": "AML Check (USDT)",
    "description": "Standard AML check using USDT",
    "check_type": "check",
    "currency": "USDT",
    "price": "1.00"
  }
]
```

Response Fields

Field	Type	Description
`code`	string	Unique plan code
`name`	string	Plan name
`currency`	string	Currency code for the plan price (AMLCHECK or USDT)
`price`	string	Price per check (1.00 for both available plans)
`check_type`	string	Type of check (currently only "check" is supported)

Get Available AML Tokens

Returns the list of available tokens (blockchain networks) for AML checks.

```
GET /aml/token
```

Required Permissions

- `allow_aml_check`

Rate Limit

120 requests per minute

Response

```
[
  {
    "id": 1,
    "symbol": "BTC",
    "name": "Bitcoin",
    "network": "bitcoin",
    "precision": 8
  },
  {
    "id": 2,
    "symbol": "ETH",
    "name": "Ethereum",
    "network": "ethereum",
    "precision": 18
  },
  {
    "id": 3,
    "symbol": "USDT",
    "name": "Tether USD ERC20",
    "network": "ethereum",
    "precision": 6
  }
]
```

Response Fields

Field	Type	Description
`id`	integer	Unique token ID
`symbol`	string	Token symbol
`name`	string	Token name
`network`	string	Network code
`precision`	integer	Decimal precision

Create AML Check

Initiates an AML check for an address or transaction.

```
POST /aml/check
```

Required Permissions

- `allow_aml_check`

Rate Limit

600 requests per minute

Request Parameters

Parameter	Type	Required	Description
`direction`	string	Yes	Direction for transaction check: "deposit" or "withdrawal"
`token`	integer	Yes	Token ID (see Get Available AML Tokens)
`address`	string	Yes	Address to check. For deposit direction is the address that receive funds. For withdrawals directions is the address that receive finds to.
`transaction`	string	No*	Transaction hash to check. For withdrawal direction is not mandatory, but for deposit direction it is.
`callback_url`	string	No	URL that will be called with status updates
`custom_id`	string	No	Custom identifier for tracking this check. It also implements idempotency pattern. If set in requests should be unique per user.

Example Request for Address Check

```
{
  "direction": "withdrawal",
  "token": 1,
```

```
  "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
  "callback_url": "https://example.com/callbacks/aml",
  "custom_id": "my-aml-check-123"
}
```

Example Request for Transaction Check

```
{
  "direction": "deposit",
  "token": 1,
  "transaction": "7b5968e39e0d4c980a234931b8a2fd21e804eb7e9f7eb5af75b3d89e2972ad3d",
  "callback_url": "https://example.com/callbacks/aml",
  "custom_id": "my-aml-check-456"
}
```

Response

```
{
  "code": "AML123456",
  "custom_id": "my-aml-check-123",
  "callback_url": "https://example.com/callbacks/aml",
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:30:15Z",
  "plan": "amlcheck",
  "token": 1,
  "currency": "AMLCHECK",
  "price": "1.00",
  "status": "pending",
  "reason_code": null,
  "reason_message": null,
  "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
  "transaction": null,
  "direction": "deposit",
  "report": null,
  "refund_on_fail": true
}
```

Response Fields

Field	Type	Description
`code`	string	Unique check code
`custom_id`	string	Custom identifier provided in the request
`callback_url`	string	URL that will be called with status updates
`created_at`	string	Creation timestamp (ISO 8601 format)
`updated_at`	string	Last update timestamp (ISO 8601 format)
`plan`	string	Plan code
`token`	integer	Token ID
`currency`	string	Currency code for the plan price
`price`	string	Price for this check
`status`	string	Current status of the check
`reason_code`	string	Error code (if the check failed)
`reason_message`	string	Error message (if the check failed)
`address`	string	Address being checked (if provided)
`transaction`	string	Transaction being checked (if provided)
`direction`	string	Direction for transaction check
`report`	object	Check report (null until completed)
`refund_on_fail`	boolean	Whether to refund the check price if it fails

List AML Checks

Returns a list of AML checks.

```
GET /aml/check
```

Required Permissions

- `allow_aml_check`

Rate Limit

600 requests per minute

Query Parameters

Parameter	Type	Required	Description
`status`	string	No	Filter by status (see AML Check Status Values)
`plan`	string	No	Filter by plan (see Get Available AML Plans)
`token`	integer	No	Filter by token ID (see Get Available AML Tokens)
`address`	string	No	Filter by exact address value
`transaction`	string	No	Filter by exact transaction hash value
`code`	string	No	Filter by internal code value
`custom_id`	string	No	Filter by custom ID
`start_at`	string	No	Filter by start datetime on field `created_at` (inclusive, ISO 8601 format)
`end_at`	string	No	Filter by end datetime on field `created_at` (inclusive, ISO 8601 format)
`limit`	integer	No	Number of results to return (default: 100, max: 200)
`offset`	integer	No	Number of results to skip (for pagination)

Response


