Make a CharityChoice
choose from over 2500 charities
When meaningful matters...

CharityChoice REST API

VERSION: 1.4.1

Base URI: https://www.charitygiftcertificates.org/api

API Endpoints List – Quick Reference

The base URI for all calls is **https://www.charitygiftcertificates.org/API**

Function	Method	Endpoint URI https://www.charitygiftcertificates.org/api …	Description
Activate Codes	POST	/codes/activate	Acquire active CharityChoice redemption codes.
Activate Supplied Codes	POST	/codes/activateCodes	Active the given list of codes for redemption of the given amount.
Charity Donation	POST	/donate	Donate to one or more charities.
Charity Information	GET	/charity/charityId:int	Gets information for a particular charity. The charityId can be obtained from a call to either Get Charity List or Search for a Charity.
De-activate Codes	DELETE	/codes/remove/codes:string	De-activate one or more redemption codes. "codes" is a comma separated string.
Donate Retailer Gift Card	POST	/dyc/donate	Donate a retailer gift card to charity.

Generate and Reserve Codes	GET	/codes/generateCodes/prefix:string/numberOfCodes:int	Acquire a list of valid non-active CharityChoice redemption codes. These codes can be activated afterwards by doing an Activate Supplied Codes request. The generated codes are guaranteed to stay "reserved" for 30 days from the time of generation.
Get Category List	GET	/charity/cats/list	Gets a list of the CharityChoice Charity Categories.
Get Charity List	GET	/charity/list/listName:string?/categoryId:int?	Gets a list of charities. The listName (optional) can be "main", "local", or "all" (default is "all"). The categoryId (optional) is one of the CharityChoice category ids. The list of categories can be acquired from an API call to Get Category List.
Get Code Status	GET	/codes/status/redemptionCode:regex(^[a-zA-Z]*\d+$)	Get the status of the given redemption code. Only allowed for codes that are in your account unless you have special permissions.
Get Code Status	POST	/codes/status	Gets the status of multiple redemption codes. Only allowed for codes that are in your account unless you have special permissions.
Get Gift Card Status	GET	/dyc/donorId:int/cardId:int	Get the current status and basic information of a previously donated Retailer Gift Card
Get Order Types	GET	/donate/ordertypes	Gets a list of CharityChoice order types.
Get Orders	GET	/account/orders/orderTypeId:int?	Gets a list of orders in account. If the optional orderTypeId is supplied, only returns orders of that type.
Get Orders Report	GET	/account/orders/report/fromDate:date/toDate:date	Gets a report of orders in your account within a range of dates, with full order details - including redemption information.
GetDigitalCardColors	GET	/digital/digitalCardColors	Gets a list of the CharityChoice Digital Cards Color Schemes.
GetDigitalCardsList	GET	/digital/digitalCardsList	Gets a list of the CharityChoice Digital Cards Types and images.
Gets Physical Card Types	GET	/physical/cardTypes	Gets a list of Physical Card Types with which denominations are available and how much is available in stock.
Gets Shipping Types	GET	/physical/shippingTypes	Gets a list of Shipping Types for physical cards orders. Includes pricing information.
Gift Card Retailer Information	GET	/dyc/retailer/retailerId:int	Gets the information for a gift card retailer. The retailerId is a CharityChoice retailer id. The list of retailers can be obtained by calling Gift card Retailer List.

Gift Card Retailers List	GET	/dyc/retailer/list	Gets a list of the Gift Card Retailers that CharityChoice is able to accept for donations.
Order Physical Cards	POST	/physical	Order any number of Physical Charity Gift Cards and have them shipped to you or directly to the final recipient.
Order Status	GET	/donate/orderId:int	Returns a list of codes in this order with their redemption status.
Redeem a Code	POST	/codes/redeem	Redeem a redemption code. To redeem a code that is not in your account, you will need special permissions. Please contact CharityChoice for details.
Remove Redemption	Delete	/codes/redeem/redemptionCode:regex(^[a-zA-Z]*\d+$)}	Removed previous redemption from a redemption code. You can only remove redemption information from a code that is in your account.
Search for a Charity	GET	/charity/list/search/term:string/?includeDescription=bool	Searches for Charities by the given search term. If the optional includeDescription=true the search includes the charities short descriptions.
Send Digital Cards	POST	/digital/send	Send one or more redeemable Digital Cards. Cards can be delivered to the recipient either by email or sms.
Void Order	DELETE	/donate/orderId:int	Void a previously made order. Only allowed for orders that are in your account. USE WITH CAUTION! Removes all order information. If any of the codes were redeemed, the redemption information is removed as well. The orderId is returned from any calls to Activate Codes, Charity Donation, Send Digital Cards, and Donate Retailer Gift Card. You can also find the orderId on your account page on the CharityChoice site.

Table of Contents

API Endpoints List – Quick Reference 1

Introduction 8

Requirements for all API Requests 9

Permissions 9

Request Headers 9

Development & Testing 11

SDK's - Client Code Implementation and Sample Code 11

Node.js 11

.Net 12

Python 12

Run API calls on the fly 13

GET ORDERS 14

Request Notes: 14

Response Data: 14

GET ORDERS REPORT 15

Request Notes: 15

Response Data: 15

CHARITY INFORMATION 17

Request Notes: 17

Response Status Codes: 17

Response Data: 17

Response Notes: 17

GET CHARITY LIST 18

Request Notes: 18

Response Data: 18

Response Notes: 18

SEARCH FOR A CHARITY 19

Request Notes: 19

Response Data: 19

Response Notes: 19

GET CATEGORY LIST 20

Response Data: 20
CHARITY DONATION 21
Request: 21
Request Notes: 21
Response Status Codes: 21
Response Data: 21
ORDER STATUS 22
Request Notes: 22
Response Status Codes: 22
Response Data: 22
GIFT CARD RETAILERS LIST 23
Response Data: 23
GIFT CARD RETAILER INFORMATION 24
Request Notes: 24
Response Status Codes: 24
Response Data: 24
DONATE RETAILER GIFT CARD 25
Request: 25
Request Notes: 25
Response Status Codes: 26
Response Data: 26
Response Notes: 26
Special Note: 26
GET GIFT CARD STATUS 27
Request Notes: 27
Response Status Codes: 27
Response Data: 27
ACTIVATE CODES 28
Request: 28
Request Notes: 28
Response Status Codes: 28
Response Data: 28
Response Notes: 29

