PicThrive Api Documentation

Release 1.0.1

Adventure Soup Inc

Aug 10, 2017

Start Guide

Want to integrate PicThrive to bolster your service or build your own photo Apps?

Request access to PicThrive's API by contacting us at info@picthrive.com.

CHAPTER 1

Connecting to an Account

PicThrive uses OAuth2 Web Server flow to allow 3rd party applications to connect to user accounts.

EndPoints

- Authorize URI: https://api.picthrive.com/v1/auth/oauth2/authorize
- Access Token URI: https://api.picthrive.com/v1/auth/oauth2/token

Web Server Flow

Your Data

You will need to already have the following:

- client_id
  - Your Client Id setup through PicThrive's dev portal.
- client_secret
  - Your Client Secret as setup through PicThrive's dev portal.
- redirect_uri
  - Your Redirect URI as setup through PicThrive's dev portal.
  - Eg. https://mysite.com/oauth/complete

Step 1: Client Login and Authorization

Make a GET request in a new browser to the Authorize URI filling in your client data.:

```
https://api.picthrive.com/v1/auth/oauth2/authorize?response_type=code&client_id=<your␣
↪client id>&redirect_uri=<your redirect uri>
```

This will present the user with a chance to login and authorize your application. After authorizing we will perform a redirect.

Step 2: Redirect

The browser window will be redirected to your redirect_uri:

```
<your redirect uri>?code=<generated auth code>
```

Your server will receive this code and then perform the next step in the OAuth flow

Step 3: Server Token Exchange

Your server will make an out-of-band request to https://api.picthrive.com/v1/auth/oauth2/token.:

```
https://api.picthrive.com/v1/auth/oauth2/token?grant_type=authorization_code&client_
↪id=<your client id>&client_secret=<your client secret>&code=<generated auth code>&
↪redirect_uri=<your redirect uri>
```

We will return you a 'Bearer' Access token:

```
{
    "access_token": "adsfsdfsadfs",
    "refresh_token": "asdfsadfsadf",
    "scope": "",
    "token": "adsfsdfsadfs",
    "token_type": "Bearer"
}
```

CHAPTER 2

Making Requests

Auth

When making requests to PicThrive's servers you will need to set the **Authorization** header to the **access_token** received in the OAuth flow:

```
Header:
    Authorization: Bearer <access_token>
```

See *Connecting to an Account* for info about obtaining an access_token

Endpoint

PicThrive's Api can be accessed from:

```
https://api.picthrive.com/
```

Displaying Images

All image URLs returned by the API do not include the CDN hostname or protocol. The following prefix can be added to obtain access to all our hosted logos.:

```
https://d1rj07wouwybr9.cloudfront.net/
```

Photos are hosted based upon the region they were uploaded to.:

```
picthrive-dyn : https://d1rj07wouwybr9.cloudfront.net
picthrive-dyn-ireland : https://de9rdw742q71n.cloudfront.net
picthrive-dyn-sydney : https://d27ikeqtdcefva.cloudfront.net
picthrive-dyn-virginia : https://d1p40og7d7er5y.cloudfront.net
```

Rate Limiting

Dev account access may be rate limited at our discretion.

CHAPTER 3

Making a Photo Sale

One common integration request is to make photo sales before or after a booking. This short little guide aims to jump start that process so that you can get started faster.

Authentication

We use OAuth2 to share access to a client's account. See *Connecting to an Account* to get started.

See *Making Requests* to determine how to Authenticate your requests.

Making a Pre/Post Sale

To create a photo sale you will need to create a Token. Make a POST call to */v1/Tokens/* to create a Token. This will return a unique code, eg: **DBqLVv**. This code can be used by the customer to redeem their photos.

At this point you can either email the token through our system to the client, or you can create the store link yourself.

Send Email Through Us

Taking the Token from above, make a POST call to */v1/Email/*. Set 'Type' to 'token' and fill in the rest of the fields. Be sure to check the response body for the Status code of each email.

Send Email Yourself

If you wish to send the email yourself, you will need to construct the Store URL.

1. Determine the **VanityUrl** for the given Brand (See */v1/Brands/*). Eg. 'bobs_rafting'
2. Get the Token from the previous step. Eg. 'DBqLVv'

3. Construct the basic URL:

```
https://store.picthrive.com/bobs_rafting/?token=DBqLVv
```

This link will take them to the store front page, and the customer will be prompted to notify them that they have got X photos to redeem.

Going Further

Tokens attached to a Group

Tokens can be created with knowledge of which Group they are apart of. This allows us to directly link the customer to the correct gallery, instead of to the generic storefront. To do so, you will need to retrieve the GroupId and GroupToken and use these when making the POST /v1/Token/ call.

To retrieve info on what Groups occured on which date see */v1/Group/*.

Now that the Token has been attached to a Group, you can either make the POST /v1/Email/ call or construct a simpler URL. If you make the POST /v1/Email/ call you will need to set the Type to 'tokenAndGroup' and include the required Group information.

If you wish to send the email yourself, you can can use either our short url link, or construct the full link. The short url looks like:

```
http://ptur.co/DBqLVv
```

If the GroupToken is 'EAT4P' then the long url will look like:

```
https://store.picthrive.com/bobs_rafting/group/EAT4P=?token=DBqLVv
```

Changing to Store's default Date

If you want to link the customer to the date of their trip, instead of the general storefront page, you can create the store URL like this:

```
https://store.picthrive.com/bobs_rafting/date/2016-03-04?token=DBqLVv
```

CHAPTER 4

Brands

The following endpoint describes manipulations of a Brand. A Brand has its own store, links, logos and campaigns.

/v1/Brand/

GET

Returns the Brands that this user Owns.

Returns

Fields

- **Brands**: A list of brands. See *Get Brand Return*
- **Error**: Optional error string.