```
[
  {
    "code": "AML123456",
    "custom_id": "my-aml-check-123",
    "callback_url": "https://example.com/callbacks/aml",
    "created_at": "2023-01-15T14:30:15Z",
    "updated_at": "2023-01-15T14:35:20Z",
    "plan": "amlcheck",
    "token": 1,
    "currency": "AMLCHECK",
    "price": "1.00",
    "status": "success",
    "reason_code": null,
    "reason_message": null,
    "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
    "transaction": null,
    "direction": "deposit",
    "report": {
      "direction": "deposit",
      "status": "ready",
      "alert_grade": "low",
      "riskscore": 0.1,
      "signals": {
        "exchanger": 0.1,
        "mixer": 0.0,
        "darknet": 0.0,
        "scam": 0.0,
        "gambling": 0.0
      },
      "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
      "currency": "BTC"
    },
    "refund_on_fail": true
  }
  // More checks...
]
```


Response Fields

ÏSame as the create check response, with the addition of report object.

Pagination

The API uses simple offset-based pagination with `limit` and `offset` parameters. Offset is limited for maximum 10000, if you set more - error will be returned. The default limit is 100, and the maximum limit is 200. The response is a direct list of AML check objects without pagination metadata wrapper. If you need to fetch more than 200 checks, you should use the `start_at` and `end_at` parameters to fetch checks in smaller date ranges. List is sorted by creation date in descending order.

Report Object

The result field includes the complete report from the AML check provider:

Field	Type	Description
`currency`	string	Blockchain currency code
`token_id`	int	Token id on AML system
`direction`	string	Direction of the transaction (deposit/withdrawal)
`address`	string	Blockchain address being checked
`tx`	string	Transaction hash/ID (for transaction checks)
`time`	int	Timestamp of report
`fiat`	float	Fiat equivalent of transfer amount
`amount`	float	Transfer amount
`riskscore`	float	Risk score as a value between 0 and 1. For deposit transfers, it is the average Risk Score of the transaction inputs, for withdrawal addresses it is the Risk Score of the specified address.
`signals`	object	Dictionary of signal codes and their values
`risky_volume`	float	Amount of risk funds of transfer
`risky_volume_fiat`	float	Fiat equivalent of amount of risk funds
`fiat_code_effective`	string	Fiat currency code
`created_at`	int	Unix timestamp when report created
`updated_at`	int	Unix timestamp when report updated
`is_pool`	bool	Indicates if transfer address is from pool
`status`	string	Status of report. One of: ready, updating, pending, missed, failed, new. Report useful when status is ready.
`counterparty`	object	Details of counterparty entity of transfer
`alert_grade`	string	medium, high, severe
`alert_type`	string	counterparty, pattern, amount, precheck, iscript and other possible

Signals Object

Signals object represents detailed information about the risk factors of the address or transaction. Known signals is: atm, dark_market, dark_service, exchange_fraudulent, exchange_mlrisk_high, exchange_mlrisk_low, exchange_mlrisk_moderate, exchange_mlrisk_veryhigh, gambling, ransom, marketplace, miner, mixer, p2p_exchange_mlrisk_high, p2p_exchange_mlrisk_low, payment, scam, wallet, stolen_coins, illegal_service, terrorism_financing, child_exploitation, seized_assets, sanctions, liquidity_pools. Note that more signals may appear in the future.

Counterparty Object

This object represents known entity that is counterparty for address in transaction. All fields are optional and may not be present en reports.

Field	Type	Description
`address`	string	If an address does not belong to an entity, we display only the address hash in the counterparty
`type`	string	Entity type. For example `exchange_mlrisk_high`
`name`	string	Entity name
`slug`	string	Entity subtype (if available)
`signals`	object	Contains two sub-objects with signals.
`signals.bwd`	object	Signals details for received funds.
`signals.fwd`	object	Details for sent funds.

Interpretation of Risk Score

The risk score is a value between 0 and 1, where 0 means no risk and 1 means high risk. The risk score is calculated by the AML provider based on various factors, including the presence of known risk signals, the amount of funds involved, and the counterparty's risk profile.

Your system can use the risk score to determine the appropriate action for the address or transaction. For example, you might decide to block a transaction with a high risk score or flag an account for manual review.

Get AML Check Details

Returns details of a specific AML check.

```
GET /aml/check/{code}
```

Required Permissions

- `allow_aml_check`

Rate Limit

600 requests per minute

Path Parameters

Parameter	Type	Required	Description
`code`	string	Yes	Unique check code

Response

Same as a single check object from the list endpoint.

AML Check Status Values

Status	Description
`queued`	Check is queued for execution
`pending`	Check is pending execution
`success`	Check has been successfully completed
`failed`	Check has failed (see reason_code for details)

Callback Notifications

If a `callback_url` was provided when creating the check, the system will send POST requests to this URL when the check status changes.

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

Callback Security

Currently, callbacks are **not signed**. The signature mechanism for callbacks is planned for a future update. You should validate callback data by making an authenticated API request to verify the check status.

Callback Payload

The callback payload contains the same data as the check details response, with essential fields for status updates:


```
{
  "code": "AML123456",
  "custom_id": "my-aml-check-123",
  "created_at": "2023-01-15T14:30:15Z",
  "updated_at": "2023-01-15T14:35:20Z",
  "status": "success",
  "reason_code": null,
  "reason_message": null,
  "plan": "amlcheck",
  "token": 1,
  "currency": "AMLCHECK",
  "price": "1.00",
  "report": {
    "direction": "deposit",
    "status": "READY",
    "alert_grade": "low",
    "riskscore": 0.1,
    "signals": {
      "exchanger": 0.1,
      "mixer": 0.0,
      "darknet": 0.0,
```

```
      "scam": 0.0,
      "gambling": 0.0
    },
    "address": "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh",
    "currency": "BTC"
  }
}
```

Securing Callbacks

Since callbacks are not currently signed, you should implement these security measures:

1. **Whitelist IPs**: Accept callbacks only from the service's IP addresses. Contact support to get the list of server IPs.
2. **Verify Data**: Always make an authenticated API request to `/aml/check/{code}` to verify the status and details of any check after receiving a callback.
3. **Use HTTPS**: Ensure your callback URL uses HTTPS to encrypt the data in transit.

Common Use Cases

- **Customer verification**: Check the risk level of customer source addresses
- **Transaction monitoring**: Assess the risk level of incoming and outgoing transactions
- **Regulatory compliance**: Comply with AML/CTF regulations
- **Risk management**: Identify and mitigate risks associated with cryptocurrency transactions

March 21, 2025

March 19, 2025

4. Examples

4.1 Python Integration Examples

This section provides examples of how to integrate with the API using Python.

Authentication