Sample Request: 30
Sample Response: 30
ACTIVATE SUPPLIED CODES 33
Request: 33
Request Notes: 33
Response Status Codes: 33
Response Data: 33
DE-ACTIVATE CODES 34
Request Notes: 34
Response Status Codes: 34
Response Data: 34
Response Notes: 34
SEND DIGITAL CARDS 35
Request: 35
Request Notes: 35
Response Status Codes: 36
Response Data: 36
Response Notes: 36
ORDER PHYSICAL CARDS 37
Request: 37
Request Notes: 38
Response Status Codes: 38
Response Data: 38
Response Notes: 39
GET PHYSICAL CARD TYPES 40
Response Data: 40
GET SHIPPING TYPES 41
Response Data: 41
Response Notes: 41
GENERATE AND RESERVE REDEMPTION CODES 42
Request Notes: 42
Response Data: 42
Response Notes: 42

GET CODE STATUS 43
Request Notes: 43
Response Status Codes: 43
Response Data: 43
Response Notes: 43
GET CODES STATUS 44
Request 44
Response Status Codes: 44
Response Data: 44
Response Notes: 44
VOID ORDER 45
Request Notes: 45
Response Status Codes: 45
Response Data: 45
REDEEM A CODE 46
Request: 46
Request Notes: 46
Response Status Codes: 46
Response Data: 46
REMOVE REDEMPTION 47
Request Notes: 47
Response Status Codes: 47
Response Data: 47
GET ORDER TYPES 48
Response Data: 48
GET DIGITAL CARDS LIST 49
Response Data: 49
Response Notes: 49
GET DIGITAL CARD COLORS 50
Response Data: 50
Response Notes: 50

Introduction

This is the documentation for the CharityChoice Rest API.

The CharityChoice Rest API allows an authorized client to access much of the functionality that CharityChoice supplies to the end user from the main CharityChoice web site of https://www.charitygiftcertificates.org/.

It also includes some special account-related functions and lookup functions to help alleviate the need for API users to store any CharityChoice data.

The base URI for all REST API calls is:

https://www.charitygiftcertificates.org/api

Requirements for all API Requests

Permissions

The API can only be accessed by accounts that are given explicit authorization permissions.

To acquire permissions to access the CharityChoice API, please contact us at admin@charity-choice.org.

Request Headers

The following http headers need to be added to each and every API call:

1. Header: ***Authorization*** Value: ***basic* AUTHORIZATION_KEY** (see explanation below)
2. Header: ***cgc-app-id*** Value: ***CGC_REST_API_VERSION_0142*** (may change in the future)
3. Header: ***cgc-mode*** Value: ***prod*** OR ***dev*** If you use "*prod*", requests will be directed to the live production database. If you use "*dev*", your requests will be directed to the development "sandbox" environment.

Authorization Key

The authorization key is composed of a base 64 string made up as follows:
UserId:ClientId:Password

- **UserId** is supplied by CharityChoice when you receive approval to use the API.
- **ClientId** is supplied by CharityChoice when you receive approval to use the API.
- **Password** is the same as your charity choice account password.

So for example, if your UserId is 45885 and your ClientId is 8960 and your password is %%6453De1_223, you can retrieve your AUTHORIZATION_KEY with one of the following code snippets:

Java

```
import java.util.Base64;

String authorization = new
String(Base64.getEncoder().encode("45885:8960:%%6453De1_223".getBytes()));
```

C#

```
string authorization =
Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes("45885:8960:%%6453De1_223")
);
```

Python

```
import base64

authorization = base64.b64encode(b'45885:8960:%%6453De1_223').decode()
```

Ruby

```
require "base64"

authorization = Base64.strict_encode64('45885:8960:%%6453De1_223')
```

Javascript

```
var authorization = btoa("45885:8960:%%6453De1_223");
```

Javascript - Node.js

```
var authorization = new Buffer("45885:8960:%%6453De1_223").toString('base64');
```

The base 64 string that should be returned for all of the above is:
NDU4ODU6ODk2MDolJTY0NTNEZTFfMjIz

So, in this case, the authorization header would look like:

authorization:basic NDU4ODU6ODk2MDolJTY0NTNEZTFfMjIz

Development & Testing

During the development process, you can use the CharityChoice development “sandbox” environment by setting “dev” as the value of the **cgc-mode** request header (see above).

Any orders, donations etc. made using in “dev” mode will not show in your account on www.CharityGiftCertificates.org.

SDK’s - Client Code Implementation and Sample Code

We have created a few SDK’s (client code implementations) and examples to help make your integration easier.

Node.js

The Node.js SDK contains call wrappers for all of the API calls described in this document.

The SDK can be added to you project by running:

```
npm i charitychoice_api –save
```

OR

```
yarn add charitychoice_api
```

To use the SDK in your code, use:

```
 const uid = 0000, // Your CharityChoice User Id
   cid = 0000, // Your CharityChoice Client Id
   pw = '****', // Your CharityChoice Password
   apiModule = require('charitychoice_api'),
   api = new apiModule.CharityChoiceApi(id, cid, pw);

//Example call to get Gift Card Retailers List
api.giftCardRetailersList(
   data => console.log(data), //On success callback
   data => console.warn(data), //On fail callback
   data => console.error(data)); //On error callback
```

More example code of how to use the different calls can be found in the package in the file: *UsageExample.js.*

.Net

The .Net SDK contains call wrappers for all of the API calls described in this document.

NuGet package **CharityChoice.ApiClient**.

To install the package use:

```
Install-Package CharityChoice.ApiClient
```

OR

```
dotnet add package CharityChoice.ApiClient
```

Note:

Calling the API functions is done by running the static functions found in the ''APICaller" class.

Before doing any API calls, your code needs to set the following 3 static properties:

1. CharityChoiceApi. ApiCaller. UserId
2. CharityChoiceApi. ApiCaller. ClientId
3. CharityChoiceApi. ApiCaller. Password

You can download the source code and usage example code from:
https://www.charitygiftcertificates.org/API/Samples/DotNet/SourceCode_DoNetClient_CharityChoiceAPI.zip

You can also download the compiled DLL from:
https://www.charitygiftcertificates.org/API/Samples/DotNet/DoNetClient_CharityChoiceAPI.zip

Python

The python SDK contains call wrappers for all of the API calls described in this document.

You can acquire the python client implementation from
https://www.charitygiftcertificates.org/API/Samples/Python/CharityChoiceAPI_PythonClient.zip

Note:

Calling the API functions is done by running the functions found in the ''APICaller" class. See the helper file: "UsageExample.py" that shows how to call each function.

Run API calls on the fly

You can run API calls on-the-fly using the run API tool at https://www.charitygiftcertificates.org/API/.

NOTE: When doing POST calls, you will need to supply valid JSON data in the correct format into the “Data” field. Refer to this document for the required data and format for each call.

