NEOCURRENCY®

API SPECIFICATIONS

VERSION 1.8.5

UPDATED 05.15.2023

INDEX

CONTENTS

SUMMARY

STANDARD USER FLOW

The Production Environment base URL is: https://redeem.yourdigitalreward.com

The standard user flow for using the Production Environment is to first Authenticate using the Authentication endpoint. After which to create a new campaign using the Create Campaign endpoint. After a new Campaign has been created, the user should activate the campaign using the Activate Campaign endpoint before being able to crate codes using the Create Codes endpoint. After a campaign has been created and activated additional codes can be added at a later time using the Create Codes endpoint.

Before starting with the Production Environment, we highly recommend using our Sandbox Environment which mimics the functionality of the Production Environment and allows for easy and safe testing of endpoints and user flow. Please ask about access to our Sandbox Environment.

DIFFERENCES BETWEEN SANDBOX AND PRODUCTION

There are some differences between the Production Environment and the Sandbox Environment. The two major differences between the Sandbox Environment and the Production Environment are as follows.

Firstly, when calling Sandbox Environment methods there is a /sandbox/ prefix attached to the method declarations. For example, the Sandbox Environment brands method is called via `api/sandbox/brands` while the Production Environment brands method is called via `api/brands.`

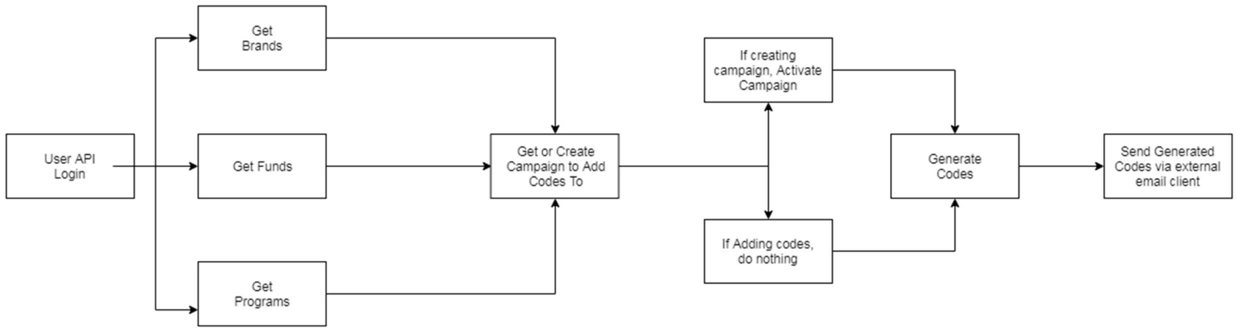
Secondly, the code redemption in the Sandbox Environment is simulated. So when attempting to redeem codes via https://redeem.yourdigitalreward.com/sandbox/activate-code/{code} user will receive a message that the code has been redeemed but that is the extent of the code redemption in the Sandbox Environment.

DISCUSSION: EXPIRATION DATES

The NeoCurrency System Campaigns have an end date associated with each campaign created in the NeoCurrency System. The end date of a campaign is the date after which all codes generated for this campaign become expired or invalid. This is done so that the system is more flexible in handling both limited duration and unlimited duration campaigns. In the case of a client wanting to run a campaign only for a limited amount of time, we suggest entering a campaign end date when the client wants the system generated codes to expire. If on the other hand the client would like to have an unlimited length campaign, we recommend setting a campaign end date five or ten years in the future. As a recap, NeoCurrency system generated codes are only valid while the campaign which has generated those codes has not passed its end date.

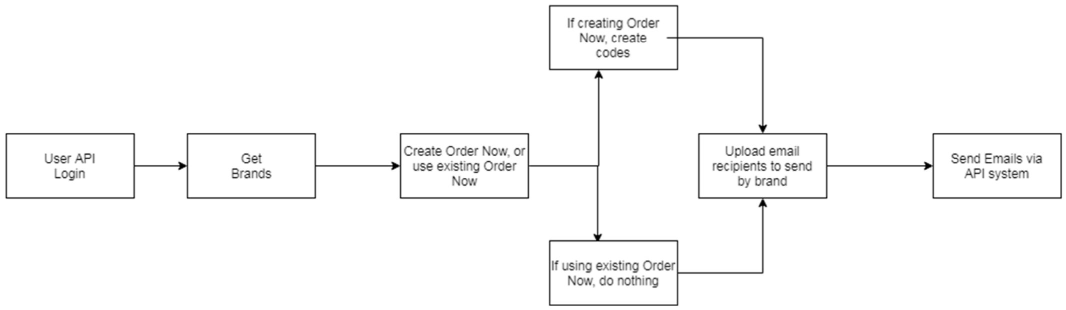
DISCUSSION: CREATEORDERNOW vs CREATECODES

In the NeoCurrency System, there is a distinction between Order Now orders and creating additional codes for an already existing campaign. The main distinction is the Order Now orders allow the user to upload and send emails for the codes created with Order Now orders while this is not allowed for create/codes standard campaigns. If user wishes to use the email functionality, user should use the createordernow order endpoint and flow. Below is a flow diagram comparing the ordering process between using Order Now and using the standard code creation.



Create codes API flow

User API Login
Get Brands
Get Funds
Get Programs
Get or Create Campaign to Add Codes To
If creating campaign, Activate Campaign
If Adding codes, do nothing
Generate Codes
Send Generated Codes via external email client



Order Now API flow

User API Login
Get Brands
Create Order Now, or use existing Order Now
If creating Order Now, create codes
If using existing Order Now, do nothing
Upload email recipients to send by brand
Send Emails via API system

EXAMPLE: CREATING YOUR FIRST ORDER

In this example, we will go over the needed endpoints to create your first order.