```python
import hashlib
import hmac
import time
import requests
import json

class ApiClient:
    def __init__(self, base_url, access_key, secret_key):
        self.base_url = base_url.rstrip('/')
        self.access_key = access_key
        self.secret_key = secret_key

    def _get_timestamp(self):
        return str(int(time.time() * 1000))

    def _generate_signature(self, path, timestamp, body=''):
        message = f"{self.access_key}{path}{timestamp}{body}"
        signature = hmac.new(
            self.secret_key.encode('utf-8'),
            message.encode('utf-8'),
            hashlib.sha256
        ).hexdigest()
        return signature

    def request(self, method, endpoint, params=None, data=None):
        path = f"/api/v1{endpoint}"
        url = f"{self.base_url}{path}"

        timestamp = self._get_timestamp()

        headers = {
            'X-Access-Key': self.access_key,
            'X-Timestamp': timestamp,
        }

        if method.upper() in ['POST', 'PUT', 'PATCH'] and data is not None:
            body = json.dumps(data)
            headers['Content-Type'] = 'application/json'
        else:
            body = ''

        headers['X-Signature'] = self._generate_signature(path, timestamp, body)

        response = requests.request(
            method=method,
            url=url,
            params=params,
            data=body if body else None,
            headers=headers
        )

        return response.json()

    def get(self, endpoint, params=None):
        return self.request('GET', endpoint, params=params)

    def post(self, endpoint, data=None):
        return self.request('POST', endpoint, data=data)
```

Usage Examples

Check Server Status

```python
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

# Status endpoint doesn't require authentication
status = requests.get('https://api.001k.world/api/v1/status').json()
print(f"Server timestamp: {status['timestamp']}")
```

Get Account Balances

```python
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

balances = api.get('/balance')
for balance in balances:
    print(f"{balance['currency']}: {balance['available']} (available), {balance['frozen']} (frozen)")
```

Generate Deposit Address

```python
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

response = api.post('/deposit/generate_address', {
    'currency': 'BTC',
    'method': 'btc_native',
    'callback_url': 'https://your-server.com/callbacks/deposits',
```

```
    'custom_id': 'my-deposit-address-123'
})

print(f"Deposit address: {response['address']}")
if response.get('tag'):
    print(f"Tag: {response['tag']}")
```

Create Withdrawal

```
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

response = api.post('/withdraw', {
    'currency': 'BTC',
    'method': 'btc_native',
    'amount': '0.01000000',
    'address': 'bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh',
    'callback_url': 'https://your-server.com/callbacks/withdrawals',
    'custom_id': 'my-withdrawal-123'
})

print(f"Withdrawal created: {response['code']}")
print(f"Status: {response['status']}")
```

Create Swap

```
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

# First, get available swap directions
directions = api.get('/swap-directions')
for direction in directions:
    print(f"{direction['code']}: {direction['spend_currency']} -> {direction['receive_currency']} (rate: {direction['rate']})")

# Create a swap
response = api.post('/swap/open', {
    'direction': 'BTC-USDT',
    'spend_amount': '0.01000000',
    'callback_url': 'https://your-server.com/callbacks/swaps',
    'custom_id': 'my-swap-123'
})

print(f"Swap created: {response['code']}")
print(f"Spend: {response['spend_amount']} {response['spend_currency']}")
print(f"Receive: {response['receive_amount']} {response['receive_currency']}")
print(f"Status: {response['status']}")
```

Create Internal Transfer

```
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

response = api.post('/internal-transfer', {
    'currency_code': 'BTC',
    'amount': '0.01000000',
    'recipient': 'user123',
    'callback_url': 'https://your-server.com/callbacks/transfers',
    'custom_id': 'my-transfer-123',
    'note': 'Payment for services'
})

print(f"Transfer created: {response['code']}")
print(f"Status: {response['status']}")
```

Create Limit Order

```
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

# First, get available trading pairs
pairs = api.get('/limit-pairs')
for pair in pairs:
    print(f"{pair['symbol']}: {pair['current_price']} (fee: {pair['fee_percent']}%)")

# Create a limit order
response = api.post('/limit-order', {
    'pair': 'BTCUSDT',
    'side': 'buy',
    'receive_amount': '0.01000000',
    'limit_price': '29000.00',
    'callback_url': 'https://your-server.com/callbacks/orders',
    'custom_id': 'my-order-123'
})

print(f"Order created: {response['code']}")
print(f"Status: {response['status']}")
print(f"Base amount: {response['base_amount']}")
print(f"Quote amount: {response['quote_amount']}")
print(f"Filled base amount: {response['filled_base_amount']}")
print(f"Filled quote amount: {response['filled_quote_amount']}")
print(f"Fee currency: {response['fee_currency']}")
print(f"Filled fee amount: {response['filled_fee_amount']}")
print(f"Created at: {response['created_at']}")
print(f"Updated at: {response['updated_at']}")
print(f"Done at: {response.get('done_at', 'N/A')}")
```

Create AML Check

```
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

# First, get available plans and tokens
plans = api.get('/aml/plan')
```

```
tokens = api.get('/aml/token')

for plan in plans:
    print(f"Plan: {plan['code']} - {plan['name']} (price: {plan['price']} {plan['currency']})")

for token in tokens:
    print(f"Token: {token['symbol']} on {token['network']} (ID: {token['id']})")

# Create an AML check
response = api.post('/aml/check', {
    'direction': 'withdrawal',
    'token': 1,  # Bitcoin
    'address': 'bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh',
    'callback_url': 'https://your-server.com/callbacks/aml',
    'custom_id': 'my-aml-check-123'
})

print(f"AML check created: {response['code']}")
print(f"Status: {response['status']}")
print(f"Direction: {response['direction']}")
print(f"Token: {response['token']}")
print(f"Address: {response['address']}")
print(f"Transaction: {response.get('transaction', 'N/A')}")
print(f"Created at: {response['created_at']}")
print(f"Updated at: {response['updated_at']}")
if response.get('report'):
    print(f"Report status: {response['report']['status']}")
    print(f"Risk score: {response['report']['riskscore']}")
    print(f"Alert grade: {response['report']['alert_grade']}")
```

Callback Handling

When the status of an operation changes, the API will send a callback to the URL provided in the `callback_url` parameter. Here's an example of how to handle these callbacks in a Flask application:

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

```
from flask import Flask, request, jsonify
import requests
from functools import wraps

app = Flask(__name__)