GET ORDERS

Gets a list of orders in your account.

GET **/account/orders/orderTypeId:int?**

Request Notes:

- The optional *orderTypeId*, if supplied, filters the orders to only those of that type. The list of order type Ids, can be acquired by running Get Order Types.

Response Data:

```
{
      OrderId :int
      PurchaseDate :datetime
      Type :string
      Denomination :float
      Quantity :int
      HideAmount :bool
      HideDonor :bool
      TotalForCards: float
      TotalForShipping: float
}
```

GET ORDERS REPORT

Gets a report of orders in your account with full order details - including redemption information.

To get a report of all orders within a range of dates use:

GET **/account/orders/report/fromDate:date/toDate:date**

To get a report for all the orders within a single month use:

GET **/account/orders/report/month:int/year:int**

Request Notes:

- When acquiring a report for range of dates, both the *fromDate* and *toDate* need to be supplied. The format should be MM-DD-YYYY (11-02-2025) or 02-NOV-2025.

Response Data:

```
{
    Succeeded: bool,
    Report: {
        Orders: [ {
            OrderId: int,
            OrderDate: DateTime,
            TotalForCards: float,
            TotalForShipping: float,
            TotalExtraForCustomPhysicalCards: float,
            TotalWithoutDiscounts: float,
            TotalWithoutDiscountsAtCheckout: float,
            DiscountPercentAtCheckout: float,
            FinalPriceAtCheckout: float,
            OrderItems: [ {
                OrderId: int,
                ItemId: int,
                Type: string,
                Denomination: float,
                Codes: [ {
                    CodeId: int,
                    ItemId: int,
                    RedemptionCode: string,
                    WasRedeemed: boolean,
                    RedeemDate: DateTime,
                    RedeemerFirstName: string,
                    RedeemerLastName: string,
                    RedeemerEmail: string,
```

```
                Charities: [ int ]
            } ]
        } ]
    } ]
},
ErrorMessage: string? }
```

CHARITY INFORMATION

Gets information of a particular charity

GET **/charity/charityId:int**

Request Notes:

- The *charityId* needs to be a valid CharityChoice charity Id. The list of Charities with their Ids, can be acquired by running Get Charity List or Search for a Charity.

Response Status Codes:

- **400** – The *charityId* supplied is not a valid CharityChoice *charityId*.
- **200** – The request was completed successfully. The response data will contain the charity information.

Response Data:

```
{
      Succeeded: bool,
      CharityInfo:
      {
            CharityId :int,
            CategoryId :int,
            CategoryName :string,
            CharityName :string,
            ShortDescription :string,
            LogoURL: string,
            URL: string,
            City: string,
            State: string,
            Country: string,
            IsLocal :bool
      },
      ErrorMessage: string?
}
```

Response Notes:

- The *LogoURL* (if available) is a URL to an image containing the charity Logo.
- The *URL* is the URL of the Charity's home page.
- *City, State* and *Country* are the location where the charity is based or registered.
- The *Country* is most often blank or null. This usually means a USA based charity.

GET CHARITY LIST

Gets a list of charities.

GET **/charity/list/listName:string?/categoryId:int?**

Request Notes:

- *listName* (optional) can be "main", "local", or "all" (default is "all")
- *categoryId* (optional) is one of the CharityChoice charity category ids. The list of charity categories can be acquired from an API call to Get Category List.

Response Data: (Array)

```
[{
      CharityId :int,
      CategoryId :int,
      CategoryName :string,
      CharityName :string,
      ShortDescription :string,
      LogoURL: string,
      URL: string,
      City: string,
      State: string,
      Country: string,
      IsLocal :bool
}]
```

Response Notes:

- The *LogoURL* (if available) is a URL to an image containing the charity Logo.
- The *URL* is the URL of the Charity's home page.
- *City, State* and *Country* are the location where the charity is based or registered.
- The *Country* is most often blank or null. This usually means a USA based charity.

SEARCH FOR A CHARITY

Searches for Charities by the given search text

GET **/charity/list/search/term:string/?includeDescription=bool?**

Request Notes:

- The *term* can be any part of any word in the charity name
- If the optional querystring parameter *includeDescription* is *true,* the search will also look for charities that their short description matches the search term.

Response Data: (Array)

```
[{
      CharityId :int,
      CharityName :string,
      CategoryId :int,
      CategoryName :string,
      ShortDescription :string,
      LogoURL: string,
      URL: string,
      City: string,
      State: string,
      Country: string,
      IsLocal :bool
}]
```

Response Notes:

- The *LogoURL* (if available) is a URL to an image containing the charity Logo.
- The *URL* is the URL to the Charity's home page.
- *City, State* and *Country* are the location where the charity is based or registered.
- The *Country* is most often blank or null. This usually means a USA based charity.

GET CATEGORY LIST

Gets a list of the CharityChoice Charity Categories.

GET **/charity/cats/list**

Response Data: (Array)

```
[{
      CategoryId :int,
      CategoryName :string
}]
```

CHARITY DONATION

Donate to one or more charities.

POST **/donate**

Request:

```
{
      Amount :float,
      CharityIds :[int],
      Optional DonorInformation: {
                  DonorFirstName :string?,
                  DonorLastName :string?,
                  DonorEmail :string,
                  DonorPhone :string?,
                  DonorMailingAddress :string?
            }
}
```

Request Notes:

- Up to three charities are allowed.
- The *Amount* can be any number (including floating point numbers) between 1 and 10,000.
- The *Amount* will be evenly divided among the charities.
- The *DonorInformation* field is optional. If it is not supplied, your CharityChoice account is set as the donor. If supplied, the supplied donor will be shown in your CharityChoice account as the "Redeemer" for this donation. Within the *DonorInformation*, only the *DonorEmail* cannot be left blank.

Response Status Codes:

400 – The donation failed. This usually indicates that the request was not in the required format. The *ErrorMessage* should contain more information about the failure. If the information is not helpful and this is a recurring response, please contact CharityChoice for assistance.
201 – The donation was completed successfully.

Response Data:

```
{
      Succeeded :bool
      Amount :float
      CharityIds :[int]
      OrderId :int
      ErrorMessage :string?
}
```

ORDER STATUS

Returns a list of codes in this order with their redemption status.

GET **/donate/orderId:int**

Request Notes:

- The *orderId* is returned from Activate Codes, Charity Donation and Donate Retailer Gift Card API requests. You can also find the *orderId* on your account page on the CharityChoice site.