The overall flow is as follows:

POST /api/sandbox/createordernow

Below is an example of the requests and responses from this transaction flow:

POST /api/sandbox/createordernow
Request:

```
{
"custom1":"this is my first order", ← This is a custom field that can be used for things like
custom data which will not be shown to the end customer.
            "brands" : [ {
                      "id" : 623,  ← This is the brand you are looking to order
                      "denomination" : 0.05,
                      "quantity" : 1
            } ]
}
```

Response:

```
{
    "success": {
           "order_id": 6208,
           "custom1":”this is my first order”,
           "data": [
           {
                      "campaign_brand_id": 10031,
                      "denomination": "0.05",
                       "currency": "USD",
                      "codes": [
                                 "c613e814692e94906be3d8f592551894e5bd1f85"
                       ],
                      "uuids": {

            "c613e814692e94906be3d8f592551894e5bd1f85": "3520f5a13757d26d717ac35c5c1f0
d7f202103111028291"
                       }
           }
       ]
    }
}
```

EXAMPLE: CREATING YOUR FIRST ORDER AND SENDING YOUR FIRST EMAIL

In this example, we will go over the needed endpoints to create your first order and send your first email.

The overall flow is as follows:

POST /api/sandbox/createordernow
POST /api/sandbox/addemails
POST /api/sandbox/sendemails

Below is an example of the requests and responses from this transaction flow:

POST /api/sandbox/createordernow

Request:

```
{
"custom1":"this is my first order", ← This is a custom field that can be used for things like
custom data which will not be shown to the end customer.
            "brands" : [ {
                      "id" : 623,  ← This is the brand you are looking to order
                      "denomination" : 0.05,
                      "quantity" : 1
            } ]
}
```

Response:

```
{
   "success": {
          "order_id": 6208,  ← This will be needed when adding and sending emails
          "custom1":”this is my first order”,
          "data": [
          {
                      "campaign_brand_id": 10031,  ← This will be needed when adding and sending
emails
                      "denomination": "0.05",
                       "currency": "USD",
                      "codes": [
                                 "c613e814692e94906be3d8f592551894e5bd1f85"
                       ],
                      "uuids": {
```

```
            "c613e814692e94906be3d8f592551894e5bd1f85": "3520f5a13757d26d717ac35c5c1f0
d7f202103111028291"
                        }
            }
        ]
    }
}
```

POST /api/sandbox/addemails

Request:

```
{
"order_id": 6208,
"campaign_brand_id":10031,
"emails": [
{
"email": "johndoe@test.com",
"first_name": "John",
"last_name": "Doe",
"subject": " This is a test"
}
]
}
```

Response:

```
{
    "success": "You have successfully added one email!"
}
```

POST   api/sandbox/sendemails

Request:

```
{
            "order_id": 6208,
            "campaign_brand_id": 10031
}
```

Response:

```
{
    "send": 1
}
```

HEADERS

In order to work properly, the NeoCurrency REST API needs the header “**`Content-Type`**” with the value “application/json” to be included in each request.

AUTHENTICATION

AUTHENTICATE

The NeoCurrency REST API uses oAuth2 authentication with Password Grant. In order to successfully authenticate and send requests to the REST API, the user sets the request Authorization header to Bearer and transmits the access_token received as part of a successful authentication response.

Request

Method	URL
POST	api/get-token

Headers	Value
Content-Type	**application/json**

Type	Params	Values
POST	client_id	string
POST	client_secret	string
POST	email	string
POST	password	string

`client_id`

Client ID provided by the NeoCurrency REST API

`client_secret`

Companion to the client_id that is provided by the NeoCurrency REST API

`email`

The email of the user that wishes to authenticate with the API.

`password`

The password of the user that wishes to authenticate with the API

Request example:

```
{
    "client_id":"JOI8EX4J3K8706L3HQ0KIL3H",
    "client_secret":"Q0JBQEHWT1J899BG3UM83GNMWG53OW4OQB5GMCNYH7DYDN",
    "email" : "api@test.com",
    "password": "123456"
}
```

Response

Status	Response
`200`	The rest API will respond with a JSON object containing the following properties: - access_token: The access token. This expires in 1 day
`308`	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
`401`	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
`500`	The rest API will respond with a JSON object containing the following properties: - error: “500 Internal Server Error.”

Response example:

```
{
    "access_token": "alphanumeric code here"
}
```

METHODS

1. GET USER

Get information for the authenticated user

Request

Method	URL
GET	api/user

Type	Params	Values
Headers	Authorization: Bearer	access_token

access_token
access_token must be sent with all client requests. The access_token helps the server to validate the request source.

Response

Status	Response
200	The rest API will respond with a JSON object containing the following properties: - id: Integer - name: String - email: String - role_id: Integer - created_at: Datetime - updated_at: Datetime

	- active: Integer 1-Active, 0-Not Active
308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
{
    "id": 3,
    "name": "Api test",
    "email": "api@neocurrency.com",
    "role_id": "5",
    "created_at": "2017-08-13 19:46:45",
    "updated_at": "2017-12-07 13:46:24",
    "active": "1"
}
```

2. GET BRANDS

Getting all the brands that the authenticated user has access to.

Request

Method	URL
`GET`	`api/brands`

Type	Params	Values
`Header`	`Authorization: Bearer`	`access_token`

`access_token`

`access_token` must be sent with all client requests. The access_token helps the server to validate the request source.

Response

Status	Response
`200`	The rest API will respond with an array of JSON objects, each of them containing the following properties: - id: Integer - name: String - active: Integer 1-Active, 0-Not Active - created_at: Datetime - updated_at: Datetime - user_id: Integer - ID of authenticated user
`308`	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
`401`	The rest API will respond with a JSON object containing the following properties: error: “Unauthenticated.”
`500`	The rest API will respond with a JSON object containing the following properties:

	- error: ”500 Internal Server Error.”

Response example:

```
[
   {
      "id": 69,
      "name": "Burger King $5-$1000",
      "real_name": "Burger King",
      "value": 0,
      "min": 5,
      "max": 1000,
      "description": "Every day, more than 11 million guests visit BURGER KING® restaurants around the world. And they do so because our restaurants are known for serving high-quality, great-tasting, and affordable food. Founded in 1954, BURGER KING® is the second largest fast food hamburger chain in the world. The original HOME OF THE WHOPPER®, our commitment to premium ingredients, signature recipes, and family-friendly dining experiences is what has defined our brand for more than 50 successful years.",
      "terms": "Not redeemable for cash.",
      "logo":
"https://redeem.yourdigitalreward.com/brands/7YAcQCnG0LplDB.2CfY_ZFN5KATIdfnA6YD9.png",
      "card_img":
"https://redeem.yourdigitalreward.com/brands/7YAcQCnG0LplDB.2CfY_ZFN5KATIdfnA6YD9.png",
      "country_code": "US",
      "currency_name": "USD"
   },

   {
     "id": 119,
      "name": "Gilt $25",
      "real_name": "Gilt",
      "value": 25,
      "min": 0,
      "max": 0,
      "description": "Gilt.com E-Gift Cards provide instant insider access to top designer brands and experiences. New flash sales start daily for women, men, home and children. Plus, get access to luxury lifestyle experiences locally with Gilt City.",
      "terms": "Read our E-Gift Card terms of use at http://www.gilt.com/company/giftCardTermsOfUse",
      "logo": "https://api.giftango.com/imageservice/Images/300x190/162528_vgc.png",
      "card_img": "https://api.giftango.com/imageservice/Images/300x190/162528_vgc.png",
      "country_code": "US",
      "currency_name": "USD"
}
]
```

3. GET FUNDS

Getting all the funds that the authenticated user has access to.

Request

Method	URL
`GET`	`api/funds`

Type	Params	Values
`Header`	`Authorization: Bearer`	`access_token`

`access_token`

`access_token` must be sent with all client requests. The access_token helps the server to validate the request source.

Response

Status	Response
`200`	The rest API will respond with an array of JSON objects
`308`	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
`401`	The rest API will respond with a JSON object containing the following properties: error: “Unauthenticated.”
`500`	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
[
   {
      "id": 1,
      "client_id": 3,
      "currency_id": "USD",
      "value": 1000
   },
   {
      "id": 9,
      "client_id": 3,
      "currency_id": "EUR",
      "value": 500
   }
]
```

4. GET CURRENT BALANCE

Getting the total of the current funds and any balance distributed over any Projects the user has access to. This endpoint shows the current total balance that the client has access to, to create and redeem codes with. This endpoint will provided more up to date information than the /funds endpoint which only shows the current fund balance for the user.

Request

Method	URL
`GET`	`api/currentbalances`

Type	Params	Values
`Header`	`Authorization: Bearer`	`access_token`

`access_token`

`access_token` must be sent with all client requests. The access_token helps the server to validate the request source.

Response

Status	Response
`200`	The rest API will respond with an array of JSON objects
`308`	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
`401`	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
`500`	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
[
   {
      "id": 1,
      "client_id": 3,
      "currency_id": "USD",
      "value": 1469
   },
   {
      "id": 5,
      "client_id": 3,
      "currency_id": "EUR",
      "value": 100
   }
]
```

5. GET PROJECTS

Getting all the projects that the authenticated user has access to.

Request

Method	URL
`GET`	`api/projects`

Type	Params	Values
`Header`	`Authorization: Bearer`	`access_token`

`access_token`

`access_token` must be sent with all client requests. The access_token helps the server to validate the request source.

Response

Status	Response
`200`	The rest API will respond with an array of JSON objects
`308`	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
`401`	The rest API will respond with a JSON object containing the following properties: - error: Unauthenticated."
`500`	The rest API will respond with a JSON object containing the following properties: - error: "500 Internal Server Error."

Response example:

```
[
    {
        "id": 1,
        "p_group_id": 3,
        "p_name": "Summer Campaigns",
        "p_description": "Summer Campaigns Project",
        "p_active": 1,
        "created_at": "2017-09-20 14:52:15",
        "updated_at": "2017-09-30 12:14:15"
    },
    {
        "id": 2,
        "p_group_id": 3,
        "p_name": "Winter Campaigns",
        "p_description": "Winter Campaign Project",
        "p_active": 1,
        "created_at": "2017-09-20 20:12:23",
        "updated_at": "2017-12-07 10:02:37"
    }
]
```

6. CREATE CAMPAIGN

Method called when creating a campaign. Once this method is successfully executed a campaign with pending status will be created for the user.

Request

Method	URL
POST	api/campaigns/create

Type	Params	Values
Header	Authorization: Bearer	access_token

Type	Params	Values
POST	name	string
POST	start_date	Date: datetime format(MM/DD/YYYY HH:ii)
POST	end_date	Date: datetime format (MM/DD/YYYY HH:ii)
POST	timezone	string
POST	project_id	integer
POST	fund_id	integer

access_token
access_token must be sent with all client requests. The access_token helps the server to validate the request source.

name
The name of the campaign, there are no restrictions on name and multiple campaigns can have the same name

start_date
The date, down to the hour and minute combination of when the campaign will start

end_date
The date, down to the hour and minute combination of when the campaign will end.

timezone

The timezone for the start and end date of the campaign. We use PHP timezones (http://php.net/manual/en/timezones.php) as these automatically take into account daylight savings time. The possible values for timezone are: “Eastern Standard Time”, “Central Standard Time”, “Mountain Standard Time”, “Pacific Standard Time”, “Hawaii”.

project_id

The ID of the Project attached to this Campaign. See get all projects by user.

fund_id

The ID of the Fund attached to this Campaign. See get all funds by user.

Request example:

```
{
    "name":"api campaign 1",
    "start_date": "12/27/2017 13:00",
    "end_date": "12/31/2017 13:00",
    "timezone": "Eastern Standard Time",
    "project_id": 1,
    "fund_id": 1
}
```

Response

Status	Response
200	The rest API will respond with a JSON objects containing the following properties: - success
308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
422	The rest API will respond with an array of JSON objects, each of them, containing the following properties: - name_of_the_problem_field: array of validation error messages
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”

500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
{
   "success": [
      {
         "create_campaign": "You have successfully created a campaign width ID: 45!"
      }
   ]
}

{
   "errors": [
      {
         "timezone": "Time Zone is not correct!"
      },
      {
         "start_date": "A start date must be before end date!"
      }
   ]
}
```

7. CREATE CODES

This method is used to get the generated codes for the campaign with the specified campaign ID. The endpoint returns both the codes in the codes array as well as UUID pair which can be used to distinguish individual codes on client end. The UUID has no monetary value and can be used to refer to the reward code without having to show it directly.

Request

Method	URL
POST	api/codes/create

Type	Params	Values
Header	Authorization: Bearer	access_token

Type	Params	Values
POST	campaign_id	integer
POST	brands	array
	id	integer
	denomination	float
	quantity	integer
	secret (optional)	boolean (true/false)
	secret_activation_time (optional)	double
	client_id (optional)	integer
	expiration_days (optional)	double

access_token
access_token must be sent with all client requests. The access_token helps the server to validate the request source.

campaign_id
The ID of the campaign for which these codes are being generated for.

`brands`
The array of brands used in the campaign for which codes need to be generated. For each brand an `id` for the brand, a `denomination` and `quantity` are required.

`secret (optional)`
Secret is a true or false value which determines if the code generated will have an additional 2 factor authentication code when redeemed. This is an optional parameter.

`secret_activation_time (optional)`
If the brand has secret set to true, then the secret_activation_time is the time (in hours) of when the redeemed code will be activated after the end user has requested a redemption of the code. An example where a code is created today with a secret code and a secret_activation_time of 24 (hours) means the following. If the code is created today and accessed by the end client tomorrow, the 24 hours will begin counting down from tomorrow (the time end user activated the reward). After the 24 hours wait time, the end client will be able to claim their reward. The minimum time is 0.25 of an hour or 15 minutes while there is no maximum.

`expiration_days (optional)`
This optional parameter controls how many days after redemption a code for a given brand will expire. For example if a brand is ordered with an expiration days value of 1, then the code will expire 1 day after being redeemed. Expiration in this context means that the end client will have a limited amount of time after redeeming the code to use said code.

`client_id`
This optional parameter is something which is required to place orders for some brands. The value here will be provided at Account Setup. In most cases the value will start from “1” and increase as more client_id’s are added for a specific brand.

Request example:

```
{
        "campaign_id":9,
        "brands":[
                {
                "id" : 999,
                "denomination": 1.01,
                "quantity": 2
                "secret" : true,
                "secret_activation_time": 0.25,
                "client_id": 29,
                “expiration_days”: 30
```

```
                    }
          ]
}
```

Response

Status	Response	Explanation
407	Unauthorized access1	Attempted to connect to live API with sandbox creds or sandbox API with live creds
422	The Campaign ID is required!	Request did not have a valid campaign_id passed
422	The brands or groups is required and must be array!	Request did not contain an array object of brands (or groups) or it was an empty array
422	Expiration days is not correct!	Request contained expiration_days and it was either not an interger or was null or was less then 1
422	A brand_id: xxx is not correct!	Request contained a brand_id which the requestor does not have access to, or is no longer available, or is temporarily unavailable
422	A brand_id: xxx with quantity: yyy is not correct!	Request contained a quantity which was not a number or was less than or equal to 0
422	A brand_id: xxx with denomination yyy is not correct!	Request contained a denomination less than or equal to 0 or a non-numeric denomination
422	A brand_id: xxx with secret: zzz is not correct!	Request contained secret but it was not Boolean value (did not pass true/false)

422	A brand_id xxx with secret activation time yyy is not correct!	Request contained activation_time but it was less than or equal to 0 or it was non-numeric
422	A brand_id: xxx with secret activation time: yyy is below the minimum of 15 minutes, which corresponds to (0.25)!	Request had activation_time and it was numeric but was under the minimum allowed of 0.25
422	A brand_id: xxx with secret activation time: yyy must have secret!	Request had activation_time but did not specify secret: true. To you the functionality for activation time secret must be true
422	A brand_id xxx is not correct!	Request contained a brand_id but the brand was not found. This error can also be thrown if a campaign is not found or if a fund is not found for the specified campaign or if attempting to use a brand with one currency in a campaign with a different currency
422	The Campaign is not active!	The campaign is not active. A campaign must be active before codes can be crated for it. A campaign can be activated either by the interface or the campaign/activate endpoint. By default order-now campaigns (which is what /crateordernow uses are active)
422	A brand_id: xxx with quantity: yyy is not correct! Available: ccc	Some brands (very few) are stocked brands where if the request attempts to exceed the number of stocked rewards you will receive this error. For example, if the system only had 2 Amazon gift cards for a specific denomination but the request