Return Example (200 OK):

```
{
    "Brands": [
        {
            "OwnerId": "<owner uuid>",
            "BrandId": "<brand uuid>",
            "VanityUrl": "<brand vanity url>",
            "Name": "name of brand",
            "Website": "http://...",
            "BookingLink": "http://...",
            "Description": "some 140 character description",
            "Social": {
                "FacebookUrl": "https://...",
                "TwitterUrl": "https://...",
                "GooglePlusUrl": "https://..."
```

```
            },
            "Review": {
                "TripAdvisorUrl": "https://"
            },
            "Banner300": "/image/suffix.jpg",
            "Banner1280": "/image/suffix.jpg",
            "Logo100": "/image/suffix.jpg",
            "Logo150": "/image/suffix.jpg",
            "Logo300": "/image/suffix.jpg",
            "GoogleAnalytics": "UA-000000-01",
            "Pricing": {
                "Mode": "flat",
                "Flat": 500,
                "Currency": "USD",
                "Levels": [],
                "FullAlbum": 0,
            },
            "MailChimp": {
                "Account": "string",
                "List": "string",
                "AccountName": "string",
                "ListName": "string"
            },
            "Stripe": {
                "Account": "string",
                "AccountName": "string",
                "PublishableKey": "string"
            },
            "Campaign": {
                "Headline": "string",
                "CreatedAt": 1449791030,
                "Type": "string"
            },
            "Ads": [],
            "AdsUpdated": 1449791030,
            "GoogleRemarketingCode": "string",
            "FacebookRemarketingCode": "string",
            "RemarketingUpdated": 1449791030
        },
        ...
    ]
}
```


POST

Creates a brand.

PostData

Only 'Name' and 'VanityUrl' are required. All other Brand fields are optional. A VanityUrl must be unique and is case insensitive.

Fields

- **VanityUrl**: Required. The url suffix for the brand. Eg. 'picthrivedemo' will result in the storefront url of https://store.picthrive.com/picthrivedemo

- **Name**: Required. The name of the Brand.

Example PostData:

```
{
    "VanityUrl": "string",
    "Name": "string name"
}
```

Returns

Fields

- See *Get Brand Return*

Example Return (201 Created):

```
{
    "OwnerId": "<owner uuid>",
    "BrandId": "<brand uuid>",
    "VanityUrl": "<brand vanity url>",
    "Name": "name of brand"
}
```

/v1/Brand/<brand uuid>

GET

Returns info about the specified Brand.

Returns

Fields

- **OwnerId**: Owner UUID
- **BrandId**: Brand UUID
- **VanityUrl**: Vanity Url Suffix for this brand. Eg. 'picthrivedemo'. Which means the storefront url is 'https://store.picthrive.com/picthrivedemo'
- **Name**: Name of the Brand.
- **Website**: Optional. Website of the brand.
- **BookingLink**: Optional. Booking link of the brand.
- **Description**: Optional. Description of the brand.
- **Social**: Optional. Set of social links.
  - **FacebookUrl**: Optional. Url to their Facebook page.
  - **TwitterUrl**: Optional. Url to their Twitter page.
  - **GooglePlusUrl**: Optional. Url to their Google+ page.
- **Review**: Optional. Set of review based links to TripAdvisor and Google+.

  - **TripAdvisorUrl**: Optional. Link to their TripAdvisor page.
- **Banner300**: Optional. Their storefront banner url suffix, sized at a width of 300px.
- **Banner1280**: Optional. Their storefront banner url suffix, sized at a width of 1280px.
- **Logo100**: Optional. Their logo url suffix, sized at a width of 100px.
- **Logo150**: Optional. Their logo url suffix, sized at a width of 150px.
- **Logo300**: Optional. Their logo url suffix, sized at a width of 300px.
- **GoogleAnalytics**: Optional. Their Google Analytics code to inject onto their store and album pages.
- **Pricing**: Optional. Pricing Information for their store.
  - **Mode**: Three possible values { off, flat, grad }
  - **Flat**: Only present if **Mode** is **flat**. The flat cost in cents for each photo.
  - **Currency**: The currency to charge in { “CAD”, “USD” }
  - **Levels**: Only present if **Mode** is **grad**. The set of graduated levels of pricing.
    - **Level**: The number of photos. <= this value.
    - **Amount**: The cost in cents for this level.
  - **FullAlbum**: Only present if **Mode** is **grad**. The total price to pay if all levels are exceeded.
- **MailChimp**: Optional. Describes their selected account. Does not manage the MailChimp credentials.
  - **Account**: The selected Account Id.
  - **List**: The selected List Id.
  - **AccountName**: The Name of the selected account.
  - **ListName**: The Name of the selected list.
- **Stripe**: Optional. Describes their selected account. Does not manage the Stripe credentials.
  - **Account**: The selected Account Id.
  - **AccountName**: The selected Account Name.
- **Campaign**: Optional. Info about which campaign is currently running.
  - **Headline**: The title of the prompt.
  - **CreatedAt**: When the campaign was started.
  - **Type**: Type of campaign. {“Subscribe”, “Review”, “Share”, “Website”, “Book”, “Like”}
- **Ads**: Optional. List of banner ads to run.
  - **Chance**: The chance of running this add. [0.0 - 1.0]
  - **ImageUrl**: The url suffix of the banner image.
  - **Link**: The link to navigate to when clicking the ad.
- **AdsUpdated**: Optional. Unix time of the last udpate to ads.
- **GoogleRemarketingCode**: Optional. Google Remarketing code.
- **FacebookRemarketingCode**: Optional. Facebook Retargeting code.
- **RemarketingUpdated**: Optional. Unix time of last remarketing update.