Response Status Codes:

- **400** – The order id supplied is not a valid CharityChoice order id.
- **401** – You are not authorized to view this order. Usually due to the supplied orderId not being in your account.
- **200** – The request was completed successfully. The response data will contain a list of codes in this order.

Response Data:

```
{
      Succeeded :bool,
      Codes :[{
            Code :string,
            WasRedeemed: bool,
            RedeemDate: datetime
      }],
      ErrorMessage :string
}
```

GIFT CARD RETAILERS LIST

Gets a list of Card Retailers.

GET **/dyc/retailer/list**

Response Data: (Array)

```
[{
      RetailerId :int
      AcceptsEcode :bool,
      Name :string,
      IsCreditCard :bool,
      ECodePercent :float,
      PhysicalPercent :float,
      Minimum :float,
      Maximum :float,
      ValidationRules: [{
          CardNumberLength :int,
          PinLength :int
      }]

}]
```

GIFT CARD RETAILER INFORMATION

Gets the information for a gift card retailer.

GET **/dyc/retailer/retailerId:int**

Request Notes:

- The *retailerId* is a CharityChoice Gift Card Retailer id. The list of gift card retailers can be obtained by calling Gift Card Retailers List.

Response Status Codes:

- **400** – The *retailerId* supplied is not a valid CharityChoice Gift Card Retailer Id.
- **200** – The request was completed successfully. The response data will contain the retailer information.

Response Data:

```
{
      Succeeded: bool,
      RetailerInfo:
      {
            RetailerId :int
            AcceptsEcode :bool,
            Name :string,
            IsCreditCard :bool,
            ECodePercent :float,
            PhysicalPercent :float,
            Minimum :float,
            Maximum :float,
            ValidationRules: [{
                CardNumberLength :int,
                PinLength :int
            }]

      }
      ErrorMessage: string?
}
```

DONATE RETAILER GIFT CARD

Donate a retailer gift card to charity.

POST **/dyc/donate**

Request:

```
{
      RetailerId :int,
      CardNumber :string,
      PIN :string?,
      CVV :string?,
      ExpireDate :string?,
      Balance :float,
      CharityIds :[int],
      ContactInfo:
      {
            FirstName :string,
            LastName :string,
            Email :string,
            CompanyName :string?,
            Phone :string?,
            Address :string?,
            AddressLine2 :string?,
            City :string?,
            State :string?,
            Zip :string?,
            Country :string?,
            ShareWithCharities :bool?
      }
}
```

Request Notes:

- The *RetailerId* needs to be a valid CharityChoice Gift card retailer Id. You can access the list of Retailers by calling Gift Card Retailers List.
- The *PIN*, *CVV* and *ExpireDate* are only needed for Credit Card Retailer cards. For all other cards, set the values to null.
- Put whatever you believe to be the remaining balance into the *Balance* field. When the card is verified, an email will be sent out with the final balance that was found on the card at the time of verification.
- Up to three charities are allowed. The final *Balance* will be evenly divided among the charities.
- All the fields in the *ContactInfo* section must be provided in the request data. If there is a question-mark after the data type in the list above, you may set the value to null.

Response Status Codes:

- **400** – The request is not in the correct format. Some of the required fields are missing or empty. The *ErrorMessage* field in the response data will often contain an explanation of which information is missing or invalid. If the information is not helpful and this is a recurring response, please contact CharityChoice for assistance.
- **417** – This card has already been donated.
- **201** – The gift card donation was completed successfully. The response data will contain the order id.

Response Data:

```
{
      Succeeded :bool
      OrderId :int
      DonorId :int
      CardId :int
      ErrorMessage :string?
}
```

Response Notes:

- The Response includes a Location header with the URI of the Get Gift Card Status call for this donation. You should save this URI in order to check the cards current status. You can also build a Get Gift Card Status call using the returned *DonorId* and *CardId*.

Special Note:

Even after the API request has completed successfully, the balance on the Retail Gift Card has not yet been verified.
The user that was specified in the *ContactInfo* section of the request, will receive one email immediately, confirming the details that were supplied to us in the API request for this card donation.
Usually within 48 hours, the user will receive an additional email containing the tax receipt and card balance confirmation. The card balance will also become zero at this time.
It is important to inform the user that occasionally, tax acknowledgments are delayed by several business days, due to technical issues sometimes encountered during the verification processes of some of the Gift Card Retailers.
You can always check the current status of the card with a Get Gift Card Status call.

GET GIFT CARD STATUS

Get the current status and basic information of a previously donated Retailer Gift Card.

GET **/dyc/donorId:int/cardId:int**

Request Notes:

- The *donorId* is returned by Donate Retailer Gift Card call.
- The *cardId* is also returned by Donate Retailer Gift Card call.

Response Status Codes:

- **400** – The *donorId* and *cardId* combination supplied does not correctly refer to a previously donated Retailer Gift Card.
- **200** – The request was completed successfully. The response data will contain the card information.

Response Data:

```
{
      Succeeded: bool,
      DateDonated: datetime,
      RetailerId: int,
      CardNumber: string,
      PIN: string?,
      CVV: string?,
      ExpireDate: datetime?,
      Balance :float,
      IsVerified :boolean,
      CharityIds :[int],
      ErrorMessage: string?
}
```

ACTIVATE CODES

Acquire active CharityChoice redemption codes. Creates an order in your account.

POST **/codes/activate**

Request:

```
{
      Denomination :float,
      NumberOfCodes :int,
      HideDenomination: bool (optional. Default: false)
      CharityIds: [int] (optional)
}
```

Request Notes:

- The *Denomination* can be any number (including floating point numbers) between 0 and 5,000.
- If *HideDenomination* is set to true, the user will not be shown the denomination of the card during redemption – if the card is redeemed on the CharityChoice site.
- If the optional *CharityIds* is supplied, in addition to the newly activated list of codes which is returned in the response, for each CharityId supplied in the CharityId's field, a special redeem code will be returned for each code, that when redeemed, the code will be redeemed specifically for that charity.

Response Status Codes:

- **400** – The activation failed. This usually indicates that the request was not in the required format. The *ErrorMessage* should contain more information about the failure. If the information is not helpful and this is a recurring response, please contact CharityChoice for assistance.
- **201** – The code activation was completed successfully. The response data will contain the order id and the list of newly activated codes.