# Your API client
api = ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key')

# IP whitelist for callback security
ALLOWED_IPS = ['192.168.1.1', '10.0.0.1']  # Replace with actual IPs provided by support

def verify_ip(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        client_ip = request.remote_addr
        if client_ip not in ALLOWED_IPS:
            return jsonify({'error': 'Unauthorized IP'}), 403
        return f(*args, **kwargs)
    return decorated_function

def verify_data(endpoint, code):
    """Verify callback data by making an API request to get the current status"""
    try:
        # Make an authenticated API request to verify the status
        response = api.get(f'{endpoint}/{code}')
        return response
    except Exception as e:
        # Handle any API request errors
        print(f"Error verifying data: {e}")
        return None

@app.route('/callbacks/deposits', methods=['POST'])
@verify_ip
def deposit_callback():
    data = request.json
    code = data.get('code')
    status = data.get('status')
    custom_id = data.get('custom_id')

    print(f"Deposit callback received: {code} ({custom_id}) - Status: {status}")

    # Verify the data by making an API request
    verified_data = verify_data('/deposit', code)
    if verified_data and verified_data.get('status') == status:
        # Process the callback (update database, send notifications, etc.)
        print("Deposit status verified")
    else:
        print("Warning: Callback data verification failed!")

    return jsonify({'success': True})

@app.route('/callbacks/withdrawals', methods=['POST'])
@verify_ip
def withdrawal_callback():
    data = request.json
    code = data.get('code')
    status = data.get('status')
    custom_id = data.get('custom_id')

    print(f"Withdrawal callback received: {code} ({custom_id}) - Status: {status}")

    # Verify the data by making an API request
    verified_data = verify_data('/withdraw', code)
    if verified_data and verified_data.get('status') == status:
        # Process the callback
        print("Withdrawal status verified")
    else:
```

```
        print("Warning: Callback data verification failed!")

    return jsonify({'success': True})

@app.route('/callbacks/swaps', methods=['POST'])
@verify_ip
def swap_callback():
    data = request.json
    code = data.get('code')
    status = data.get('status')
    custom_id = data.get('custom_id')

    print(f"Swap callback received: {code} ({custom_id}) - Status: {status}")

    # Verify the data by making an API request
    verified_data = verify_data('/swap', code)
    if verified_data and verified_data.get('status') == status:
        # Process the callback
        print("Swap status verified")
    else:
        print("Warning: Callback data verification failed!")

    return jsonify({'success': True})

@app.route('/callbacks/transfers', methods=['POST'])
@verify_ip
def transfer_callback():
    data = request.json
    code = data.get('code')
    status = data.get('status')
    custom_id = data.get('custom_id')

    print(f"Transfer callback received: {code} ({custom_id}) - Status: {status}")

    # Verify the data by making an API request
    verified_data = verify_data('/internal-transfer', code)
    if verified_data and verified_data.get('status') == status:
        # Process the callback
        print("Transfer status verified")
    else:
        print("Warning: Callback data verification failed!")

    return jsonify({'success': True})

@app.route('/callbacks/orders', methods=['POST'])
@verify_ip
def order_callback():
    data = request.json
    code = data.get('code')
    status = data.get('status')
    custom_id = data.get('custom_id')

    print(f"Order callback received: {code} ({custom_id}) - Status: {status}")

    # Verify the data by making an API request
    verified_data = verify_data('/limit-order', code)
    if verified_data and verified_data.get('status') == status:
        # Process the callback
        print("Order status verified")
    else:
        print("Warning: Callback data verification failed!")

    return jsonify({'success': True})

@app.route('/callbacks/aml', methods=['POST'])
@verify_ip
def aml_callback():
    data = request.json
    code = data.get('code')
    status = data.get('status')
    custom_id = data.get('custom_id')

    print(f"AML check callback received: {code} ({custom_id}) - Status: {status}")

    # Verify the data by making an API request
    verified_data = verify_data('/aml/check', code)
    if verified_data and verified_data.get('status') == status:
        # Process the callback
        print("AML check status verified")
    else:
        print("Warning: Callback data verification failed!")

    return jsonify({'success': True})

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

March 21, 2025

March 19, 2025

4.2 PHP Integration Examples

This section provides examples of how to integrate with the API using PHP.

Authentication

```php
<?php

class ApiClient {
    private $baseUrl;
    private $accessKey;
    private $secretKey;

    public function __construct($baseUrl, $accessKey, $secretKey) {
        $this->baseUrl = rtrim($baseUrl, '/');
        $this->accessKey = $accessKey;
        $this->secretKey = $secretKey;
    }

    private function getTimestamp() {
        return (string)(int)(microtime(true) * 1000);
    }

    private function generateSignature($path, $timestamp, $body = '') {
        $message = $this->accessKey . $path . $timestamp . $body;
        $signature = hash_hmac('sha256', $message, $this->secretKey);
        return $signature;
    }

    public function request($method, $endpoint, $params = [], $data = null) {
        $path = "/api/v1" . $endpoint;
        $url = $this->baseUrl . $path;

        $timestamp = $this->getTimestamp();

        $headers = [
            'X-Access-Key: ' . $this->accessKey,
            'X-Timestamp: ' . $timestamp
        ];

        $body = '';
        if (in_array(strtoupper($method), ['POST', 'PUT', 'PATCH']) && $data !== null) {
            $body = json_encode($data);
            $headers[] = 'Content-Type: application/json';
        }

        $headers[] = 'X-Signature: ' . $this->generateSignature($path, $timestamp, $body);

        $ch = curl_init();

        if (strtoupper($method) === 'GET' && !empty($params)) {
            $url .= '?' . http_build_query($params);
        }

        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

        switch (strtoupper($method)) {
            case 'POST':
                curl_setopt($ch, CURLOPT_POST, true);
                curl_setopt($ch, CURLOPT_POSTFIELDS, $body);
                break;
            case 'PUT':
                curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
                curl_setopt($ch, CURLOPT_POSTFIELDS, $body);
                break;
            case 'DELETE':
                curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'DELETE');
                break;
            default:
                // GET
                break;
        }

        $response = curl_exec($ch);
        $error = curl_error($ch);
        curl_close($ch);

        if ($error) {
            throw new Exception("cURL Error: $error");
        }

        return json_decode($response, true);
    }

    public function get($endpoint, $params = []) {
        return $this->request('GET', $endpoint, $params);
    }

    public function post($endpoint, $data = []) {
        return $this->request('POST', $endpoint, [], $data);
    }
}
```