Example Returns (200 OK):

```
{
    "OwnerId": "<owner uuid>",
    "BrandId": "<brand uuid>",
    "VanityUrl": "<brand vanity url>",
    "Name": "name of brand",
    "Website": "http://...",
    "BookingLink": "http://...",
    "Description": "some 140 character description",
    "Social": {
        "FacebookUrl": "https://...",
        "TwitterUrl": "https://...",
        "GooglePlusUrl": "https://..."
    },
    "Review": {
        "TripAdvisorUrl": "https://"
    },
    "Banner300": "/image/suffix.jpg",
    "Banner1280": "/image/suffix.jpg",
    "Logo100": "/image/suffix.jpg",
    "Logo150": "/image/suffix.jpg",
    "Logo300": "/image/suffix.jpg",
    "GoogleAnalytics": "UA-000000-01",
    "Pricing": {
        "Mode": "flat",
        "Flat": 500,
        "Currency": "USD",
        "Levels": [
            {
                "Level": 1,
                "Amount": 500
            },
            ...
        ],
        "FullAlbum": 0,
    },
    "MailChimp": {
        "Account": "string",
        "List": "string",
        "AccountName": "string",
        "ListName": "string"
    },
    "Stripe": {
        "Account": "string",
        "AccountName": "string"
    },
    "Campaign": {
        "Headline": "string",
        "CreatedAt": 1449791030,
        "Type": "string"
    },
    "Ads": [
        {
            "Chance": 1.00,
            "ImageUrl": "/image/suffix.jpg",
            "Link": "https://..."
        },
        ...
    ],
```

```
    "AdsUpdated": 1449791030,
    "GoogleRemarketingCode": "string",
    "FacebookRemarketingCode": "string",
    "RemarketingUpdated": 1449791030
}
```

PUT

Updates a Brand. Missing fields are ignored. Empty fields, for things like description, are interpreted as 'delete'.

PostData

Fields

- **Name**: Name of the Brand.
- **Website**: Optional. Website of the brand. Set to "" to delete.
- **BookingLink**: Optional. Booking link of the brand. Set to "" to delete.
- **Description**: Optional. Description of the brand. Set to "" to delete.
- **Social**: Optional. Set of social links.
  - **FacebookUrl**: Optional. Url to their Facebook page. Set to "" to delete.
  - **TwitterUrl**: Optional. Url to their Twitter page. Set to "" to delete.
  - **GooglePlusUrl**: Optional. Url to their Google+ page. Set to "" to delete.
- **Review**: Optional. Set of review based links to TripAdvisor and Google+.
  - **TripAdvisorUrl**: Optional. Link to their TripAdvisor page. Set to "" to delete.
- **Banner**: Optional. Base64 encoded image to use as the banner.
- **Logo**: Optional. Base64 encoded image to use as the logo.
- **GoogleAnalytics**: Optional. Their Google Analytics code to inject onto their store and album pages. Set to "" to delete.
- **Pricing**: Optional. Pricing Information for their store.
  - **Mode**: Three possible values { off, flat, grad }. Set to 'off' to remove pricing.
  - **Flat**: Only present if **Mode** is **flat**. The flat cost in cents for each photo.
  - **Currency**: The currency to charge in. { "CAD", "USD" }
  - **Levels**: Only present if **Mode** is **grad**. The set of graduated levels of pricing. Any posted levels will overwrite other levels.
    - **Level**: The number of photos. <= this value.
    - **Amount**: The cost in cents for this level.
  - **FullAlbum**: Only present if **Mode** is **grad**. The total price to pay if all levels are exceeded.
- **MailChimp**: Optional. Describes their selected account. Does not manage the MailChimp credentials.
  - **Account**: The selected Account Id.
  - **List**: The selected List Id.

  - **AccountName**: The Name of the selected account.
  - **ListName**: The Name of the selected list.
- **Stripe**: Optional. Describes their selected account. Does not manage the Stripe credentials.
  - **Account**: The selected Account Id.
  - **AccountName**: The selected Account Name.
- **Campaign**: Optional. Info about which campaign is currently running. Set 'Type' to 'Delete' to remove campaigns.
  - **Headline**: The title of the prompt.
  - **Type**: Type of campaign. {"Subscribe", "Review", "Share", "Website", "Book", "Like"}
- **Ads**: Optional. List of banner ads to run.
  - **Chance**: The chance of this ad running. [0.0 - 1.0]
  - **ImageBase64**: Base64 encoded image to use as the ad.
  - **Link**: The link to navigate to when clicking the ad.
- **GoogleRemarketingCode**: Optional. Google Remarketing code.
- **FacebookRemarketingCode**: Optional. Facebook Retargeting code.

Example Returns (200 OK):

```
{
    "OwnerId": "<owner uuid>",
    "BrandId": "<brand uuid>",
    "VanityUrl": "<brand vanity url>",
    "Name": "name of brand",
    "Website": "http://...",
    "BookingLink": "http://...",
    "Description": "some 140 character description",
    "Social": {
        "FacebookUrl": "https://...",
        "TwitterUrl": "https://...",
        "GooglePlusUrl": "https://..."
    },
    "Review": {
        "TripAdvisorUrl": "https://"
    },
    "Banner300": "/image/suffix.jpg",
    "Banner1280": "/image/suffix.jpg",
    "Logo100": "/image/suffix.jpg",
    "Logo150": "/image/suffix.jpg",
    "Logo300": "/image/suffix.jpg",
    "GoogleAnalytics": "UA-000000-01",
    "Pricing": {
        "Mode": "flat",
        "Flat": 500,
        "Currency": "USD",
        "Levels": [
            {
                "Level": 1,
                "Amount": 500
            },
            ...
        ],
```

```
        "FullAlbum": 0,
    },
    "MailChimp": {
        "Account": "string",
        "List": "string",
        "AccountName": "string",
        "ListName": "string"
    },
    "Stripe": {
        "Account": "string",
        "AccountName": "string"
    },
    "Campaign": {
        "Headline": "string",
        "CreatedAt": 1449791030,
        "Type": "string"
    },
    "Ads": [
        {
            "Chance": 1.00,
            "ImageUrl": "/image/suffix.jpg",
            "Link": "https://..."
        },
        ...
    ],
    "AdsUpdated": 1449791030,
    "GoogleRemarketingCode": "string",
    "FacebookRemarketingCode": "string",
    "RemarketingUpdated": 1449791030
}
```


/v1/Brand/<brand uuid>/Subscribe

POST

Adds an email to the Brand's subscription list.

PostData

Fields

- **Email**: The email to subscribe.

PostData:


```
{
    "Email": "example@example.com"
}
```


Returns

Example Returns (201 Created):

```
{
    "Error": "optional error string"
}
```

/v1/Brand/<brand uuid>/Group/

GET

Returns all the groups for the given day for the specified brand.

See *Get Groups* for query params and return data.

CHAPTER 5

Emails

The following endpoint allows for emails to be sent to customers, and for the status to be checked.

/v1/Email/

POST

Sends an email.