Response Data:

```
{
      Succeeded :bool,
      OrderId :int,
      Codes : [string],
      RedeemURLs:[{
            RedeemCode: string,
            RedeemURL: string,
            QRCodeImageURL: string
      }],
      DonationQCCode: string,
      DonationQCCodeRedeemUrl: {
            RedeemCode: string,
```

```
            RedeemURL: string,
            QRCodeImageURL: string
      },
      CharitySpecificCodes : [{
            CharityId: int,
            Codes: [string],
            RedeemURLs:[{
                  RedeemCode: string,
                  RedeemURL: string,
                  QRCodeImageURL: string
            }],
            DonationQCCode: string,
            DonationQCCodeRedeemUrl: {
                  RedeemCode: string,
                  RedeemURL: string,
                  QRCodeImageURL: string
            }
      }],
      ErrorMessage :string
}
```

Response Notes:

- The *RedeemURLs* field contains an array of objects - one for each Redemption Code in your *Codes* list. The array contains the properties:
  - ***RedeemCode –*** the redeem code
  - ***RedeemURL –*** a unique URL to the redemption page on the CharityChoice site with this redemption code automatically filled in for the user. If you have created a custom redemption page on CharityChoice, the user will be directed to your custom page with this code prefilled. You can also use the *Redeem* api endpoint to redeem your codes on your own site or application.
  - ***QRCodeImageURL –*** A URL to an image of a QR Code that points to the unique *RedeemURL* for this code. The URL will be formed something like this: `https://www.charitygiftcertificates.org/HTTPHandlers/UrlQRCode.ashx?size=3&URL=/Redeem/&RC=YOURCODE123456`. Note that you can change the value of the "size" query-string variable to set the size of the returned QR code image. Size=20 is about 400 pixels in width.
- The *DonationQCCode* field is a special redemption code for the entire order that can be used to redeem any code in your list. When using this code during redemption, the first un-redeemed code in your list will be redeemed. This will save you the need to give each recipient their own unique redeem code, allowing the printing and distributing of a single code, or URL, or QR Code, for all the codes in your order.
- The *DonationQCCodeRedeemUrl* field is a unique URL to the redemption page on the CharityChoice site with the *DonationQCCode* automatically filled in for the user. If you have created a custom redemption page on CharityChoice, the user will be directed to your custom page with the *DonationQCCode* prefilled. Note: you can also use the Redeem api endpoint to redeem the *DonationQCCode* on your own site or application.
- The *CharitySpecificCodes* field is only filled in if the request specified one or more CharityIds in the requests *CharityIds* field. In this case the *CharitySpecificCodes* field will contain a list of objects, one for each requested CharityId. Each one of these objects will contains a list of special redemption codes that when redeemed will redeem directly to that Charity. The *CharitySpecificCodes* also contains the following for each charity in your *CharityIds* list:
  - The *RedeemURLs* field contains an array of objects - one for each Redemption Code in your *Codes* list. The array contains the properties:

- ***RedeemCode* –** the redeem code that when redeemed will redeem directly to this Charity
- ***RedeemURL* –** a unique URL on the CharityChoice site to redeem this code for this charity. You can also use the *Redeem* api endpoint to redeem your codes on your own site or application.
- ***QRCodeImageURL* –** A URL to an image of a QR Code that points to the unique *RedeemURL* for this code. The URL will be formed something like this: `https://www.charitygiftcertificates.org/HTTPHandlers/UrlQRCode.ashx?size=3&URL=/Redeem/&RC=CODE123456C78`. Note that you can change the value of the "size" query-string variable to set the size of the returned QR code image. Size=20 is about 400 pixels in width.

- The *DonationQCCode* field is a special redemption code for the entire order that can be used to redeem any code in your list to this particular Charity. When using this code during redemption, the first un-redeemed code in your list will be redeemed. This will save you the need to give each recipient their own unique redeem code, allowing the printing and distributing a single code, or URL, or QR Code, for all the redemptions in your order.
- The *DonationQCCodeRedeemUrl* field is a unique URL on the CharityChoice site to redeem the first available redeem code to this particular charity. Note: you can also use the Redeem api endpoint to redeem the *DonationQCCode* on your own site or application.

Sample Request:

```
{
     "Denomination" = 5,
     "NumberOfCodes" = 2,
     "HideDenomination" = "false",
     "CharityIds" = [88, 89]

}
```

Sample Response:

```
{
    "OrderId": 165445,
    "Codes": [
        "ZH126011",
        "ZH126029"
    ],
    "RedeemURLs": [
        {
            "RedeemCode": "ZH126011",
            "RedeemURL": "https://redeem.ccgiftcards.org/ZH126011",
            "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
        },
        {
            "RedeemCode": "ZH126029",
            "RedeemURL": "https://redeem.ccgiftcards.org/ZH126029",
            "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
```

```
    }
],
"DonationQCCode": "QC2228690",
"DonationQCCodeRedeemUrl": {
    "RedeemCode": "QC2228690",
    "RedeemURL": "https://redeem.ccgiftcards.org/QC2228690",
    "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
},
"CharitySpecificCodes": [
    {
        "CharityId": 88,
        "Codes": [
            "ZH126011C88",
            "ZH126029C88"
        ],
        "RedeemURLs": [
            {
                "RedeemCode": "ZH126011C88",
                "RedeemURL": "https://redeem.ccgiftcards.org/ZH126011C88",
                "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
            },
            {
                "RedeemCode": "ZH126029C88",
                "RedeemURL": "https://redeem.ccgiftcards.org/ZH126029C88",
                "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
            }
        ],
        "DonationQCCode": "QC2228690C88",
        "DonationQCCodeRedeemUrl": {
            "RedeemCode": "QC2228690C88",
            "RedeemURL": "https://redeem.ccgiftcards.org/QC2228690C88",
            "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
        }
    },
    {
        "CharityId": 89,
        "Codes": [
            "ZH126011C89",
            "ZH126029C89"
        ],
        "RedeemURLs": [
            {
                "RedeemCode": "ZH126011C89",
                "RedeemURL": "https://redeem.ccgiftcards.org/ZH126011C89",
                "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
```

```
                },
                {
                    "RedeemCode": "ZH126029C89",
                    "RedeemURL": "https://redeem.ccgiftcards.org/ZH126029C89",
                    "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
                }
            ],
            "DonationQCCode": "QC2228690C89",
            "DonationQCCodeRedeemUrl": {
                "RedeemCode": "QC2228690C89",
                "RedeemURL": "https://redeem.ccgiftcards.org/QC2228690C89",
                "QRCodeImageURL": "https://www.charitygiftcertificates.org/..."
            }
        }
    ],
    "Succeeded": true,
    "ErrorMessage": null
}
```

ACTIVATE SUPPLIED CODES

Active the given list of codes for redemption of the given amount. Creates an order in your account.

NOTE: You can acquire and reserve a list of valid redemption codes with Generate and Reserve Redemption Codes.

