audata

Audata Promo API Documentation

Version 1.9. June 2020.

Commercial in Confidence.

1. Overview

ABOUT AUDATA PROMO

Audata Promo is a cloud-based, software-as-a-service application that helps broadcast radio & media businesses award and manage prizes, and collect data from prize winners using SMS.

Unlike a conventional prize management system or an alternative such as a spreadsheet, busy on-air staff (producers, presenters) don't need to collect and enter all the details from prize winners. Instead, a user enters a winners mobile phone number and Audata Promo will send the winner an SMS with a unique link to provide their details and confirm their prize.

ABOUT THE AUDATA PROMO API

Audata Promo offers a comprehensive API to allow developers to integrate our platform with third-party software and other enterprise systems.

The Audata Promo API is organised around REST, and uses predictable resource-oriented URL patterns, returns JSON responses, and uses standard HTTP verbs and response codes.

NOTE

Some API features are not available for all Audata plans or in all countries. Contact your Audata representative for more information.

2. Developer Support

For support with the Audata Promo API, email developers@audata.io.

3. Endpoint

This version of the Audata Promo API can be accessed at the below URL

```
https://promo.audata.io/api/v1/
```

4. Authentication

Audata uses API keys for authentication. Almost all data in Audata Promo is scoped by station, and each Station can have a single API key which allows unrestricted access to retrieve, update, create and delete all records belonging to that Station.

Because API keys are unrestricted and do not have role-based permissions like Users do, ensure you keep your keys highly secure and do not store them in version control or expose them publicly. All API requests must be made over HTTPS. Calls to the API using plain HTTP will be rejected.

To authenticate your request, pass your API key as a **HTTP Bearer Token**. In a curl request, this might look like...
You can generate your API key under *Administration > API Keys*. Generating a new API key is

```
  -H “Authorization: Bearer YOUR_API_KEY”
```

destructive and will delete the previous key, which will break any applications using that key. Take care when generating new keys.

5. Errors

The Audata API uses standard HTTP response codes to indicate the success or failure of a request.

In general, codes in the **2xx** range indicate success, codes in the **4xx** range indicate an error that failed due to the content of the request (such as a missing token), and codes in the **5xx** range indicate an error on the server, though these are rare.

6. Versioning

When we make backwards-incompatible changes to the API we will release a new version. The version number is specified in the request URL.7. Pagination
Top level API resources allow bulk fetches and include pagination support. By default, you will see the last 20 records, but you can supply the **page** and **per_page** parameters in your request to adjust what records are returned.

```
/prizes?per_page=10
```

For example, this request will show the last 10 prizes created...
And this request will show the following 10...

```
/prizes?per_page=10&page=2
```

For most resources, there is an upper limit of 100 records you can query in a single request.

When this document indicates that a certain request supports pagination, you can also supply these URL parameters.

8. Request Bodies

Generally speaking, you are usually required to supply a JSON object in your request body whenever making POST, PUT, or PATCH requests. If you're creating a Prize, for example, your request might look like the below.

```
POST /api/v1/prizes HTTP/1.1
Authorization: Bearer YOUR_API_KEY
Content-Type: text/plain; charset=utf-8

{
  "prize_type": "cash",
  "phone_number": "0431319208",
  "cash_amount": 50.00
}
```

9. Webhooks

OVERVIEW

Webhooks are HTTPS calls sent from Audata, to an external endpoint, when a certain action is taken or a record is changed. For example, when a Listener record is updated, you can trigger a request to your app, instead of polling Audata for changes periodically.

EXAMPLE WEBHOOK CALL

Audata Webhooks are HTTPS GET or POST requests. For GET requests, the data is passed as URL parameters, and for POST requests, the data is passed as JSON in the body of the request.

The below is an example of a common POST webhook (in this case, when a Listener is updated).

```
{
    "event": "Listener.updated",
    "sentAt": "2020-03-18T17:41:56.555+11:00",
    "station": “developerfm",
    "sender_id": 1788495,
    "sender": {
        "id": 1788495,
        "first_name": "John",
        "last_name": "Smith",
        "email": "jsmith@audata.io",
        "dob": “1990-07-29",
        "station_id": 10405,
        "gender": "male",
        "address1": "58 Stead St",
        "address2": "",
        "suburb": "South Melbourne",
        "postcode": "3205",
        "created_at": "2020-02-26T09:50:31.759+11:00",
        "updated_at": "2020-03-17T23:02:25.624+11:00",
        "uuid": "34c16742-b5a3-444f-954e-cc245564f86b",
        "phone_number": "0412123123",
        "full_address": “123 Stead St South Melbourne VIC 3205",
        "country": "AU",
        "state": "VIC"
    },
    "webhookCallId": "2ae54ecb-930f-4a7a-974d-d84788d9dd12"
}
```