Usage Examples

Check Server Status

```php
<?php

require_once 'ApiClient.php';
```

```php
// Status endpoint doesn't require authentication
$ch = curl_init('https://api.001k.world/api/v1/status');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
curl_close($ch);

$status = json_decode($response, true);
echo "Server timestamp: " . $status['timestamp'] . "\n";

// For authenticated endpoints
$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');
```

Get Account Balances

```php
<?php

require_once 'ApiClient.php';

$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

$balances = $api->get('/balance');
foreach ($balances as $balance) {
    echo $balance['currency'] . ': ' . $balance['available'] . ' (available), ' . $balance['frozen'] . " (frozen)\n";
}
```

Generate Deposit Address

```php
<?php

require_once 'ApiClient.php';

$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

$response = $api->post('/deposit/generate_address', [
    'currency' => 'BTC',
    'method' => 'btc_native',
    'callback_url' => 'https://your-server.com/callbacks/deposits',
    'custom_id' => 'my-deposit-address-123'
]);

echo "Deposit address: " . $response['address'] . "\n";
if (isset($response['tag'])) {
    echo "Tag: " . $response['tag'] . "\n";
}
```

Create Withdrawal

```php
<?php

require_once 'ApiClient.php';

$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

$response = $api->post('/withdraw', [
    'currency' => 'BTC',
    'method' => 'btc_native',
    'amount' => '0.01000000',
    'address' => 'bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh',
    'callback_url' => 'https://your-server.com/callbacks/withdrawals',
    'custom_id' => 'my-withdrawal-123'
]);

echo "Withdrawal created: " . $response['code'] . "\n";
echo "Status: " . $response['status'] . "\n";
```

Create Swap

```php
<?php

require_once 'ApiClient.php';

$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

// First, get available swap directions
$directions = $api->get('/swap-directions');
foreach ($directions as $direction) {
    echo $direction['code'] . ': ' . $direction['spend_currency'] . ' -> '
        . $direction['receive_currency'] . ' (rate: ' . $direction['rate'] . ")\n";
}

// Create a swap
$response = $api->post('/swap/open', [
    'direction' => 'BTC-USDT',
    'spend_amount' => '0.01000000',
    'callback_url' => 'https://your-server.com/callbacks/swaps',
    'custom_id' => 'my-swap-123'
]);

echo "Swap created: " . $response['code'] . "\n";
echo "Spend: " . $response['spend_amount'] . ' ' . $response['spend_currency'] . "\n";
echo "Receive: " . $response['receive_amount'] . ' ' . $response['receive_currency'] . "\n";
echo "Status: " . $response['status'] . "\n";
```

Create Internal Transfer

```php
<?php
```

```php
require_once 'ApiClient.php';

$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

$response = $api->post('/internal-transfer', [
    'currency_code' => 'BTC',
    'amount' => '0.01000000',
    'recipient' => 'user123',
    'callback_url' => 'https://your-server.com/callbacks/transfers',
    'custom_id' => 'my-transfer-123',
    'note' => 'Payment for services'
]);

echo "Transfer created: " . $response['code'] . "\n";
echo "Status: " . $response['status'] . "\n";
```

Create Limit Order

```php
<?php

require_once 'ApiClient.php';

$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

// First, get available trading pairs
$pairs = $api->get('/limit-pairs');
foreach ($pairs as $pair) {
    echo $pair['symbol'] . ': ' . $pair['current_price'] . ' (fee: ' . $pair['fee_percent'] . "%)\n";
}

// Create a limit order
$response = $api->post('/limit-order', [
    'pair' => 'BTCUSDT',
    'side' => 'buy',
    'receive_amount' => '0.01000000',
    'limit_price' => '29000.00',
    'callback_url' => 'https://your-server.com/callbacks/orders',
    'custom_id' => 'my-order-123'
]);

echo "Order created: " . $response['code'] . "\n";
echo "Status: " . $response['status'] . "\n";
echo "Base amount: " . $response['base_amount'] . "\n";
echo "Quote amount: " . $response['quote_amount'] . "\n";
echo "Filled base amount: " . $response['filled_base_amount'] . "\n";
echo "Filled quote amount: " . $response['filled_quote_amount'] . "\n";
echo "Fee currency: " . $response['fee_currency'] . "\n";
echo "Filled fee amount: " . $response['filled_fee_amount'] . "\n";
echo "Created at: " . $response['created_at'] . "\n";
echo "Updated at: " . $response['updated_at'] . "\n";
echo "Done at: " . ($response['done_at'] ?? 'N/A') . "\n";
```

Create AML Check

```php
<?php

require_once 'ApiClient.php';

$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

// First, get available plans and tokens
$plans = $api->get('/aml/plan');
$tokens = $api->get('/aml/token');

foreach ($plans as $plan) {
    echo "Plan: " . $plan['code'] . ' - ' . $plan['name']
        . ' (price: ' . $plan['price'] . ' ' . $plan['currency'] . ")\n";
}

foreach ($tokens as $token) {
    echo "Token: " . $token['symbol'] . ' on ' . $token['network']
        . ' (ID: ' . $token['id'] . ")\n";
}

// Create an AML check
$response = $api->post('/aml/check', [
    'direction' => 'withdrawal',
    'token' => 1,  // Bitcoin
    'address' => 'bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh',
    'callback_url' => 'https://your-server.com/callbacks/aml',
    'custom_id' => 'my-aml-check-123'
]);

echo "AML check created: " . $response['code'] . "\n";
echo "Status: " . $response['status'] . "\n";
echo "Direction: " . $response['direction'] . "\n";
echo "Token: " . $response['token'] . "\n";
echo "Address: " . $response['address'] . "\n";
echo "Transaction: " . ($response['transaction'] ?? 'N/A') . "\n";
echo "Created at: " . $response['created_at'] . "\n";
echo "Updated at: " . $response['updated_at'] . "\n";
if (isset($response['report'])) {
    echo "Report status: " . $response['report']['status'] . "\n";
    echo "Risk score: " . $response['report']['riskscore'] . "\n";
    echo "Alert grade: " . $response['report']['alert_grade'] . "\n";
}
```