POST **/codes/activateCodes**

Request:

```
{
      Denomination :float,
      Codes :[string]
}
```

Request Notes:

- The Denomination can be any number (including floating point numbers) between 0 and 5,000.
- Each code in the list needs to match the following Regular Expression: **^[a-zA-Z]{4,}[0-9]{4,}$**.
  This translates to:
  1. The code must start with 4 or more alphabetical characters.
  2. …followed by 4 or more numbers.

  NOTE: redemption codes are not case sensitive.

Response Status Codes:

- **400** – There were no codes in the request data.
- **417** – The activation failed. None of the codes were activated. The *ErrorMessage* will contain information for each failed removal.
- **201** – The code activation was completed successfully. All of the codes were successfully activated

Response Data:

```
{
      Succeeded :bool,
      OrderId :int,
      ErrorMessage :string
}
```

DE-ACTIVATE CODES

De-activate one or more redemption codes.

You can only de-activate codes that are in your account.

DELETE **/codes/remove/codes:string**

Request Notes:

- "codes" is a comma-separated list of codes to be de-activated. No spaces allowed. The regular expression used to test for a valid "codes" parameter is *^([a-zA-Z]*\d+,?)+$*

Response Status Codes:

- **400** – There were no codes in the request data.
- **417** – The deactivation failed. None of the codes were de-activated. The *ErrorMessage* will contain information for each failed removal.
- **200** – The code de-activation was completed successfully. All of the codes were successfully removed
- **202** – The code de-activation was only partially completed successfully. Some of the codes were not de-activated. The *ErrorMessage* will contain information for each failed removal.

Response Data:

```
{
      Succeeded :bool,
      ErrorMessage :string
}
```

Response Notes:

- If **all** if the codes were successfully removed, the returned status will be 200 and *Succeeded* will be true.
- If **none** of the codes were removed the returned status will be 417. The *ErrorMessage* will contain information for each failed removal.
- If only **some** of the codes were removed, *Succeeded* will be **false** and the returned status will be 202. The *ErrorMessage* will contain information for each failed removal.

SEND DIGITAL CARDS

Send one or more redeemable Digital Cards.

Cards can be delivered to the recipient either by email or sms.

The sent cards can be customized using the Greeting, Message, CardColor and CardId or CustomImageURL fields.

For cards that are sent by email, there is an option to set the date and time when the email should go out. This option is not currently available for cards sent by sms, and even if the SendDate is set, the cards will go out immediately.

POST **/digital/send**

Request:

```
{
      Denomination: float,
      Greeting: string,
      Message: string,
      CardId: int,
      CardColor: string,
      CustomImageURL: string,
      SendDate: Date,
      DonorName: string,
      DonorEmail: string,
      Recipients:[{
            FirstName: string,
            LastName: string,
            Email: string,
            Cel: string
      }]
}
```

Request Notes:

- "*Greeting*" is shown in emphasized text above the digital card image. (HTML is not allowed.)
- "*Message*" is shown in regular text below the digital card image. (HTML is not allowed.)
- "*CardId"* is the ID for one of the CharityChoice digital card images. You can use any one of our many card images or use your own image. You can get a list of CharityChoice cards with their images, by doing a call to GetDigitalCardsList. To use your own image, use the "*CustomImageURL*" field.
- "*CardColor*" (optional) is one of the Card Color Names returned from a call to GetDigitalCardColors. The default color is "Soft_White".
- "*CustomImageURL*" (optional) allows you to use your own image in the ecard. The URL should be a valid and accessible URL to an image. Please use a secure URL. (https). The recommended image size is 792 pixels by 460 pixels.
  Do not use large images as the ecard may then take a while to open.
- "*DonorName*" (optional) is if you wish to customize the name shown on the ecard as the ecard sender. The default is the name registered for the account whose credential were supplied for this api call.

- "*DonorEmail*" (optional) is if you wish to customize the email address shown as the ecard sender. The default is the email registered for the account whose credential were supplied for this api call.
- "*SendDate*" (optional) is when the email should go out. If this is not supplied or is set to null, the email will go out immediately.
  NOTE: This option is not currently available for cards sent by sms, and even if *SendDate* is set, the cards will go out immediately.
- *"Recipients"* is the list of digital cards recipients that are to receive codes. One of the name fields must be supplied. If the *Email* field is supplied, the user will receive a Digital Card via email. If the *Cel* field is supplied the user will receive the Digital Card via an SMS message.

Response Status Codes:

- `400 – The request is not valid. The response ErrorMessage will contain details about which information is incorrect. See below for further details.`
- `200 – The donation was completed successfully. All of the codes were successfully sent to the recipients. The OrderId will contain the Order Id for this transaction.`

Response Data:

```
{
      Succeeded :bool,
      ErrorMessage :string,
      OrderId: int,
      Cards:[{
            FirstName: string,
            LastName: string,
            Email: string,
            Cel: string,
            RedemptionCode: string,
            RedeemURL: string,
            IsValid: Boolean,
            Notes: string
      }]

}
```

Response Notes:

- If all if the Digital Cards were successfully sent, the returned status will be 200 and *Succeeded* will be true. The *Cards* field will contain the Redemption Code and unique Redeem URL for each sent card.
- If there was an issue with the request data, the returned status will be 400.
  In that case, the ErrorMessage will contain general information explaining which request field is missing or invalid.
  If the problem was with one of the recipients in the request Recipients list, then for that recipient's entry in the response's Cards list, the IsValid field will be false and the "Notes" field will contain a detailed explanation.

ORDER PHYSICAL CARDS

Order any number of Physical Charity Gift Cards and have them shipped to you or directly to the final recipient.

The cards denomination appears on the actual gift card. The CharityChoice Ribbon and Season cards have the amount on the front bottom left corner; while the Flower and Birthday cards have the amount on the back of the card.

Orders are shipped on the same business day for orders made before noon Eastern Time; otherwise on next business day.

Only one shipping charge for your entire order, when sent to the same address. You are not charged shipping for each card.