The request body contains the following components:

event	string	A key describing what action has triggered the webhook. The format for event keys is RecordName.action - for example, Listener.updated or Prize.created. The supported actions are “created”, “updated”, and “deleted”.
sentAt	string	The time the webhook call was sent as a ISO 8601 string
station	string	The unique subdomain of the station that sent the webhook
sender_Id	integer	The ID of the object / record that sent the webhook
sender	dictionary	A JSON dictionary containing the data of the object / record that sent the webhook
webhookCallId	string	A unique UUID for this webhook call

CREATING A WEBHOOK

Webhooks can be defined in the Audata application without code.

1. Open the **Administration** panel.
2. Click **Webhooks** in the admin menu.
3. Click **New Webhook**.
4. Select the **Event** that will trigger the webhook to be called (for example “Listener.updated” or “Prize.created”).
5. Enter the **Endpoint** where you would like the webhook to be sent.
6. Select whether the webhook should be a HTTPS **GET** or **POST** request.
7. Click **Save Webhook**.

NOTE

Webhooks must be sent over HTTPS and the specified endpoint must begin with “https://“. Plain HTTP endpoints will be rejected.

10. Listeners

THE LISTENER OBJECT

A Listener is a profile record in Audata for a listener of the station they belong to. Usually these records are created when a listener wins a prize from a station or enters a competition, however they can also be manually created in the app or via the API,

id	integer	The primary key for the record
first_name*	string	The listeners first name
last_name*	string	The listeners last name
email*	string	The listeners email address
dob*	string	Listeners date of birth in YYYY-MM-DD format
gender*	string	Either “male” or “female”
address1*	string	The first line of the listeners address
address2	string or null	The second line of the listeners address (if provided)
suburb*	string	The listeners suburb or city name
postcode*	string	The listeners postal or zip code
state	string or null	The listeners state or null for countries without states
country*	string	The country of the listeners address as 2-letter ISO code
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record
phone_number*	string	The listeners phone number
custom_data	dictionary	A JSON dictionary of custom field attributes

CREATE A LISTENER

Endpoint

```
POST  /listeners
```

The below is an example of a request to create a Listener in your Audata database
In this example we're also including an example of a **custom_data** attribute called "newsletter_opt_in". Custom data attributes need to be created in the Audata Promo web application first before you can include them in API calls.

```
$ curl https://promo.audata.io/api/v1/listeners \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -X POST \
  -d listener[first_name]="John" \
  -d listener[last_name]="Smith" \
  -d listener[email]="john.smith@example.com" \
  -d listener[dob]="1985-07-29" \
  -d listener[gender]="male" \
  -d listener[address1]="123 Collins St" \
  -d listener[suburb]="Melbourne" \
  -d listener[state]="VIC" \
  -d listener[postcode]="3000" \
  -d listener[country]="AU" \
  -d listener[phone_number]="0431123123" \
  -d listener[custom_data][newsletter_opt_in]="true"
```

If the Listener is created successfully, the server will return a JSON object representing the Listener with the status **200**. Otherwise, the relevant HTTP error code will be returned along with a JSON dictionary containing information about the errors.

BROWSE LISTENERS

You can retrieve Listener records in batch with a GET request to /listeners

```
GET  /listeners
```

By default this will show the 20 most recent Listener records. You can use **page** and **per_page** parameters in the URL (see "Pagination").

RETRIEVE A LISTENER

`GET` `/listeners/listener_id`

You can retrieve a Listener by the ID of the Listener record
This will return a JSON object representing the listener and the status code **200** if successful.

SEARCH LISTENERS

If you don't know the ID for a Listener, you can search all listeners using a string query that can

`GET` `/listeners/search?query=your_search_query`

contain their name, email address, or phone number.
This will return an array of Listeners as JSON. You can also use pagination parameters on search requests.

UPDATE A LISTENER

Update a Listener in the same way you would create a new Listener, however using a PUT or PATCH request to the Listener record.

`PUT` `/listeners/listener_id`

You don't need to include all Listener attributes, only the ones you wish to update.
Here's an example of a request to update a Listener's email address.

```
$ curl https://promo.audata.io/api/v1/listeners/LISTENER_ID \
  -X PUT
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d listener[email]="new.email@example.com" \
```

DELETE A LISTENER

`DELETE` `/listeners/listener_id`

Delete a Listener by using the HTTP "DELETE" verb and the URL of the record...