PostData

Fields

- **Messages**: A list of emails to send.
  - **To**: Email address to send to.
  - **Type**: Type of email being sent. {“tokenAndGroup”, “group”, “token”}
  - **BrandId**: The Brand this email is sent from.
  - **Token**: Used in “tokenAndGroup” or “token”. This specifies a purchase token.
  - **GroupId**: Used in “group” or “tokenAndGroup”. Specifies the GroupId you want to direct the customer to.
  - **NumPhotos**: Used in “tokenAndGroup” or “token”. Specifies the # of photos a customer receives. -1 => unlimited.

Example PostData:

```
{
    "Messages": [
        {
            "To": "example@example.com",
```

```
            "Type": "token",
            "BrandId": "<brand uuid>",
            "Token": "abcde",
            "NumPhotos": 10
        },
        ...
    ]
}
```

Returns

Fields

- **Results**: A list of the send results.
  - **Error**: Optional error string on why the email can't be sent.
  - **MessageId**: The id of the email sent.
  - **Status**: http status code (Eg. 200) on email send result.
- **Error**: Optional error string

Example Returns (200 OK):

```
{
    "Results": [
        {
            "Status": 200,
            "MessageId": "<message uuid>"
        }
    ]
}
```

GET

Returns the email history.

Query Params

Fields

- **BrandId**: Optional. Filter based on the specified BrandId.
- **Next**: Optional. Return the next set of results, take from the result of a previous call.
- **After**: Optional. Unix time that emails cannot be updated after. Default: Now.
- **State**: Optional. Which state the email should be in {"sent", "delivered", "bounced", "complaint"}
- **Email**: Optional. Which email to search for. Other Query Params will be ignored. Must be a direct match.

Returns

Fields

- **Emails**: List of email status and errors.

  - **Email**: The email address that this was sent to.
  - **Type**: The type of email that was sent: {"tokenAndGroup", "group", "token", "purchase"}
  - **MessageId**: The UUID of this email message.
  - **SentAt**: The unix time this email was sent at.
  - **LastUpdated**: The unix time this email was last updated at. Eg. Delivered notification.
  - **State**: The state of this email: {"sent", "delivered", "bounced", "complaint"}
  - **BrandId**: The Brand UUID this email was sent from.
  - **OwnerId**: The Owner UUID this email was sent from.
  - **MsgDetails**: Details about how the message was sent, and its content.
    * **To**: Email to send to.
    * **Type**: Type of email being sent. {"tokenAndGroup", "group", "token", "purchase"}
    * **BrandId**: The Brand this email is being sent from.
    * **Token**: Used in "tokenAndGroup" or "token". This specifies a purchase token.
    * **GroupId**: Used in "group" or "tokenAndGroup". Specifies the GroupId we want to direct the customer to.
    * **NumPhotos**: Used in "tokenAndGroup" or "token". Specifies the # of photos a customer received. -1 => unlimited.
    * **OrderId**: Used in "purchase". UUID of the customer's order.
  - **ComplaintType**: Type of complaint.
  - **BounceType**: Type of bounce.
  - **BounceSubType**: Subtype of bounce.
  - **BounceDetails**: Rawer Details about the bounce.
    * **bounceSubType**: See http://docs.aws.amazon.com/ses/latest/DeveloperGuide/notification-contents.html#bounce-types
    * **bounceType**: See http://docs.aws.amazon.com/ses/latest/DeveloperGuide/notification-contents.html#bounce-types
    * **reportingMTA**: See http://docs.aws.amazon.com/ses/latest/DeveloperGuide/notification-contents.html#bounce-object
    * **bouncedRecipients**: See http://docs.aws.amazon.com/ses/latest/DeveloperGuide/notification-contents.html#bounce-object.
    * **timestamp**: ISO8601 format timestamp.
    * **feedbackId**: unique id for this bounce.
  - **DeliveredDate**: Unix time this message was delivered at.
- **Next**: Used to get the next set of data. Pass in as query param 'Next'.
- **Error**: Optional error string

Example Returns (200 OK):

```
{
    "Emails": [
        {
            "Email": "example@example.com",
            "Type": "tokenAndGroup",
            "MessageId": "<message uuid>",
            "SentAt": 1449791030,
            "LastUpdated": 1449792030,
            "State": "delivered",
            "BrandId": "<brand uuid>",
            "OwnerId": "<owner uuid>",
            "MsgDetails": {
                "To": "example@example.com",
                "Type": "tokenAndGroup",
                "BrandId": "<brand uuid>",
                "Token": "abcde",
                "NumPhotos": 10,
                "GroupId": "<group uuid>"
            },
            "ComplaintType": "",
            "BounceType": "",
            "BounceSubType": "",
            "DeliveredDate": 1449792030
        },
        ...
    ]
}
```

CHAPTER 6

Groups

Groups are the Owner visible arrangement of 'Albums'. A Group contains a set of photos that can be sold. Each Group has 1 parent Brand.

/v1/Group/

GET

Returns the list of Groups for an owner on the given day.

Query Params

Fields:

- **when**: Optional. Date string (Eg. 2015-12-31) for which day to return. Default: Today.

Returns

Fields

- **Groups**: A list of groups for the given day.
  - See *Get Group Returns* below for field definitions.
- **Error**: Optional error string.

Example Returns (200 OK):

```
{
    "Groups": [
        {
            "OwnerId": "<owner uuid>",
            "GroupId": "<group uuid>",
```

```
            "Name": "Name of the group",
            "Token": "df3Gas",
            "Count": 8,
            "Thumb": "/dsfsdfas/sfsdfsf.jpg",
            "BrandId": "<brand uuid>",
            "WhenUnix": 1449791030,
            "Deleted": false,
            "LastUpdatedUnix": 1449791030
        }
    ],
    "Error": "optional error string"
}
```

POST

Creates a new Group for the owner.

PostData

Fields

- **Name**: Optional. Name of the group.
- **BrandId**: Optional. Brand to associate this group with. Default: Owner's Primary Brand.
- **WhenUnix**: Optional. What date this group should fill in. Default: Now.