Callback Handling

When the status of an operation changes, the API will send a callback to the URL provided in the `callback_url` parameter. Here's an example of how to handle these callbacks in a PHP application:

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

```php
<?php

require_once 'ApiClient.php';

// Your API client
$api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

// IP whitelist for callback security
$allowedIps = ['192.168.1.1', '10.0.0.1']; // Replace with actual IPs provided by support

// Verify client IP
function verifyIp() {
    global $allowedIps;
    $clientIp = $_SERVER['REMOTE_ADDR'];

    if (!in_array($clientIp, $allowedIps)) {
        http_response_code(403);
        echo json_encode(['error' => 'Unauthorized IP']);
        exit;
    }

    return true;
}

// Verify callback data by making an API request
function verifyData($endpoint, $code) {
    global $api;

    try {
        // Make an authenticated API request to verify the status
        $response = $api->get("$endpoint/$code");
        return $response;
    } catch (Exception $e) {
        // Handle any API request errors
        error_log("Error verifying data: " . $e->getMessage());
        return null;
    }
}

// Get the request body
$body = file_get_contents('php://input');
$data = json_decode($body, true);

// Process the callback based on the endpoint
$endpoint = $_SERVER['REQUEST_URI'];

// Verify IP for all callbacks
verifyIp();

switch ($endpoint) {
    case '/callbacks/deposits':
        // Handle deposit callbacks
        $code = $data['code'] ?? '';
        $status = $data['status'] ?? '';
        $customId = $data['custom_id'] ?? '';

        error_log("Deposit callback received: $code ($customId) - Status: $status");

        // Verify the data by making an API request
        $verifiedData = verifyData('/deposit', $code);
        if ($verifiedData && $verifiedData['status'] === $status) {
            // Process the callback (update database, send notifications, etc.)
            error_log("Deposit status verified");
        } else {
            error_log("Warning: Callback data verification failed!");
        }

        break;

    case '/callbacks/withdrawals':
        // Handle withdrawal callbacks
        $code = $data['code'] ?? '';
        $status = $data['status'] ?? '';
        $customId = $data['custom_id'] ?? '';

        error_log("Withdrawal callback received: $code ($customId) - Status: $status");

        // Verify the data by making an API request
        $verifiedData = verifyData('/withdraw', $code);
        if ($verifiedData && $verifiedData['status'] === $status) {
            // Process the callback
            error_log("Withdrawal status verified");
        } else {
            error_log("Warning: Callback data verification failed!");
        }

        break;

    case '/callbacks/swaps':
        // Handle swap callbacks
        $code = $data['code'] ?? '';
        $status = $data['status'] ?? '';
        $customId = $data['custom_id'] ?? '';

        error_log("Swap callback received: $code ($customId) - Status: $status");

        // Verify the data by making an API request
        $verifiedData = verifyData('/swap', $code);
        if ($verifiedData && $verifiedData['status'] === $status) {
            // Process the callback
            error_log("Swap status verified");
        } else {
            error_log("Warning: Callback data verification failed!");
        }

        break;
```

```
    case '/callbacks/transfers':
        // Handle transfer callbacks
        $code = $data['code'] ?? '';
        $status = $data['status'] ?? '';
        $customId = $data['custom_id'] ?? '';

        error_log("Transfer callback received: $code ($customId) - Status: $status");

        // Verify the data by making an API request
        $verifiedData = verifyData('/internal-transfer', $code);
        if ($verifiedData && $verifiedData['status'] === $status) {
            // Process the callback
            error_log("Transfer status verified");
        } else {
            error_log("Warning: Callback data verification failed!");
        }

        break;

    case '/callbacks/orders':
        // Handle order callbacks
        $code = $data['code'] ?? '';
        $status = $data['status'] ?? '';
        $customId = $data['custom_id'] ?? '';

        error_log("Order callback received: $code ($customId) - Status: $status");

        // Verify the data by making an API request
        $verifiedData = verifyData('/limit-order', $code);
        if ($verifiedData && $verifiedData['status'] === $status) {
            // Process the callback
            error_log("Order status verified");
        } else {
            error_log("Warning: Callback data verification failed!");
        }

        break;

    case '/callbacks/aml':
        // Handle AML check callbacks
        $code = $data['code'] ?? '';
        $status = $data['status'] ?? '';
        $customId = $data['custom_id'] ?? '';

        error_log("AML check callback received: $code ($customId) - Status: $status");

        // Verify the data by making an API request
        $verifiedData = verifyData('/aml/check', $code);
        if ($verifiedData && $verifiedData['status'] === $status) {
            // Process the callback
            error_log("AML check status verified");
        } else {
            error_log("Warning: Callback data verification failed!");
        }

        break;

    default:
        http_response_code(404);
        echo json_encode(['error' => 'Unknown endpoint']);
        exit;
}

// Return a success response
http_response_code(200);
echo json_encode(['success' => true]);
```

March 21, 2025

March 19, 2025

4.3 JavaScript Integration Examples

This section provides examples of how to integrate with the API using JavaScript.

Authentication

```
// api-client.js
const crypto = require('crypto');
const axios = require('axios');

class ApiClient {
  constructor(baseUrl, accessKey, secretKey) {
    this.baseUrl = baseUrl.replace(/\/$/, '');
    this.accessKey = accessKey;
    this.secretKey = secretKey;
  }

  getTimestamp() {
    return Date.now().toString();
  }

  generateSignature(path, timestamp, body = '') {
    const message = `${this.accessKey}${path}${timestamp}${body}`;
    return crypto
      .createHmac('sha256', this.secretKey)
      .update(message)
      .digest('hex');
  }

  async request(method, endpoint, params = {}, data = null) {
    const path = `/api/v1${endpoint}`;
    const url = `${this.baseUrl}${path}`;
    const timestamp = this.getTimestamp();

    const headers = {
      'X-Access-Key': this.accessKey,
      'X-Timestamp': timestamp,
    };

    let body = '';
    if (['POST', 'PUT', 'PATCH'].includes(method.toUpperCase()) && data !== null) {
      body = JSON.stringify(data);
      headers['Content-Type'] = 'application/json';
    }

    headers['X-Signature'] = this.generateSignature(path, timestamp, body);

    try {
      const response = await axios({
        method,
        url,
        params: method.toUpperCase() === 'GET' ? params : undefined,
        data: body || undefined,
        headers,
      });

      return response.data;
    } catch (error) {
      if (error.response) {
        throw new Error(`API Error: ${JSON.stringify(error.response.data)}`);
      }
      throw error;
    }
  }

  async get(endpoint, params = {}) {
    return this.request('GET', endpoint, params);
  }

  async post(endpoint, data = {}) {
    return this.request('POST', endpoint, {}, data);
  }
}

module.exports = ApiClient;
```