		asked for 5, then the user will receive this error
422	A brand_id: xxx requires additional parameter client_id!	For this request to succeed the brand requires the additional parameter client_id. In cases where the brand does NOT require a client_id then this will be ignored
422	A brand_id: xxx with parameter client_id vvv is not correct!	The client_id parameters are provided, usually per campaign and can be located in the dashboard interface. If a client_id which does not exist is is not attributed to the user making the request is found, then the user will receive this error.
422	A campaign_id is not correct!	This error occurs when a project is not found for this campaign/request
422	A brand_id: xxx with denomination: yyy is not correct!	This error occurs when the denomination sent by the request is outside the bounds of the brand. For example is brand xxx has denomination range $1 to $100 and the request attempts to make a $200 order, the user will see this error
403	You do not have enough money on this program	There was not enough in the fund to cover the codes which are to be generated
403	You have entered an invalid quantity! Available : xxx	This can happen for stocked brands where at the time of creating rewards there are not enough available to fulfill the ordered quantity
406	There was an error processing your request, please contact customer support	There was an unexpected error when attempting to process your request. No order is placed.

500	Service Unavailable	The request timed out because of server issues. No order was placed

Response example:

```
{
          "success": [
          {
          "denomination": "0.11",
          "currency": "USD",
          "codes": [
             "80cbdeae755c97e61e47918addb1a268f36aca6d",
             "df7a458d8314f8bcac2803d47647bcd711e8a43d"
                   ],
                   "uuids": {
                      "80cbdeae755c97e61e47918addb1a268f36aca6d": "462e14a70135493e89b39fa6cd1364c3202102051036531",
                      "df7a458d8314f8bcac2803d47647bcd711e8a43d": "83cab43947d05f43267e3fb47c337834202102051036532"
                   },
               "codeSecretOffset": {
                         "80cbdeae755c97e61e47918addb1a268f36aca6d": 0.25,
              "80cbdeae755c97e61e47918addb1a268f36aca6d": 0.25
                      },
                      "codeSecret": {
                         "80cbdeae755c97e61e47918addb1a268f36aca6d ": "REXWJ6n0fJ",
                 "df7a458d8314f8bcac2803d47647bcd711e8a43d”: "AER1JbO6f9"
                      },

                   "order_id": 6149
                }
             ]
          }
```

8. GET CAMPAIGNS

Gets all the Campaigns that the authenticated user has access to.

Request

Method	URL
GET	api/campaigns

Type	Params	Values
Header	Authorization: Bearer	access_token

access_token

access_token must be sent with all client requests. The access_token helps the server to validate the request source.

Response

Status	Response
200	The rest API will respond with an array of JSON objects
308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
[
  {
        "id": 34,
        "name": "api campaign 1",
        "description": null,
        "created_at": {
            "date": "2017-12-27 12:58:53.000000",
            "timezone_type": 3,
            "timezone": "UTC"
        },
        "status": "pending",
        "start_date": "2017-12-27 13:00:00",
        "end_date": "2017-12-31 13:00:00",
        "timezone": "Eastern Standard Time",
        "project_id": 1,
        "fund_id": 1
    },
    {
        "id": 35,
        "name": "api campaign 1",
        "description": null,
        "created_at": {
            "date": "2017-12-27 12:59:11.000000",
            "timezone_type": 3,
            "timezone": "UTC"
        },
        "status": "pending",
        "start_date": "2017-12-27 13:00:00",
        "end_date": "2017-12-11 13:00:00",
        "timezone": "Eastern Standard Time",
        "project_id": 1,
        "fund_id": 1
    }
]
```

9. ACTIVATE CAMPAIGN

Activate a campaign so that codes for this campaign can be generated.

Request

Method	URL
`POST`	`api/campaign/active`

Type	Params	Values
`Header`	`Authorization: Bearer`	`access_token`

Type	Params	Values
`POST`	`campaign_id`	`integer`

`access_token`

`access_token` must be sent with all client requests. The access_token helps the server to validate the request source.

`campaign_id`

`campaign_id` is the ID of the campaign that needs to be activated. A campaign needs to be activated before any codes can be generated for this campaign.

Request example:

```
{
   "campaign_id":46
}
```

Response

Status	Response
`200`	The rest API will respond with a JSON objects containing the following properties: - success

308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
{
  "success": {
      "id": 46,
      "name": "SANDBOX api campaign 1",
      "description": null,
      "expiration_days": 331,
      "created_at": "2018-01-04 15:18:09",
      "updated_at": "2018-01-05 10:36:18",
      "status": "active",
      "start_date": "2018-01-04 13:00:00",
      "end_date": "2018-12-01 13:00:00",
      "timezone": "Eastern Standard Time",
      "project_id": 1,
      "fund_id": 1
    }
}
```

10. ORDER NOW

Lists all the orders from Order Now functionality. The Order Now type orders are those which are eligible to be used with the email functionality available in the API. To create an Order Now type order, use the /api/createordernow endpoint.

Request

Method	URL
GET	api/ordernow

Type	Params	Values
Header	Authorization: Bearer	access_token

access_token

access_token must be sent with all client requests. The access_token helps the server to validate the request source.