PostData Example:

```
{
    "Name": "name of the group",
    "BrandId": "<brand uuid>",
    "WhenUnix": 1449791030
}
```

Returns

Fields

- See *Get Group Returns* below for field definitions.

Example Return (201 Created):

```
{
    "OwnerId": "<owner uuid>",
    "GroupId": "<group uuid>",
    "Name": "Name of the group",
    "Token": "df3Gas",
    "Count": 8,
    "Thumb": "/dsfsdfas/sfsdfsf.jpg",
    "BrandId": "<brand uuid>",
    "WhenUnix": 1449791030,
    "Deleted": false,
    "LastUpdatedUnix": 1449791030
}
```

/v1/Group/Search

GET

Performs a basic search for the given group Token.

Query Params

Fields:

- **q**: The Token to search for.

Returns

Fields

- **Groups**: A list of matching groups. See *Get Group Returns* below for field definitions.
- **Error**: Optional error string.

Example Return (200 OK):

```
{
    "Groups": [
        {
            "OwnerId": "<owner uuid>",
            "GroupId": "<group uuid>",
            "Name": "Name of the group",
            "Token": "df3Gas",
            "Count": 8,
            "Thumb": "/dsfsdfas/sfsdfsf.jpg",
            "BrandId": "<brand uuid>",
            "WhenUnix": 1449791030,
            "Deleted": false,
            "LastUpdatedUnix": 1449791030
        },
        ...
    ],
    "Error": "error string"
}
```

/v1/Group/<group uuid / token>

Used to manipulate an individual group. Some methods require the full Group UUID, while others work with the Token or the UUID.

GET

Returns the given Group.

Returns

Fields

- **OwnerId**: Owner UUID of this group.
- **GroupId**: Group UUID.
- **Name**: Name of this group.
- **Token**: Token / Short ID of this group.
- **Count**: Number of photos in the group.
- **Thumb**: The URL suffix of the thumbnail of this group.
- **BrandId**: Brand UUID that this group belongs to.
- **WhenUnix**: Date this group was created for.
- **Deleted**: Optional. Boolean if this group was deleted.
- **LastUpdatedUnix**: The date this group was last updated on.

Example Return (200 OK):

```
{
    "OwnerId": "<owner uuid>",
    "GroupId": "<group uuid>",
    "Name": "Name of the group",
    "Token": "df3Gas",
    "Count": 8,
    "Thumb": "/dsfsdfas/sfsdfsf.jpg",
    "BrandId": "<brand uuid>",
    "WhenUnix": 1449791030,
    "Deleted": false,
    "LastUpdatedUnix": 1449791030
}
```

PUT

Updates the given group. **Requires** that the Group UUID be provided.

PostData

Fields

- **Thumb**: Optional. The image thumbnail to set. If empty not changed.
- **Name**: Optional. Name to set on the group. If empty not changed.
- **BrandId**: Optional. The Brand to associate with this group. If empty not changed.

PostData:

```
{
    "Thumb": "/dasdf/asdfdasf.jpg",
    "Name": "New name of group",
    "BrandId": "<brand uuid>"
}
```

Returns

Fields

- **Error**: Optional error string.

Example Return (200 OK):

```
{
    "Error": "Error string"
}
```

DELETE

Deletes the given group. **Requires** that the Group UUID be provided.

Returns

Fields

- **Error**: Optional error string.

Example Return (200 OK):

```
{
    "Error": "Error string"
}
```

/v1/Group/<group uuid>/Zip

POST

Downloads the given photos from the group into a zip file. Returns a zip process. Check CloudConvert in order to determine how to check the status of the zip progress.

PostData

Fields

- **Photos**: List of photos to include in the zip file.
  - **File**: The URL suffix of the file to zip.
  - **FileName**: What to name this file.

Example PostData:

```
{
    "Photos": [
        {
            "File": "/dsfsf/dsfafds.jpg",
            "FileName": "IMG_01.jpg"
        },
        {
```

```
            "File": "/dsfsf/hjkhkhjkh.jpg",
            "FileName": "IMG_02.jpg"
        },
        ...
    ]
}
```

Returns

Fields

- **Status**: URL to check zip progress at. Note: this is a cloudconvert url. See https://cloudconvert.com/apidoc#status
- **Error**: Optional error string.

Example Return (201 Created):

```
{
    "Status": "<cloud convert process url>",
    "Error": "Error string"
}
```

/v1/Group/<group uuid>/Photo

GET

Returns all the photos inside the given group. If not authorized, then "Photos" are returned as "Public" and some fields are hidden.

Returns

Fields

- **Photos**: A list of Photos. Returned when authorized. See *Get Photo Returns* for inner field definitions.
- **Public**: A list of Photos with some data filtered out. Returned when unauthorized.
- **Error**: Optional error string.

Example Return (200 OK):

```
{
    "Photos": [
        {
            "OwnerId": "<owner uuid>",
            "GroupId": "<group uuid>",
            "PhotoId": "<photo uuid>",
            "Created": 1449791030,
            "Medium": "/sdfadsf/asfsdaf.jpg",
            "Original": "/adfdaf/sadfsfs.jpg",
            "Small": "/sdfsaf/sdafafs.jpg",
            "Watermark": "/dfsdfsadf/sdfsaf.jpg",
            "WatermarkSmall": "/asdfsaf/sdfasf.jpg",
            "Deleted": false,
```

```
            "SortOrder": "IMG_01",
            "Ratio": 1.5
        },
        ...
    ],
    "Error": "Error string"
}
```

POST

Request to add photos to the given group. This is best done in batches of 25. The returned list provides a url for each photo to be uploaded to. **Note**: The number of requested photos may not match the number of returned items.

PostData

Fields

AddPhotos: number of photos you wish to upload. Range: 1-25 inclusive.

Example:

```
{
    "AddPhotos": 12
}
```

Returns

Fields

- **AddedPhotos**: A list of upload metadata. May not match the # of photos requested.
  - **PhotoId**: The UUID of the photo.
  - **UploadUrl**: The URL to PUT the multipart upload. See *Upload Format* for how to correctly upload content.
  - **Expires**: The time at which the upload URL expires and cannot be used anymore.
- **Error**: Optional error string.