Usage Examples

Check Server Status

```
// status.js
const axios = require('axios');
const ApiClient = require('./api-client');

async function main() {
  try {
    // Status endpoint doesn't require authentication
    const statusResponse = await axios.get('https://api.001k.world/api/v1/status');
    console.log(`Server timestamp: ${statusResponse.data.timestamp}`);

    // For authenticated endpoints
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    // ... other examples
  } catch (error) {
    console.error('Error:', error.message);
  }
}

main();
```

Get Account Balances

```
// balances.js
const ApiClient = require('./api-client');

async function getBalances() {
  try {
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    const balances = await api.get('/balance');
    balances.forEach(balance => {
      console.log(`${balance.currency}: ${balance.available} (available), ${balance.frozen} (frozen)`);
    });
  } catch (error) {
    console.error('Error:', error.message);
  }
}

getBalances();
```

Generate Deposit Address

```
// generate-deposit-address.js
const ApiClient = require('./api-client');

async function generateDepositAddress() {
  try {
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    const response = await api.post('/deposit/generate_address', {
      currency: 'BTC',
      method: 'btc_native',
      callback_url: 'https://your-server.com/callbacks/deposits',
      custom_id: 'my-deposit-address-123'
    });

    console.log(`Deposit address: ${response.address}`);
    if (response.tag) {
      console.log(`Tag: ${response.tag}`);
    }
  } catch (error) {
    console.error('Error:', error.message);
  }
}

generateDepositAddress();
```

Create Withdrawal

```
// create-withdrawal.js
const ApiClient = require('./api-client');

async function createWithdrawal() {
  try {
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    const response = await api.post('/withdraw', {
      currency: 'BTC',
      method: 'btc_native',
      amount: '0.01000000',
      address: 'bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh',
      callback_url: 'https://your-server.com/callbacks/withdrawals',
      custom_id: 'my-withdrawal-123'
    });

    console.log(`Withdrawal created: ${response.code}`);
    console.log(`Status: ${response.status}`);
  } catch (error) {
    console.error('Error:', error.message);
  }
}

createWithdrawal();
```

Create Swap

```
// create-swap.js
const ApiClient = require('./api-client');

async function createSwap() {
  try {
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    // First, get available swap directions
    const directions = await api.get('/swap-directions');
    directions.forEach(direction => {
      console.log(`${direction.code}: ${direction.spend_currency} -> ${direction.receive_currency} (rate: ${direction.rate})`);
    });

    // Create a swap
    const response = await api.post('/swap/open', {
      direction: 'BTC-USDT',
      spend_amount: '0.01000000',
      callback_url: 'https://your-server.com/callbacks/swaps',
      custom_id: 'my-swap-123'
    });

    console.log(`Swap created: ${response.code}`);
    console.log(`Spend: ${response.spend_amount} ${response.spend_currency}`);
    console.log(`Receive: ${response.receive_amount} ${response.receive_currency}`);
    console.log(`Status: ${response.status}`);
  } catch (error) {
    console.error('Error:', error.message);
```

```
  }
}

createSwap();
```

Create Internal Transfer

```
// create-transfer.js
const ApiClient = require('./api-client');

async function createTransfer() {
  try {
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    const response = await api.post('/internal-transfer', {
      currency_code: 'BTC',
      amount: '0.01000000',
      recipient: 'user123',
      callback_url: 'https://your-server.com/callbacks/transfers',
      custom_id: 'my-transfer-123',
      note: 'Payment for services'
    });

    console.log(`Transfer created: ${response.code}`);
    console.log(`Status: ${response.status}`);
  } catch (error) {
    console.error('Error:', error.message);
  }
}

createTransfer();
```

Create Limit Order

```
// create-limit-order.js
const ApiClient = require('./api-client');

async function createLimitOrder() {
  try {
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    // First, get available trading pairs
    const pairs = await api.get('/limit-pairs');
    pairs.forEach(pair => {
      console.log(`${pair.symbol}: ${pair.current_price} (fee: ${pair.fee_percent}%)`);
    });

    // Create a limit order
    const response = await api.post('/limit-order', {
      pair: 'BTCUSDT',
      side: 'buy',
      receive_amount: '0.01000000',
      limit_price: '29000.00',
      callback_url: 'https://your-server.com/callbacks/orders',
      custom_id: 'my-order-123'
    });

    console.log(`Order created: ${response.code}`);
    console.log(`Status: ${response.status}`);
    console.log(`Base amount: ${response.base_amount}`);
    console.log(`Quote amount: ${response.quote_amount}`);
    console.log(`Filled base amount: ${response.filled_base_amount}`);
    console.log(`Filled quote amount: ${response.filled_quote_amount}`);
    console.log(`Fee currency: ${response.fee_currency}`);
    console.log(`Filled fee amount: ${response.filled_fee_amount}`);
    console.log(`Created at: ${response.created_at}`);
    console.log(`Updated at: ${response.updated_at}`);
    console.log(`Done at: ${response.done_at || 'N/A'}`);
  } catch (error) {
    console.error('Error:', error.message);
  }
}

createLimitOrder();
```