For cards that are sent directly to the final recipient, a customizable packing slip is included. The packing slip is styled to match the style of card being shipped. You can see what the slip looks like at: https://www.charitygiftcertificates.org/images/stationary.jpg

POST **/physical**

Request:

```
{
    "Cards": [{
        "Denomination": int,
        "PhysicalCardTypeId": int,
        "NumberOfCards": int
    }],
    "ShippingInfo": {
        "ShippingTypeId": int,
        "FirstName": string,
        "LastName": string,
        "Attention": string?,
        "AddressLine1": string,
        "AddressLine2": string?,
        "City": string,
        "State": string,
        "Zip": string,
        "Email": string?,
        "Phone": string?,
        "Greeting": string?,
        "Message": string?
    },
    "ShipsToRecipient": bool?
}
```

Request Notes:

- "*Cards*" this is an array of cards containing the list of how many from each card type and denomination being ordered. You can have a multiple card types and/or denominations shipped to the same address. Note, that even in this case, you are only charged for a single shipping.
- "*Cards/Denomination*" the redeemable charity value of these cards. The cards denomination appears on the actual gift card. The denominations that are available for each card type and the number of cards that are currently available in stock, can be obtained by a call to Get Physical Card Types.
- "*Cards/PhysicalCardTypeId*" the type of card being ordered. A list of all the available card types, information about them (including a link to preview what the card looks like), their PhysicalCardTypeId's, which denominations are available for each card type, and the number of cards that are currently available in stock, can be obtained by a call to Get Physical Card Types.
- "*Cards/NumberOfCards*" the number of cards of this type to order.
- *"ShippingInfo"* all the cards in this order will be shipped to the shipping information supplied in this field.
- *"ShippingInfo/ShippingTypeId"* this field sets the shipping carrier and type. To acquire a list of shipping types with their ShippingTypeId's, price and full details, do a call to Get Shipping Types.
  Important note: some of the shipping types are only available between certain dates. For example, during holiday season, there are special discounted rates and for a few days even free shipping. It is therefore important to do the call to Get Shipping Types to determine which shipping types are available today.
- *"ShippingInfo/State"* the 2-character state abbreviation.
- "*ShippingInfo/Greeting*" (optional) for cards where *ShipsToRecipient* is set to true, the *Greeting* is printed on the included packing slip.
- "*ShippingInfo/Message*" (optional) for cards where *ShipsToRecipient* is set to true, the *Message* is printed on the included packing slip.
- "*ShipsToRecipient*" (optional, default is false). Set this to true only if this order is being shipped to the final recipient. When set to true, a packing slip will be shipped together with the cards. In this case, the *ShippingInfo/Greeting* and *ShippingInfo/Message* should be supplied to customize the text on the packing slip.

Response Status Codes:

- `400 – The request is not valid. The response ErrorMessage will contain details about which information is incorrect.`
- `201 – The order was completed successfully. The OrderId will contain the Order Id for this transaction.`

Response Data:

```
{
      Succeeded :bool,
      ErrorMessage :string,
      OrderId: int,
      TotalForCards: float,
      TotalForShipping: float,
      TotalCost: float,
      Cards: [{
              RedemptionCode: string
              Denomination: int,
              PhysicalCardTypeId: int,
              RedeemURL: string,
```

```
        }],
        ShippingInfo: {
            ShippingTypeId: int,
            FirstName: string,
            LastName: string,
            Attention: string,
            AddressLine1: string,
            AddressLine2: string,
            City: string,
            State: string,
            Zip: string,
            Email: string,
            Phone: string,
            Greeting: string,
            Message: string
        },
        ShipsToRecipient: bool
}
```

Response Notes:

- If the order was completed successfully, the returned status will be 201 and *Succeeded* will be true.
- If there was an issue with the request data, the returned status will be 400. In that case, the *ErrorMessage* will contain general information explaining which request field is missing or invalid.
- The *TotalForCards* field contains the total charge for just the cards themselves without the shipping cost. Note: the cards only cost their denomination / face value; which is the same amount that will be available to the recipient for redemption to charity.
- The *TotalForShipping* field contains the total cost for shipping for the entire order. The cards are shipped together, and there will only be a single shipping charge for the entire order.
- The *TotalCost* field simply contains the sum of the *TotalForCards* and the *TotalForShipping*.
- The *Cards* field contains the Redemption Code and unique Redeem URL for each sent card.

GET PHYSICAL CARD TYPES

Gets a list of Physical Card Types with which denominations are available and how much is available in stock.

GET **/physical/cardTypes**

Response Data:

```
{
      Succeeded :bool,
      ErrorMessage :string,
      List: [{
            PhysicalCardTypeId : int
            CardTypeName : string,
            Description : string,
            ImageUrl : string,
            Denominations: [{
                Denomination: int,
                Stock: int
            }]
      }]
}
```

GET SHIPPING TYPES

Gets a list of Shipping Types for physical cards orders.

Includes pricing information.

GET **/physical/shippingTypes**

Response Data:

```
{
      Succeeded :bool,
      ErrorMessage :string,
      List: [{
            ShippingTypeId: int
            ShippingCompanyName: string,
            FullDisplayText: string,
            IsExpeditedShipping: bool,
            BaseShippingCost: float,
            ExpeditedShippingCost: float
      }]
}
```

Response Notes:

- **Important note**: some of the shipping types are only available between certain dates. For example, during holiday season, there are special discounted rates and for a few days even free shipping. It is therefore important to do the call to Get Shipping Types to determine which shipping types are available today.
- *List/ShippingTypeId* This field is the unique identifying Id for each shipping type. It should be used as the value for the *"ShippingInfo/ShippingTypeId" field in* the call to Order Physical Cards to set the shipping carrier and type.

GENERATE AND RESERVE REDEMPTION CODES

Acquire a list of valid non-active CharityChoice redemption codes.

These codes can be activated afterwards by doing an Activate Supplied Codes request.

The generated codes are guaranteed to stay "reserved" for 30 days from the time of generation.

This means, that when activating these codes within 30 days you can safely assume that they will not have been already activated.

After 30 days, you don't have this guarantee and may receive a "Code already active" error when attempting to activate them.

GET **/codes/generateCodes/prefix:string/numberOfCodes:int**

Request Notes:

- The prefix needs to be a string of at least 4 alphabetical characters.
  The regular expression **^[a-zA-Z]{4,}$** is used to check the *prefix* section of the URL.
  This causes that if the prefix format does not match the pattern, the API function will not be accessed and a 404 resource-not-found error will be returned.
- If you wish to generate redeem codes that can only be redeemed to a particular charity, the CharityChoice Charity Id for that charity, should be supplied in request in the following format: ***/codes/generateCodes/prefix:string/numberOfCodes:int/charityId:int***.

NOTE: redemption codes are not case sensitive.

Response Data:

```
{
      Succeeded :bool,
      Codes :[string],
      ErrorMessage :string
}
```

Response Notes:

- The codes returned in the *Codes* list are "reserved" for 30 days.
  This means, that when activating these codes within 30 days you can safely assume that they will not have been already activated.
  After 30 days, you don't have this guarantee and may receive a "Code already active" error when attempting to activate them.

GET CODE STATUS

Get the status of the given redemption code.
Only allowed for codes that are in your account unless you have special permissions.