Example Return (200 OK):

```
{
    "AddedPhotos": [
        {
            "PhotoId": "<photo uuid>",
            "UploadUrl": "https://..",
            "Expires": 1449791030
        },
        ...
    ],
    "Error": "Error string"
}
```

Upload Format

Upload data is expected in multipart format. There are 3 parts that are expected.

In the example below we are uploading for the Group "3acb836b-d45a-43e2-ba79-82f3cbf8b8d5", with the returned photo id of "694a7a02-3c8f-427f-9aee-15a8ca1073ef".

Eg:

```
------WebKitFormBoundaryTAfxe9HopeoFTY9R
Content-Disposition: form-data; name="GroupId"

3acb836b-d45a-43e2-ba79-82f3cbf8b8d5
------WebKitFormBoundaryTAfxe9HopeoFTY9R
Content-Disposition: form-data; name="PhotoId"

694a7a02-3c8f-427f-9aee-15a8ca1073ef
------WebKitFormBoundaryTAfxe9HopeoFTY9R
Content-Disposition: form-data; name="file"; filename="IMG_0124.JPG"
Content-Type: application/octet-stream

<file content goes here>
```

CHAPTER 7

Owner

Details about the root account holder, including their preferences and settings.

/Owner/

GET

Returns details about the Owner account.

Returns

Fields

- **OwnerId**: Owner UUID.
- **Created**: Unix time this account was created.
- **StripeSubStatus**: The status of this account: {“trialing”, “active”, “past_due”, “canceled”, “unpaid”}
- **CurrentPeriodEnd**: Unix time this account’s period ends.
- **CardDetails**: Optional. String description of attached card.
- **LastName**: Optional.
- **FirstName**: Optional.
- **Email**: Optional.
- **Plan**: Optional.
- **PrimaryBrand**: Optional. UUID of their ‘Primary’/Default brand.
- **Timezone**: Optional. Timezone they are in. Eg. America/Vancouver

Example Returns (200 OK):

```
{
    "OwnerId": "<owner uuid>",
    "Created": 1449791030,
    "StripeSubStatus": "trialing",
    "CurrentPeriodEnd": "1449891030",
    "CardDetails": "Visa ending in 4848",
    "LastName": "Example",
    "FirstName": "Ample",
    "Email": "example@example.com",
    "Plan": "growth-month",
    "PrimaryBrand": "<brand uuid>",
    "Timezone": "America/Vancouver"
}
```

PUT

Updates an Owner.

PostData

Fields

- **LastName**: Optional.
- **FirstName**: Optional.
- **Email**: Optional.
- **PrimaryBrand**: Optional. UUID of their 'Primary'/Default brand.
- **Timezone**: Optional. Timezone they are in. Eg. America/Vancouver

Example Post Data:

```
{
    "LastName": "Thor",
    "FirstName": "The",
    "Email": "thor@example.com"
}
```

Returns

Fields

- **Error**: Optional error string.

Example Returns (200 OK):

```
{
}
```

/Owner/Invoice/

GET

Returns the list of invoices associated with this account.

Returns

Fields

- **Items**: List of stripe Invoices. See https://stripe.com/docs/api#invoices
- **Error**: Optional error string.

/Owner/Invoice/<invoiceId>

GET

Returns details about the given invoice. Use 'upcoming' as the invoice id for retrieving the upcoming invoice.

Returns

Fields

- See https://stripe.com/docs/api#invoices
- **MoreChk**: Used to get more line items for this invoice.
- **Error**: Optional Error string.

/Owner/Invoice/<invoiceId>/Lines

GET

Returns more line details about the given invoice.

Query Params

Fields

- **customer**: Stripe Customer Id from invoice.
- **check**: Returned as *MoreChk*
- **last**: Last item received in the invoice list.

Returns

Fields

- **Items**: List of invoice items. See https://stripe.com/docs/api#invoice_lines

/Owner/Invoice/<invoiceId>/Pdf

GET

Returns the link to download a PDF invoice. Not all invoices will have PDFs.

Query Params

Fields

- **customer**: Stripe Customer Id from invoice.
- **check**: Returned as *MoreChk*

Returns

Fields

- **Link**: Link to the invoice.
- **Error**: Optional error string

/Owner/Sales

GET

Returns sales data for graphing.

Query Params

Fields

- **period**: Optional. Default 'day'. {'day', 'month', 'hour'}
- **rangeStart**: Optional. Default '-1w'. {'h': hour, 'd': day, 'w': week, 'm': month}
- **rangeEnd**: Optional. Default 'now'.
- **brandId**: Optional. Default 'all'.

Returns

Fields

- **Labels**: List of x-axis labels. Will be date strings according to period.
- **Data**: List of series and value data. Is a 2D array. First array [0, 1] matches the series indexes. Second array [0][0, 1, ...] is the y-values for the series data.
- **Series**: A list of series names.
- **Error**: Optional error string

Example Returns (200 OK):

```
{
    "Labels": ["2015-01", "2015-02"],
    "Data": [
        [0, 100],
        [10, 0]
    ],
    "Series": ["AmountCharged", "AmountRefunded"]
}
```

/Owner/Balance

GET

Returns all the connected stripe accounts' balances.

Returns

Fields

- **Accounts**: List of balances
  - **Name**: Name of account.
  - **Pending**: List of pending amounts.
    - **amount**: Amount int cents.
    - **currency**: Currency.
  - **Available**: List of available amounts.
    - **amount**: Amount int cents.
    - **currency**: Currency.
- **Error**: Optional error string.

Example Returns (200 OK):

```
{
    "Accounts": [
        {
            "Name": "Example",
            "Pending": [
                {
                    "Amount": 200,
                    "Currency": "USD"
                }
            ],
            "Available": [
                {
                    "Amount": 3500,
                    "Currency": "USD"
                }
            ]
        }
```

```
    ]
}
```

CHAPTER 8

Purchase

Purchase provides an endpoint to completing transactions through the PicThrive network.