Response

Status	Response
200	The rest API will respond with a JSON objects containing the following properties: - success
308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
[
    {
        "id": 1,
        "total_cost": "$5.50",
        "payment_method": "Draw-Down Account",
        "purchase_order_number": "66",
        "brands": [
            {
                "brand_id": 2,
                "brand_name": "Amazon",
                "amount": "$1.00",
                "total_rewards": "2",
                "rewards_available_to_send": 2,
                "rewards_available_to_add_email": 2
            },
            {
                "brand_id": 23,
                "brand_name": "CVS",
                "amount": "$10.00",
                "total_rewards": "1",
                "rewards_available_to_send": 1,
                "rewards_available_to_add_email": 1
            }
        ]
    }
]
```

11. CREATE ORDER NOW

This method is used to create an order which can use the email functionality enabled in the API.

Request

Method	URL
POST	api/createordernow

Type	Params	Values
Header	Authorization: Bearer	access_token

Type	Params	Values
POST	purchase_order_number	string
POST	custom1	string
POST	brands: id: denomination: quantity: secret (optional): secret_activation_time (optional): expiration_days: (optional)	Array integer double integer boolean (true/false) double double
POST	client_id:	integer (optional)

access_token

access_token must be sent with all client requests. The access_token helps the server to validate the request source.

purchase_order_number

purchase_order_number is an optional integer value to pass when creating an order now which can help the user track the different order now orders in the system.

custom1

custom1 is an optional string field where user can enter any data they wish to pass to the

NeoCurrency system about an order now order. This information can help the user track a specific order now order in the system.

`brands`
The array of brands used in the campaign for which codes need to be generated. For each brand an `id` for the brand, a `denomination` and `quantity` are required. The `id` is from the /brands endpoint, the possible values for denomination and quantity are governed by the brand id and what is available for this brand via /brands endpoint.

`secret (optional)`
Secret is a true or false value which determines if the code generated will have an additional 2 factor authentication code when redeemed. This is an optional parameter.

`secret_activation_time (optional)`
If the brand has secret set to true, then the secret_activation_time is the time (in hours) of when the redeemed code will be activated after the end user has requested a redemption of the code. An example where a code is created today with a secret code and a secret_activation_time of 24 (hours) means the following. If the code is created today and accessed by the end client tomorrow, the 24 hours will begin counting down from tomorrow (the time end user activated the reward). After the 24 hours wait time, the end client will be able to claim their reward. The minimum time is 0.25 of an hour or 15 minutes while there is no maximum.

`expiration_days (optional)`
This optional parameter controls how many days after redemption a code for a given brand will expire. For example if a brand is ordered with an expiration days value of 1, then the code will expire 1 day after being redeemed. Expiration in this context means that the end client will have a limited amount of time after redeeming the code to use said code.

`client_id`
This optional parameter is something which is required to place order now orders for some brands. The value here will be provided at Account Setup. In most cases the value will start from “1” and increase as more client_id’s are added for a specific brand.

Request example:

```
{
        "purchase_order_number": "66",
        "custom1": "something",
        "brands": [
     {
        "id": 2,
        "denomination": 1.00,
        "quantity": 2,
        “expiration_days”: 20.5,
     },
     {
        "id": 23,
        "denomination": 10.00,
        "quantity": 1
     }
 ]
}
```

Request example 2:

```
{
  "brands" : [ {
    "id" : 623,
    "denomination" : 0.05,
    "quantity" : 1,
    "secret" : true,
    "secret_activation_time": 0.5
  } ]
}
```

Response

Status	Response	Explanation
401	Unauthorized access1	Attempted to connect to live API with sandbox creds or sandbox API with live creds
422	The Campaign ID is required!	Request did not have a valid campaign_id passed
422	The brands or groups is required and must be array!	Request did not contain an array object of brands (or groups) or it was an empty array
422	Expiration days is not correct!	Request contained expiration_days and it was either not an interger or was null or was less then 1
422	A brand_id: xxx is not correct!	Request contained a brand_id which the requestor does not have access to, or is no longer available, or is temporarily unavailable

422	A brand_id: xxx with quantity: yyy is not correct!	Request contained a quantity which was not a number or was less than or equal to 0
422	A brand_id: xxx with denomination yyy is not correct!	Request contained a denomination less than or equal to 0 or a non-numeric denomination
422	A brand_id: xxx with secret: zzz is not correct!	Request contained secret but it was not Boolean value (did not pass true/false)
422	A brand_id xxx with secret activation time yyy is not correct!	Request contained activation_time but it was less than or equal to 0 or it was non-numeric
422	A brand_id: xxx with secret activation time: yyy is below the minimum of 15 minutes, which corresponds to (0.25)!	Request had activation_time and it was numeric but was under the minimum allowed of 0.25
422	A brand_id: xxx with secret activation time: yyy must have secret!	Request had activation_time but did not specify secret: true. To you the functionality for activation time secret must be true
422	A brand_id xxx is not correct!	Request contained a brand_id but the brand was not found. This error can also be thrown if a campaign is not found or if a fund is not found for the specified campaign or if attempting to use a brand with one currency in a campaign with a different currency

422	The Campaign is not active!	The campaign is not active. A campaign must be active before codes can be crated for it. A campaign can be activated either by the interface or the campaign/activate endpoint. By default order-now campaigns (which is what /crateordernow uses are active)
422	A brand_id: xxx with quantity: yyy is not correct! Available: ccc	Some brands (very few) are stocked brands where if the request attempts to exceed the number of stocked rewards you will receive this error. For example, if the system only had 2 Amazon gift cards for a specific denomination but the request asked for 5, then the user will receive this error
422	A brand_id: xxx requires additional parameter client_id!	For this request to succeed the brand requires the additional parameter client_id. In cases where the brand does NOT require a client_id then this will be ignored
422	A brand_id: xxx with parameter client_id vvv is not correct!	The client_id parameters are provided, usually per campaign and can be located in the dashboard interface. If a client_id which does not exist is is not attributed to the user making the request is found, then the user will receive this error.
422	A campaign_id is not correct!	This error occurs when a project is not found for this campaign/request
422	A brand_id: xxx with denomination: yyy is not correct!	This error occurs when the denomination sent by the request is outside the bounds of the

		brand. For example is brand xxx has denomination range $1 to $100 and the request attempts to make a $200 order, the user will see this error
403	You do not have enough money on this program	There was not enough in the fund to cover the codes which are to be generated
403	You have entered an invalid quantity! Available : xxx	This can happen for stocked brands where at the time of creating rewards there are not enough available to fulfill the ordered quantity
401	There was an error processing your request, please contact customer support	There was an unexpected error when attempting to process your request. No order is placed.
500	Service Unavailable	The request timed out because of server issues. No order was placed

Response example:

```
{
   "success": {
      "order_id": 6208,
      "custom1": null,
      "data": [
         {
            "campaign_brand_id": 10031,
            "denomination": "0.05",
            "currency": "USD",
            "codes": [
               "c613e814692e94906be3d8f592551894e5bd1f85"
            ],
            "uuids": {
               "c613e814692e94906be3d8f592551894e5bd1f85": "3520f5a13757d26d717ac35c5
c1f0d7f202103111028291"
            },
            "codeSecretOffset": {
               "c613e814692e94906be3d8f592551894e5bd1f85": 0.5
            },
            "codeSecret": {
                    "c613e814692e94906be3d8f592551894e5bd1f85": "REXWJ6n0fJ"
                }
            }
        ]
    }
}
```

12. ORDER NOW ADD EMAILS

This method is used to create an order which can use the email functionality enabled in the API.

Request

Method	URL
POST	api/addemails

Type	Params	Values
Header	Authorization: Bearer	access_token

Type	Params	Values
POST	order_id	integer
POST	campaign_brand_id	integer
POST	emails	array with first name, last name, email, subject, custom message. Subject and Custom Message are optional

access_token

access_token must be sent with all client requests. The access_token helps the server to validate the request source.

order_id

order_id is the ID of the Order Now order for which user wishes to send emails for. To get all eligible Order Now order_ids use /api/ordernow. A single Order Now order maybe have multiple brands attached, so emails are loaded and sent by brand.

`campaign_brand_id`

`campaign_brand_id` is the campaign brand ID received from the success message of the createordernow endpoint. The campaign brand ID is an amalgamation which signifies a particular brand and denomination pair. These integer values are unique and non-repeating.

`emails`

`emails` is an array which holds first name, last name, email, subject and custom message for each code. The subject and custom message parameters are optional. If they are used, they will overwrite the email subject and default message to user. For custom message, we only allow text messages and do not, in this version, allow html or links in the message. For an example of usage, please see the Request example below.

Request example:

```
{
        "order_id": 1,
        "campaign_brand_id": 2,
        "emails": [
                {
                        "email": "john@test.com",
                        "first_name": "John",
                        "last_name": "Doe",
                        "subject": "Congratulations Winner",
                        "message":"This will be some string text message that will be displayed."
                }
        ]
}
```

Response

Status	**Response**
`200`	The rest API will respond with a JSON objects containing the following properties: - success
`308`	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.

401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
{
   "success": "You have successfully added one email!"
}
```

13. ORDER NOW SEND EMAILS

This method is used to create an order which can use the email functionality enabled in the API.

Request

Method	URL
POST	api/sendemails

Type	Params	Values
Header	Authorization: Bearer	access_token

Type	Params	Values
POST	order_id	integer
POST	campaign_brand_id	integer
POST	language	String (optional)
POST	email_template_id	integer

`access_token`
`access_token` must be sent with all client requests. The access_token helps the server to validate the request source.

`order_id`
`order_id` is the ID of the Order Now order for which user wishes to send emails for. To get all eligible Order Now order_ids use /api/ordernow. A single Order Now order maybe have multiple brands attached, so emails are loaded and sent by brand.

`campaign_brand_id`
`campaign_brand_id` is the campaign brand ID received from the success message of the createordernow endpoint. The campaign brand ID is an amalgamation which signifies a particular brand and denomination pair. These integer values are unique and non-repeating.

`language`
The optional `language` parameter allows emails to be sent in different languages. Examples include “en” for English and “fr” for French. Languages supported is based on Account Setup.

`email_template_id`
The optional `email_template_id` parameter allows emails to be sent using a pre-defined template. This template is created and curated in the web portal portion of the NeoCurrency system and can be called via id when sending emails via the API. This parameter will return a 422 type error if the user does not have access to call the specified `email_template_id`.

Request example:

```
{
        "order_id": 5041,
        "campaign_brand_id": 297
}
```

Request example 2:

```
{
        "order_id": 5106,
        "campaign_brand_id": 250,
        "language": "fr"
}
```

Request example 3:

```
{
        "order_id": 5106,
        "campaign_brand_id": 250,
        "email_template_id": 76
}
```

Response

Status	Response
200	The rest API will respond with a JSON objects containing the following properties: - success
308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
{
   "send": 2
}
```

14. ORDER NOW LOAD PHYSICAL CARDS

This method is used to create an order which can use the email functionality enabled in the API.

Request

Method	URL
POST	api/bulkuploadaddresses

Type	Params	Values
Header	Authorization: Bearer	access_token

Type	Params	Values
POST	order_id	integer
POST	campaign_brand_id	integer
POST	people	Array
	first_name	String
	last_name	String
	address_1	String
	address_2	String
	city	String
	state	String
	zip	String
	carrier_line_1	String
	carrier_line_2	String
	fourth_line	String

access_token

access_token must be sent with all client requests. The access_token helps the server to validate the request source.

order_id

order_id is the ID of the Order Now order for which user wishes to send emails for. To get all eligible Order Now order_ids use /api/ordernow. A single Order Now order maybe have multiple brands attached, so emails are loaded and sent by brand.

campaign_brand_id

campaign_brand_id is the campaign brand ID received from the success message of the createordernow endpoint. The campaign brand ID is an amalgamation which signifies a particular brand and denomination pair. These integer values are unique and non-repeating.

people

people is the array of contact information which will be used to create and send the physical cards. The information required includes contact first and last names and full address. Carrier_line_1 and carrier_line_2 are two customizable fields which will display on the card carrier letter. Fourth_line is a customizable message which displays on the card itself under the customer’s name.

Request example:

```
{
        "orderId":"5963",
        "campaign_brand_id":"9429",
        "people": [
        {
                "first_name":"John",
                "last_name": "Doe",
                "address_1": "123 Street",
                "address_2": "Apt 1a",
                "city": "New York",
                "state": "NY",
                "zip": "00001",
                "carrier_line_1":"message 1",
                "carrier_line_2":"message 2",
                "fourth_line":"text under name on card"
        }
      ]
}
```

Response

Status	Response
200	The rest API will respond with a JSON objects containing the following properties: - success
308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
{
        "success"
}
```

15. ORDER NOW REDEEM PHYSICAL CARDS

This method is used to redeem and send the physical cards to the physical addresses loaded in the system using /api/bulkuploadaddresses.

Request

Method	URL
POST	api/redeemphysicalcards

Type	Params	Values
Header	Authorization: Bearer	access_token

Type	Params	Values
POST	order_id	integer
POST	campaign_brand_id	integer

access_token

access_token must be sent with all client requests. The access_token helps the server to validate the request source.

order_id

order_id is the ID of the Order Now order for which user wishes to send emails for. To get all eligible Order Now order_ids use /api/ordernow. A single Order Now order maybe have multiple brands attached, so emails are loaded and sent by brand. Do note that only orders which have physical brands are eligible for this endpoint. If the order does not contain a physical brand and error will be returned when hitting this endpoint.

`campaign_brand_id`

campaign_brand_id is the campaign brand ID received from the success message of the createordernow endpoint. The campaign brand ID is an amalgamation which signifies a particular brand and denomination pair. These integer values are unique and non-repeating.

Request example:

```
{
        "order_id": 5041,
        "campaign_brand_id": 297
}
```

Response

Status	Response
200	The rest API will respond with a JSON objects containing the following properties: - success
308	The rest API will respond with such a code in the event of attempting to connect to the rest API via http instead of https.
401	The rest API will respond with a JSON object containing the following properties: - error: “Unauthenticated.”
500	The rest API will respond with a JSON object containing the following properties: - error: ”500 Internal Server Error.”

Response example:

```
{
        "success"
}
```

CODE ACTIVATION

Codes from the sandbox NeoCurrency system can be activated much in the same way that the live system can activate codes. The endpoint for code activation while not directly an API endpoint is https://redeem.yourdigitalreward.com/sandbox/activate-code/{code} .

GLOSSARY

Dynamic Fields

This a notation which allows the user to pass specific dynamic reward data into the email templates used for sending email rewards in the NeoCurrency system. These fields are denoted by using hash signs (#) so for example #firstname# will dynamically pull in the first name of the customer. This way it’s possibly to dynamically send out personalized email templates without having to create multiple email templates for different customers/denominations etc. A list of acceptable dynamic fields is below:

#firstname# - the first name of the customer if a name has been uploaded.
#fullname# - the full name of the customer, this is equivalent to #firstname# #lastname#.
#lastname# - the last name of the customer if a name has been uploaded.
#rewardbrand# - the name of the brand the customer is receiving.
#currency# - the currency symbol of the reward the customer is receiving (example $ or € or £).
#rewardvalue# - the denomination value of the reward customer is receiving.

An example of API usage using the /api/sendemails endpoint would be as follows

```
{
        "order_id": 1234,
        "campaign_brand_id": 111,
        "subject": "Thank you #firstname#, here is a #currency##rewardvalue# gift",
        "under_header": "#fullname# thank you for being awesome",
        "under_button": “Mr. #lastname#, please visit us again soon!",
        "language": "de"
}
```

Client_ID Discussion

When placing orders, some brands, require an additional parameter called client_id. This is provided to the user ahead of time. The client_id does not change and is tied to the specific user account. Furthermore, brands have affected brands have different client_id's between sandbox and live API environments. The client_id is used when making requests for rewards from specific brands and should be passed to both create_codes or createordernow endpoints. Failure to pass this parameter will result in an error stating that this parameter is required for the specific brand. In most cases usually Visa and Mastercard reward cards are the brands which will need this additional parameter. Below are all current brands which require client_id (note a single account may not have access to all these brands. We will try to keep the list amassed here up to date).

Brand				
608	671	672	673	674
674	676	677	678	682
695	804	857	860	861
862	863	864	865	866
867	868			

Cross Environment Testing

For some clients, we have the option to enable Cross Environment Testing. This means that the client can create campaigns in their live environment and when a campaign is created a duplicate campaign is created in their Sandbox environment. Furthermore, when placing sandbox testing codes, the client can use their live Campaign ID in the sandbox codes/create endpoint and the client will receive sandbox codes while using a live ID (for testing). The idea behind this feature is to ease campaign creation and testing for clients between their live and sandbox accounts. Do note that this functionality will ONLY work for newly created live campaigns. To enable the feature, please contact your reward manager.

Conventions

- **Client** - Client application.
- **Status** - HTTP status code of response.
- All the possible responses are listed under 'Responses' for each method. Only one of them is issued per request server.
- All responses are in JSON format.
- All request parameters are mandatory unless explicitly marked as `[optional]`

Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

`2XX` - Success of some kind
`3XX – Redirect of some kind`
`4XX` - Error occurred in client's part
`5XX` - Error occurred in server's part

Status Code	Description
200	OK
201	Created
202	Accepted (Request accepted, and queued for execution)
308	Permanently moved
400	Bad request
401	Authentication failure
403	Forbidden
404	Resource not found
405	Method Not Allowed
409	Conflict
413	Request Entity Too Large
500	Internal Server Error
501	Not Implemented
503	Service Unavailable