Remittance Public API Documentation Documentation

Dhruv Patel

Nov 02, 2019

Table of Contents:

The Client API is organized around REST. Our API has predictable, resource-oriented URLs, and uses HTTP response codes to indicate API errors. We use built-in HTTP features, like HTTP verbs, which are understood by off-the-shelf HTTP clients. We support cross-origin resource sharing, allowing you to interact securely with our API from a client-side web application (though you should never expose your secret credentials in any public website's client-side code). JSON is returned by all API responses, including errors, although our API libraries convert responses to appropriate language-specific objects.

Example HTTP Request

```
GET /client/v1 HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
```

Response

```
{
    "code": 200,
    "message": "Server is Reachable."
}
```

Response

```
{
    "code": 401,
    "type": "authentication_error",
    "message": "Incorrect X-Client-ID or X-Client-Secret"
}
```

CHAPTER 1

Authentication

All api endpoints are subject to client application authentication. Client application authentication to the API is performed via headers

```
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
```

All API requests must be made over HTTPS. Calls made over plain HTTP will fail. API requests without client authentication will also fail.

CHAPTER 2

Errors

API uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the 2xx range indicate success. Codes in the 4xx range indicate an error that failed given the information provided (e.g., a required parameter was omitted, a transaction failed, etc.). Codes in the 5xx range indicate an error with backend servers (these are rare).

Some 4xx errors that could be handled programmatically (e.g., an invalid account number) include an error code that briefly explains the error reported.

2.1 HTTP status code summary

HTTP Status Code	Description
200	Everything worked as expected.
201	A new resource was created.
400	The request was unacceptable, often due to missing a required parameter.
401	No valid authentication headers were present.
402	The parameters were valid but the request failed.
403	You are not allowed to access this resource.
404	The requested resource doesn't exist.
405	The request method is not allowed for invoked endpoint.
500	Something went wrong on server end. (These are rare.)

2.2 Error Types

Type	Description
api_error	API errors cover any other type of problem (e.g., a temporary problem with backend or 3rd Party's servers), and are extremely uncommon.
authentication_error	Failure to properly authenticate yourself in the request.
two_factor_authentication_error	User is not two factor authenticated.
authorization_error	Trying to access restricted resource.
invalid_request_error	This arise when your request has invalid parameters.

2.3 Handling Errors

Our API raise exceptions for many reasons, such as a failed transfer, invalid parameters, authentication errors, and network unavailability. We recommend writing code that gracefully handles all possible API exceptions.

Attribute	Description
code	For some errors that could be handled programmatically. A short string indicating the error code reported. See Error Codes section.
message	A human-readable message providing more details about the error.
param	If the error is parameter-specific, the parameter related to the error. For example, you can use this to display a message near the correct form field.

2.4 Error Codes

Some 4xx errors that could be handled programmatically (e.g., request contains invalid parameter) include an error code—a short string with a brief explanation—as a value for code. Below is a list of possible error codes that can be returned.

Code	Description
_required	One or more required parameters are missing.
_empty	One or more required fields contain empty value.
_invalid	One or more required values are invalid.
unique	The parameter has failed unique constraint check.
_missingHeader	One or more required headers are missing or empty.
not_found	Resource could not be found.
_sameAsCurrentPassword	New password is same as current password.
_invalidConfirmPassword	Password and confirm password do not match.
not_found	Resource could not be found.

CHAPTER 3

Settings

Endpoint: `/client/v1/settings`

Method: `GET`

Example HTTP Request

```
GET /client/v1/settings HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
```

Response

```
{
    "code": 200,
    "message": "Success",
    "settings": {
        "trading_name": "ABC Money Transfer",
        "company_name": "ABC Limited",
        "tag_line": "Sending Money with Care",
        "license": "ABC Money Transfer is the trading name of ABC Limited. Registered␣
↪company No. 07952651. We are registered as a Money Service Business and supervised␣
↪by HM Revenue & Customs (HMRC) under Money Laundering Regulations (MLR) No.XXXXXXX.
↪We are Authorised and Regulated by the Financial Conduct Authority (FCA) as a␣
↪payment institution with reference number XXXXXX",
        "cs_number": "+44 10 1234 5678",
        "cs_email": "support@abcmoneytransfer.com",
        "hq_address": "Headquarter Street",
        "hq_city": "City",
        "hq_region": "",
        "hq_postcode": "Post Code",
        "hq_country": "United Kingdom",
        "locales": [
            {
                "id": "en_GB",
                "label": "English (UK)"
```