/v1/Purchase/

POST

Create a purchase through Stripe or via Tokens. See Stripe.js to on how to collect Payment through Stripe. Use our Stripe Publishable Key:

```
pk_live_N4MLfAa4QdqxZtFbQF22N9h8
```

PostData

Fields

- **Groups**: A map of which GroupId's.
  - **key**: Map of Photo Id's that are selected
    - **value**: Boolean, always set to true.
- **Total**:
  - **NumPaid**: Number of photos paid for through Tokens.
  - **NumToPurchase**: Number of photos need to be paid via Stripe.
  - **NumPhotos**: Total number of photos being purchased.
  - **Groups**: List of Group details
    - **GroupId**: Id of a Group with photos.
    - **NumPaid**: Number of photos in this Group paid for through Tokens.

        - **NumToPurchase**: Number of photos in this Group that need to be paid via Stripe.
        - **NumPhotos**: Total number of photos being purchased from this Group.
- **TotalCents**: Total cost in Cents to charge through Stripe.
- **Email**: The email to send the order to.
- **Stripe**: Optional. The Stripe Token as returned by Stripe.js Only used when a Stripe purchase must be made.
- **Tokens**: List of Token Details that were used in this Purchase. See *Get Token* for more details about these fields.
  - **Token**: The actual Token code.
  - **OwnerId**: The Owner UUID of this token.
  - **GroupId**: The Group UUID this token is locked to. Leave empty if the Token is not locked to a Group.
  - **LockedToGroup**: Boolean. True if the token is locked to a Group.
  - **NumPhotos**: The number of photos this token can redeem.
  - **Type**: They type of token.
- **OwnerId**: The Owner UUID this purchase is taking place in.
- **BrandId**: The Brand UUID this purchase is taking place in.
- **Retry**: Optional. Array of string codes. Filled in by the Returned 'Retry' key.

Example PostData:

```
{
    "Groups": {
        "<group uuid>": {
            "<photo uuid>": true,
            "<photo uuid>": true,
            ...
        },
        ...
    },
    "Total": {
        "NumPaid": 5
        "NumToPurchase": 7
        "NumPhotos": 12
        "Groups": [
            {
                "GroupId": "<group uuid>",
                "NumPaid": 5
                "NumToPurchase": 1
                "NumPhotos": 6
            },
            ...
        ]
    },
    "TotalCents": 1500,
    "Email": "example@example.com",
    "Stripe": "<stripe token>",
    "Tokens": [
        {
            "Token": "sdfds",
            "OwnerId": "<owner uuid>",
            "GroupId": "<group uuid>",
            "LockedToGroup": true,
```

```
            "NumPhotos": 5,
            "Type": "available"
        },
        ...
    ],
    "OwnerId": "<owner uuid>",
    "BrandId": "<brand uuid>",
    "Retry": [
        "adfsf...",
        "gdfgdfg...",
        "gfhgfh..."
    ]
}
```

Returns

Fields

- **ChargeId**: Optional. Return on success fail Stripe purchase. This is their Charge / Purchase Id.
- **Cc**: Optional. Displayed Credit Card to show what card was charged. Eg: "Visa ending in 4242".
- **ChargedAmount**: The amount, in cents, that was charged to their credit card.
- **Retry**: Optional. Returned on purchase failure. To be used when retrying a failed purchase.
- **OrderId**: Optional. Returned only on success full 'payment'. Returns the UUID to their order page. Eg. https://order.picthrive.com/<order uuid>
- **Error**: Optional. Error string on why the purchase failed.
- **EmailError**: Optional. Present if the purchase was completed, but we failed to send out an email.
- **NumPhotos**: The number of photos in the purchase/order.
- **Email**: The email we sent their receipt to.

Example Return (201 Created):

```
{
    "ChargeId": "sdfsdfsfsdf",
    "Cc": "Visa ending in 4242",
    "ChargedAmount": 1500
    "Retry": [],
    "OrderId": "<order uuid>",
    "Error": "optional error string",
    "EmailError": "",
    "NumPhotos": 12,
    "Email": "example@example.com"
}
```

/v1/Purchase/Pending/

POST

Create a Pending Purchase that allows for payment to be done instore through their own POS. The PicThrive Owner will be charge as if a Token was used.

PostData

Fields

- **Groups**: See *Make Purchase* for this fields description.
- **TotalCents**: See *Make Purchase* for this fields description.
- **Email**: See *Make Purchase* for this fields description.
- **OwnerId**: See *Make Purchase* for this fields description.
- **BrandId**: See *Make Purchase* for this fields description.
- **Retry**: See *Make Purchase* for this fields description.

Example PostData:

```
{
    "Groups": {
        "<group uuid>": {
            "<photo uuid>": true,
            "<photo uuid>": true,
            ...
        },
        ...
    },
    "TotalCents": 1500,
    "Email": "example@example.com",
    "OwnerId": "<owner uuid>",
    "BrandId": "<brand uuid>",
    "Retry": []
}
```

Returns

Fields

- **Retry**: Optional. Returned on purchase failure. To be used when retrying a failed purchase.
- **PendingOrderId**: The unique id of this Pending Order.

Example Returns (201 Created):

```
{
    "Retry": [],
    "PendingOrderId": "safsdfs"
}
```

GET

Returns a list of Pending Purchases.

QueryParams

Fields

- **BrandId**: Optional. BrandId to only return.

Returns

Fields

- **Orders**: List of Pending Purchases.
  - **OwnerId**: Owner UUID.
  - **BrandId**: Brand UUID.
  - **GroupId**: List of Group UUIDs this purchase contains.
  - **PendingOrderId**: The unique id of this Pending Order.
  - **Modified**: Unix time this order was last modified.
  - **NumPhotos**: The number of photos in this order.
  - **Email**: The email this order will be sent to.
- **Error**: Optional error string.

Example Returns (200 OK):

```
{
    "Orders": [
        {
            "OwnerId": "<owner uuid>",
            "BrandId": "<brand uuid>",
            "GroupId": [
                "<group uuid>",
                ...
            ],
            "PendingOrderId": "<pending order id>",
            "Modified": 1449791030,
            "NumPhotos": 15,
            "Email": "example@example.com"
        },
        ...
    ]
}
```

/v1/Purchase/Pending/<pending order id>

PUT

Updates an pending order by either completing it or discarding it. If the order is approved a email will immediately be sent to the customer.