Create AML Check

```
// create-aml-check.js
const ApiClient = require('./api-client');

async function createAmlCheck() {
  try {
    const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

    // First, get available plans and tokens
    const plans = await api.get('/aml/plan');
    const tokens = await api.get('/aml/token');

    plans.forEach(plan => {
      console.log(`Plan: ${plan.code} - ${plan.name} (price: ${plan.price} ${plan.currency})`);
    });

    tokens.forEach(token => {
      console.log(`Token: ${token.symbol} on ${token.network} (ID: ${token.id})`);
    });

    // Create an AML check
    const response = await api.post('/aml/check', {
      direction: 'withdrawal',
      token: 1,  // Bitcoin
      address: 'bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh',
      callback_url: 'https://your-server.com/callbacks/aml',
      custom_id: 'my-aml-check-123'
    });
```

```
    console.log(`AML check created: ${response.code}`);
    console.log(`Status: ${response.status}`);
    console.log(`Direction: ${response.direction}`);
    console.log(`Token: ${response.token}`);
    console.log(`Address: ${response.address}`);
    console.log(`Transaction: ${response.transaction || 'N/A'}`);
    console.log(`Created at: ${response.created_at}`);
    console.log(`Updated at: ${response.updated_at}`);
    if (response.report) {
      console.log(`Report status: ${response.report.status}`);
      console.log(`Risk score: ${response.report.riskscore}`);
      console.log(`Alert grade: ${response.report.alert_grade}`);
    }
  } catch (error) {
    console.error('Error:', error.message);
  }
}

createAmlCheck();
```

Callback Handling

When the status of an operation changes, the API will send a callback to the URL provided in the `callback_url` parameter. Here's an example of how to handle these callbacks in an Express application:

Important Note: You may not receive callbacks for every intermediate status change or field update if changes occur very rapidly. When operations transition through multiple states quickly, intermediate callbacks may be skipped. However, you will always receive callbacks for the final states of operations. Design your callback handling to be idempotent and not dependent on receiving every single state transition.

```
// callback-server.js
const express = require('express');
const ApiClient = require('./api-client');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

// Your API client
const api = new ApiClient('https://api.001k.world', 'your_access_key', 'your_secret_key');

// IP whitelist for callback security
const ALLOWED_IPS = ['192.168.1.1', '10.0.0.1']; // Replace with actual IPs provided by support

// Verify IP middleware
function verifyIp(req, res, next) {
  const clientIp = req.ip || req.connection.remoteAddress;
  if (!ALLOWED_IPS.includes(clientIp)) {
    return res.status(403).json({ error: 'Unauthorized IP' });
  }
  next();
}

// Verify data by making an API request
async function verifyData(endpoint, code) {
  try {
    // Make an authenticated API request to verify the status
    const response = await api.get(`${endpoint}/${code}`);
    return response;
  } catch (error) {
    console.error(`Error verifying data: ${error.message}`);
    return null;
  }
}

// Parse JSON bodies
app.use(bodyParser.json());

// Deposit callbacks
app.post('/callbacks/deposits', verifyIp, async (req, res) => {
  const { code, status, custom_id } = req.body;

  console.log(`Deposit callback received: ${code} (${custom_id}) - Status: ${status}`);

  // Verify the data by making an API request
  const verifiedData = await verifyData('/deposit', code);
  if (verifiedData && verifiedData.status === status) {
    // Process the callback (update database, send notifications, etc.)
    console.log("Deposit status verified");
  } else {
    console.log("Warning: Callback data verification failed!");
  }

  res.json({ success: true });
});

// Withdrawal callbacks
app.post('/callbacks/withdrawals', verifyIp, async (req, res) => {
  const { code, status, custom_id } = req.body;

  console.log(`Withdrawal callback received: ${code} (${custom_id}) - Status: ${status}`);

  // Verify the data by making an API request
  const verifiedData = await verifyData('/withdraw', code);
  if (verifiedData && verifiedData.status === status) {
    // Process the callback
    console.log("Withdrawal status verified");
  } else {
    console.log("Warning: Callback data verification failed!");
  }

  res.json({ success: true });
});

// Swap callbacks
app.post('/callbacks/swaps', verifyIp, async (req, res) => {
  const { code, status, custom_id } = req.body;
```

```
  console.log(`Swap callback received: ${code} (${custom_id}) - Status: ${status}`);

  // Verify the data by making an API request
  const verifiedData = await verifyData('/swap', code);
  if (verifiedData && verifiedData.status === status) {
    // Process the callback
    console.log("Swap status verified");
  } else {
    console.log("Warning: Callback data verification failed!");
  }

  res.json({ success: true });
});

// Transfer callbacks
app.post('/callbacks/transfers', verifyIp, async (req, res) => {
  const { code, status, custom_id } = req.body;

  console.log(`Transfer callback received: ${code} (${custom_id}) - Status: ${status}`);

  // Verify the data by making an API request
  const verifiedData = await verifyData('/internal-transfer', code);
  if (verifiedData && verifiedData.status === status) {
    // Process the callback
    console.log("Transfer status verified");
  } else {
    console.log("Warning: Callback data verification failed!");
  }

  res.json({ success: true });
});

// Order callbacks
app.post('/callbacks/orders', verifyIp, async (req, res) => {
  const { code, status, custom_id } = req.body;

  console.log(`Order callback received: ${code} (${custom_id}) - Status: ${status}`);

  // Verify the data by making an API request
  const verifiedData = await verifyData('/limit-order', code);
  if (verifiedData && verifiedData.status === status) {
    // Process the callback
    console.log("Order status verified");
  } else {
    console.log("Warning: Callback data verification failed!");
  }

  res.json({ success: true });
});

// AML check callbacks
app.post('/callbacks/aml', verifyIp, async (req, res) => {
  const { code, status, custom_id } = req.body;

  console.log(`AML check callback received: ${code} (${custom_id}) - Status: ${status}`);

  // Verify the data by making an API request
  const verifiedData = await verifyData('/aml/check', code);
  if (verifiedData && verifiedData.status === status) {
    // Process the callback
    console.log("AML check status verified");
  } else {
    console.log("Warning: Callback data verification failed!");
  }

  res.json({ success: true });
});

app.listen(PORT, () => {
  console.log(`Callback server running on port ${PORT}`);
});
```

March 21, 2025

March 19, 2025