(continues on next page)

(continued from previous page)


```
            }
        ],
        "timezones": [
            {
                "id": "Africa/Abidjan",
                "offset": "+00:00",
                "nice": "Abidjan"
            },
            {
                "id": "Africa/Accra",
                "offset": "+00:00",
                "nice": "Accra"
            },
            {
                "id": "Africa/Addis_Ababa",
                "offset": "+03:00",
                "nice": "Addis Ababa"
            }
        ]
    }
}
```

CHAPTER 4

Password Policy

Endpoint: `/client/v1/password-policy`

Method: `GET`

Example HTTP Request

```
GET /client/v1/password-policy HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
```

Response

```
{
    "code": 200,
    "message": "Success",
    "policy": {
        "min_length": 6,
        "uppercase": true,
        "lowercase": true,
        "digit": true,
        "wildcard": true
    }
}
```

CHAPTER 5

Countries

5.1 Get All Countries

Get list of all countries in the system.

Endpoint: `/client/v1/countries`

Method: `GET`

Example HTTP Request

```
GET /client/v1/countries HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
```

Response

```
{
    "code": 200,
    "message": "Success",
    "countries": [
        {
            "known_name": "United Kingdom",
            "official_name": "United Kingdom of Great Britain and Northern Ireland",
            "endonym": "United Kingdom",
            "demonym": "British",
            "isd_code": 44,
            "iso_code_alpha2": "GB",
            "iso_code_alpha3": "GBR",
            "fips_code": "UK",
            "slug": "united-kingdom"
        },
        {
            "known_name": "United States",
            "official_name": "United States of America",
```

(continues on next page)

(continued from previous page)


```
            "endonym": "United States",
            "demonym": "American",
            "isd_code": 1,
            "iso_code_alpha2": "US",
            "iso_code_alpha3": "USA",
            "fips_code": "US",
            "slug": "united-states"
        }
    ]
}
```


5.2 Get Source Countries

Note: Customers can register or send money from these countries only.

Endpoint: `/client/v1/sources`

Method: `GET`

Example HTTP Request

```
GET /client/v1/countries/sources HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
```

Response


```
{
    "code": 200,
    "message": "Success",
    "countries": [
        {
            "known_name": "Austria",
            "official_name": "Austria",
            "endonym": "Österreich",
            "demonym": "Austrian",
            "isd_code": 43,
            "iso_code_alpha2": "AT",
            "iso_code_alpha3": "AUT",
            "fips_code": "AU",
            "slug": "austria"
        },
        {
            "known_name": "Belgium",
            "official_name": "Belgium",
            "endonym": "België",
            "demonym": "Belgian",
            "isd_code": 32,
            "iso_code_alpha2": "BE",
            "iso_code_alpha3": "BEL",
            "fips_code": "BE",
            "slug": "belgium"
        },
```


(continues on next page)

(continued from previous page)


```
        {
            "known_name": "United Kingdom",
            "official_name": "United Kingdom of Great Britain and Northern Ireland",
            "endonym": "United Kingdom",
            "demonym": "British",
            "isd_code": 44,
            "iso_code_alpha2": "GB",
            "iso_code_alpha3": "GBR",
            "fips_code": "UK",
            "slug": "united-kingdom"
        }
    ]
}
```

CHAPTER 6

Customers

6.1 Register

The API returns 201 status code on successful registration.

Endpoint: `/client/v1/register`

Method: `POST`

Additional Headers

Parameter	Description
X-Client-IP	IP Address of the customer.
X-Client-UA	User Agent of client application. For mobile apps this should be Application Version
X-Client-Fingerprint	The unique identifier for device (Device ID).