GET **/codes/status/redemptionCode:regex(^[a-zA-Z]*\d+$)**

Request Notes:

- The *redemptionCode* needs to be a valid active CharityChoice Code. The regular expression in the url causes that if the code does not match the pattern, the API function will not be accessed. This will cause the API call to return a 404 resource-not-found error.

Response Status Codes:

- **400** – The code status cannot be retrieved. Either the given redemption code is not an active CharityChoice code or it is not in your account. The *ErrorMessage* will contain more information.
- **200** – The code status request was completed successfully.

Response Data:

```
{
      Succeeded :bool,
      Codes: [{
            Denomination :float
            WasRedeemed :bool
            AmountLeftToRedeem :float
            RedeemDate: date
            CharityIds: [integer]
      }],
      ErrorMessage :string
}
```

Response Notes:

- Even though this call only takes a single code, the Response returns an array. This was done in order to have both versions of Get Code Status have the same response schema.

GET CODES STATUS

Gets the status multiple redemption codes.
Only allowed for codes that are in your account unless you have special permissions.

POST **/codes/status**

Request

```
{
      Codes :[string]
}
```

Response Status Codes:

- **400** – Either the request was not formatted correctly or the code status cannot be retrieved for one or more codes in the request. The given redemption code may not be an active CharityChoice code or it is not in your account. The *ErrorMessage* will contain information for each failed status request.
- **417** – The *Codes* field in the request is either missing or empty.
- **200** – The code status was successfully retrieved for all the code listed in the request.

Response Data:

```
{
      Succeeded :bool
      Codes: [{
            Code :string
            Denomination :float
            WasRedeemed :bool
            AmountLeftToRedeem :float
            RedeemDate: date
            CharityIds: [integer]
      }]
      ErrorMessage :string
}
```

Response Notes:

- If any of the codes failed the status request, the returned status will be 400 and Succeeded will be false.
- The ErrorMessage only contains the fail information of the first failed code.

VOID ORDER

Void a previously made order.
Only allowed for orders that are in your account.

USE WITH CAUTION!
Removes all order information. If any of the codes were redeemed, the redemption information is removed as well.

DELETE **/donate/orderId:int**

Request Notes:

- The *orderId* is returned from any calls to Activate Codes, Charity Donation and Donate Retailer Gift Card. You can also find the *orderId* on your account page on the CharityChoice site.

Response Status Codes:

- **400** – The request was not formatted correctly.
- **403** – You are not allowed to void this order. The *ErrorMessage* will contain the reason for the failure.
- **417** – Order cannot be voided. Either the order does not exist or you do not have the permissions to void this order. The *ErrorMessage* may contain more information about this failure.
- **200** – The request completed successfully. The order has been voided and removed from your account.

Response Data:

```
{
      Succeeded :bool
      ErrorMessage :string
}
```

REDEEM A CODE

Redeem a redemption code for one or more charities.

To redeem a code that is not in your account you will need special permissions. Please contact CharityChoice for details.

POST **/codes/redeem**

Request:

```
{
      RedeemCode: string,
      Charities: [{
            CharityId :int,
            CharityProject :string,
            Amount :float
      }],
      RedeemerFirstName :string,
      RedeemerLastName :string,
      RedeemerEmail :string
}
```

Request Notes:

- All fields (except for *CharityProject*) must have a valid value.
- Up to three charities are allowed.
- The *CharityProject* property is to redeem to a particular project within the charity. If you do not want to supply this information, set *CharityProject* to null.
- The *Amount* can be any number (including floating point numbers) between 0 and 5,000.
- If *Amount* is set to zero for all the charities in the list, the funds will be evenly divided among the charities.

Response Status Codes:

- **400** – The request is not correct. Some of the required fields may be missing or empty. If the information in the *ErrorMessage* is not helpful and this is a recurring response, please contact us.
- **403** – You do not have permission to redeem this code.
- **410** – The Code has already been redeemed or does not have a sufficient balance to complete this redemption. The *ErrorMessage* will contain more information.
- **201** – The Code redemption was completed successfully.

Response Data:

```
{
      Succeeded :bool
      ErrorMessage :string
}
```

REMOVE REDEMPTION

Remove a previous redemption from a CharityChoice code.
You can only remove redemption information from a code that is in your account.

DELETE **/codes/redeem/redemptionCode:regex(^[a-zA-Z]*\d+$)**

Request Notes:

- The *redemtionCode* needs to be an active CharityChoice Code. If the redemption code in the URL does not match the regular expression pattern, the API function will not be accessed.

Response Status Codes:

- `400 – The supplied Code is not a valid CharityChoice Code.`
- `403 – The Code is not in your account.`
- `417 – The Code was never redeemed.`
- `200 – The redemption information was successfully removed.`

Response Data:

```
{
      Succeeded :bool
      ErrorMessage :string
}
```

GET ORDER TYPES

Gets a list of the CharityChoice Order Types.

GET **/donate/ordertypes**

Response Data: (Array)

```
[{
      OrderTypeName :string,
      OrderTypeId :int
}]
```

GET DIGITAL CARDS LIST

Gets a list of the CharityChoice Digital Cards with their images.

GET **/digital/ digitalCardsList**

Response Data: (Array)

```
[{
      CardId: number,
      CardName: string,
      ImageUrl: string,
      OccasionId: number,
      OccasionName: string,
      HasCustomSection: Boolean,
      CustomSectionHtml: string
}]
```

Response Notes:

- *CardId* is the CharityChoice id of the card. This will be used as the *CardId* value in calls to Send Digital Cards
- *CardName* is the display description that can be used to describe this card.
- *ImageUrl* is the URL for the card image.
- *OccasionId* is the CharityChoice id of the occasion this card is part of. (Mostly for internal use)
- *OccasionName* is the occasion name for the category that this card is part of.
- *HasCustomSection* indicates if this card has a section that can be customized with HTML.
- *CustomSectionHtml* is for cards that have a customizable section, this is the default HTML of this section.

GET DIGITAL CARD COLORS

Gets a list of the CharityChoice Digital Cards Color Schemes.

GET **/digital/digitalCardColors**

Response Data: (Array)

```
[{
      SchemeName: string,
      SchemeText: string,
      BackgroundColor: string
}]
```

Response Notes:

- *SchemeName* is the id of the color scheme. This will be used as the *CardColor* value in calls to Send Digital Cards
- *SchemeText* is the display text that can be used to show to describe this color scheme.
- *BackgroundColor* is the background color of the digital card when using this scheme. The value is in hexadecimal RBG format without the leading hash mark. Such as, C0180D or 73726D.