PostData

Fields

- **NewState**: New state of the Pending Order: {“approved”, “discarded”}.
- **EmailChange**: Optional. If not empty will override the Email already in the Pending order.

Example PostData:

```
{
    "NewState": "approved",
    "EmailChange": ""
}
```

Returns

Fields

- **Error**: Optional error string.

Example Returns (200 OK):

```
{
    "Error": ""
}
```

CHAPTER 9

Photos

The following endpoint describes manipulations on single photo items.

/v1/Photo/<photo uuid>

GET

Returns a single photo item.

Returns

Fields

- **OwnerId**: Owner UUID.
- **GroupId**: Group UUID this photo belongs to.
- **PhotoId**: Photo UUID.
- **Created**: Unix time this photo entry was created at.
- **Medium**: The URL suffix that the 'Medium' sized photo is located at.
- **Original**: The URL suffix that the 'Original' sized photo is located at.
- **Small**: The URL suffix that the 'Small' sized photo is located at.
- **Watermark**: The URL suffix that the 'Medium' Watermarked photo is located at.
- **WatermarkSmall**: The URL suffix that the 'Small' Watermarked photo is located at.
- **Deleted**: Optional. True if this photo has been deleted.
- **SortOrder**: The Sorting order of this photo. Tends to be the Filename of the photo.
- **Ratio**: The size ratio of this photo (Width / Height).

- **Region**: The region this photo is stored in. Useful to tell you how to display it. See Making Requests to find how to map regions.
- **Hue**: The overal 'color' code of the photo.

Example Returns (200 OK):

```
{
    "OwnerId": "<owner uuid>",
    "GroupId": "<group uuid>",
    "PhotoId": "<photo uuid>",
    "Created": 1449791030,
    "Medium": "/sdfadsf/asfsdaf.jpg",
    "Original": "/adfdaf/sadfsfs.jpg",
    "Small": "/sdfsaf/sdafafs.jpg",
    "Watermark": "/dfsdfsadf/sdfsaf.jpg",
    "WatermarkSmall": "/asdfsaf/sdfasf.jpg",
    "Deleted": false,
    "SortOrder": "IMG_01",
    "Ratio": 1.5,
    "Region": "picthrive-dyn"
}
```

DELETE

Deletes the given photo.

Returns

Fields

- **Error**: Optional error string

Example Returns (200 OK):

```
{
    "Error": "Error string"
}
```

CHAPTER 10

Tokens

/v1/Token/

GET

Returns a list of generated tokens in this account. Tokens can be redeemed for photos.

Query Params

Fields

- **Filter**: Optional. Default 'available'. {'available', 'archived', 'used'}
- **Count**: Optional. Search for tokens with this value on it. Use -1 for unlimited.
- **Next**: Optional. Used to retrieve the next page of results.

Returns

Fields

- **Tokens**: A list of tokens.
  - See *Get Token* for field descriptions.
- **Next**: Used to retrieve the next page of results.
- **Error**: Optional error string.

Example Return (200 OK):

```
{
    "Tokens": [
        {
            "OwnerId": "<owner uuid>",
```

```
            "Type": "available",
            "TypeAndModified": "available-1449791030",
            "TypeAndNum": "available-55",
            "Token": "abcde",
            "GroupId": "<group uuid>",
            "GroupToken": "pdsfs",
            "LockedToGroup": false,
            "NumPhotos": 55
        }
    ]
}
```

POST

Creates more tokens for this Owner. Can be done in bunches. Maximum of 256 at a single time.

PostData

Fields

- **Count**: The number of tokens to generate.
- **NumPhotos**: The photo value of the token. -1 means unlimited, and forces LockedToGroup to true.
- **LockedToGroup**: If the token should be locked to be used in a single group.
- **GroupId**: Optional. If the Token should link to this group.
- **GroupToken**: Must be specified if and only if GroupId is specified. Must be that Group's Token/Short Id.
- **Archive**: Optional. True if the tokens should immediately enter 'Archived' type.

Example PostData:

```
{
    "Count": 10,
    "NumPhotos": 11,
    "LockedToGroup": false,
    "GroupId": "<group uuid>",
    "GroupToken": "pdsfs",
    "Archive": false
}
```

Returns

Fields

- **Tokens**: List of generated Tokens. May not match requested count.
  - See *Get Token* for field descriptions.
- **Error**: Optional error string.

Example Returns (201 Created):

```
{
    "Tokens": [
        {
            "OwnerId": "<owner uuid>",
            "Type": "available",
            "TypeAndModified": "available-1449791030",
            "TypeAndNum": "available-11",
            "Token": "abcde",
            "GroupId": "<group uuid>",
            "GroupToken": "pdsfs",
            "LockedToGroup": false,
            "NumPhotos": 11
        },
        ...
    ]
}
```

/v1/Token/<token>

GET

Returns info about the specified token.

Returns

Fields

- **OwnerId**: Owner UUId this token belongs to.
- **Type**: Type of token: {“available”, “used”, “archived”}
- **TypeAndModified**: Type + Unix time this token was last updated.
- **TypeAndNum**: Type + # of photos in this token.
- **Token**: The actual token string.
- **GroupId**: Optional. If this token is linked to a Group, this will be specified.
- **GroupToken**: Optional. If this token is linked to a Group, this will be specified.
- **LockedToGroup**: Optional. If set to true this token can only be used on the given Group.
- **NumPhotos**: The number of photos this token is worth. -1 => unlimited.
- **OrderId**: Optional. Only set if the token has been redeemed. Will specify the Order Id.

Example Return (200 OK):

```
{
    "OwnerId": "<owner uuid>",
    "Type": "available",
    "TypeAndModified": "available-1449791030",
    "TypeAndNum": "available-11",
    "Token": "abcde",
    "GroupId": "<group uuid>",
    "GroupToken": "pdsfs",
    "LockedToGroup": false,
```

```
    "NumPhotos": 11
}
```

PUT

Update a token's state. A token can not legally transition out of 'used'.

PostData

Fields

- **OldState**: Previous state of token.
- **NewState**: New state of token.

Example PostData:

```
{
    "OldState": "available",
    "NewState": "used"
}
```

Returns

Fields

- **Error**: Optional error string.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search