Parameters

Parameter	Description
email	Email address of the customer.
password	Password
confirm_password	Password Confirmation
email_verification_url	Enter the endpoint for creating a email verification url, you should provide *:token* variable in this url. This is going to be replaced by email verification token generated by backend. The customer will be redirect to this page for verification. eg. *https://www.example.com/users/verify-email/:token* will be *https://www.example.com/users/verify-email/bacdefghi*

Example HTTP Request

```
POST /client/v1/register HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
X-Client-IP: user-ip-address
X-Client-UA: user-agent
X-Client-Fingerprint: unique-device-fingerprint
Content-Type: application/x-www-form-urlencoded

email=test%40test.com
&password=Password%40741
&confirm_password=Password%40741
&email_verification_url=https%3A%2F%2Fwww.example.com%2Fusers%2Fverify-email%2F
↪%3Atoken
```

Response

```
{
    "code": 201,
    "message": "Success",
    "token": "63pS1fe84r940yHocou6I7......"
}
```

Response Description

Parameter	Description
token	The login token for customer.

Example Failed Response

```
{
    "code": 400,
    "type": "invalid_request_error",
    "message": "Invalid request",
    "errors": [
```

(continues on next page)

(continued from previous page)

```
        {
            "param": "email",
            "code": "unique",
            "message": "Email address is already registered."
        }
    ]
}
```

6.2 Login

The API returns 200 status code on successful login and 401 for login failure due to incorrect credentials.

Endpoint: `/client/v1/login`

Method: `POST`

Additional Headers

Parameter	Description
X-Client-IP	IP Address of the customer.
X-Client-UA	User Agent of client application. For mobile apps this should be Application Version
X-Client-Fingerprint	The unique identifier for device (Device ID).

Parameters

Parameter	Description
email	Email address of the customer.
password	Password

Example HTTP Request

```
POST /client/v1/login HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
X-Client-IP: user-ip-address
X-Client-UA: user-agent
X-Client-Fingerprint: unique-device-fingerprint
Content-Type: application/x-www-form-urlencoded

email=test%40test.com
&password=Password%40741
```

Response

```
{
    "code": 200,
    "message": "Success",
```

(continues on next page)

(continued from previous page)

```
    "token": "63pS1fe84r940yHocou6I7......"
}
```

Response Description

Parameter	Description
token	The login token for customer.

Example Failed Response

```
{
    "code": 401,
    "type": "authentication_error",
    "message": "Incorrect email address or password"
}
```

6.3 Forgot Password

Endpoint: `/client/v1/forgot-password`

Method: `POST`

Parameters

Parameter	Description
email	Email address of the customer.
reset_password_url	Enter the endpoint for creating a reset password url, you should provide *:token* variable in this url. This is going to be replaced by reset password token generated by backend. The customer will be redirect to this page for resetting password. eg. *https://www.example.com/reset-password/:token* will be *https://www.example.com/reset-password/bacdefghi*

Example HTTP Request

```
POST /client/v1/forgot-password HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
Content-Type: application/x-www-form-urlencoded

email=test%40test.com
&reset_password_url=https%3A%2F%2Fwww.example.com%2Freset-password%2F%3Atoken
```

Response

```
{
    "code": 200,
    "message": "Success",
    "recipient": "te***@test.com"
}
```

Response Description

Parameter	Description
recipient	The masked email address of customer account.

Example Failed Response

```
{
    "code": 400,
    "type": "invalid_request_error",
    "message": "Invalid request",
    "errors": [
        {
            "param": "email",
            "code": "not_found",
            "message": "Customer account not found."
        }
    ]
}
```

6.4 Reset Password

Endpoint: `/client/v1/reset-password`

Method: `POST`

Parameters

Parameter	Description
token	Request token
password	Password
confirm_password	Password Confirmation

Example HTTP Request

```
POST /client/v1/reset-password HTTP/1.1
X-Client-ID: your-client-id
X-Client-Secret: your-client-secret
Content-Type: application/x-www-form-urlencoded

token=bacdefghi
&password=Password%40741
&confirm_password=Password%40741
```

Response

```
{
    "code": 200,
    "message": "Success"
}
```

Example Failed Responses

```
{
    "code": 400,
    "type": "invalid_request_error",
    "message": "Invalid request",
    "errors": [
        {
            "param": "confirm_password",
            "code": "_invalidConfirmPassword",
            "message": "New password and confirm password are not same."
        }
    ]
}
```

```
{
    "code": 400,
    "type": "invalid_request_error",
    "message": "Invalid request",
    "errors": [
        {
            "param": "confirm_password",
            "code": "_invalidConfirmPassword",
            "message": "New password and confirm password are not same."
        }
    ]
}
```