11. Prizes

THE PRIZE OBJECT

A Prize is the record you create when you want to award a prize (either an inventory item such as a concert ticket, or a cash prize) to a listener.

There are 3 types of Prize in Audata - Cash, Inventory, and List prizes. This is represented in the “prize_type” attribute.

- Cash - a prize of a cash value.
- Inventory - an Inventory Item such as concert tickets, a gift card etc.
- List - the prize is being added to a List inside Audata, which is most commonly used to put listeners “in the draw” for another, larger prize.

id	integer	The primary key for the record
listener_id	integer or null	The ID of the Listener who has won the prize. This can be null before the listener has confirmed their details.
description	string	A description of the prize as a string.
phone_number	string	The phone number of the prize winner.
campaign_id	integer or null	The ID of the Campaign the prize belongs to.
status_id	integer or null	The ID of the current Status of the prize.
prize_type	string	The type of the prize: “cash”, “inventory” or “list”
cash_amount	decimal	The amount of cash won for a cash prize (in the default currency).
inventory_item_id	integer or null	If the prize is an Inventory prize, this will be the ID of the inventory item that was won.
inventory_qty	integer	The quantity of the inventory item that was won (if the prize is inventory).
list_id	integer or null	If the prize is a list prize, the ID of the list that the listener is being added to.
show_id	integer or null	The ID of the Show which awarded the prize.
custom_data	dictionary	A dictionary of additional prize details that were provided by the listener if requested.
value	decimal	The value of the prize in the default currency - this is the same as the “cash_amount” if a cash prize, or the value of the inventory item(s).

io_code	string	For finance - the IO code, or cost centre, imported from the campaign if present.
gl_code	string	For finance - the GL code imported from the campaign if present.
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the prize.

AWARD A PRIZE

To award a prize to a Listener, you create a new Prize record. Following REST conventions, this

```
POST  /prizes
```

is achieved by sending a POST request to the endpoint below:
The below is an example of a request to create a Prize. In this example, we're going to award a

```
$ curl https://promo.audata.io/api/v1/prizes \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -X POST \
  -d prize[prize_type]="cash" \
  -d prize[cash_amount]="50.00" \
  -d prize[phone_number]="0431123123" \
  -d prize[campaign_id]="3152006"
```

$50 cash prize to a winner when we only have their phone number.
In this example, because we provided a *phone_number* and not a *listener_id*, Audata Promo will send an SMS message to the winners phone number to collect their details. Audata automatically converts the phone_number you provide to its full international format for your country, so you don't need to include a country code. This is default behaviour - although some stations may have customised this workflow.

We also don't need to provide a status_id, as Audata Promo will assign the default one automatically, however you can if you choose to do so.

TIP

You can prevent Audata from automatically sending a notification to the winner by passing "**prevent_notifications=true**" as a URL param with this request.

NOTE

When you create a prize with the **cash** prize_type, Audata will automatically create a corresponding Payment record. If you delete the prize, Audata will also delete the associated Payment record.

LIST PRIZES

You can retrieve Prize records in batch with a GET request to /prizes

```
GET  /prizes
```

By default this will show the 20 most recent Prize records. You can use **page** and **per_page** parameters in the URL (see "Pagination").

FILTER PRIZES

You can add URL parameters to the /prizes endpoint to filter the prizes returned by **campaign_id**, **show_id**, or **inventory_item_id**.

```
GET  /prizes?inventory_item_id=501302
```

Again, the results returned will be paginated and support pagination parameters.

RETRIEVE A PRIZE

You can retrieve a Prize by the ID of the Prize record

```
GET  /prizes/prize_id
```

RETRIEVE PRIZES FOR LISTENER

You can retrieve a list of all prizes for one listener using the URL below. This will return a JSON Array of Prizes.

```
GET  /listeners/listener_id/prizes
```

DELETE A PRIZE

You can delete prize records using a DELETE request.

`DELETE` `/prizes/**prize_id**`

NOTE

To maintain an audit trail, we recommend instead of deleting prizes, that you change their status to “withdrawn”, or a similar status in your instance, and retain the original record.

12. Inventory Items

THE INVENTORY ITEM OBJECT

An Inventory Item is a prize you have available to award (excluding cash prizes, which do not require an inventory item to be created). Examples of inventory items in a radio station database might include concert tickets or gift cards.

id	integer	The primary key for the record.
description	string	The name / description of the inventory item.
qty_available	integer	The original quantity of the item available to award.
qty_on_hand	integer	The original quantity of the item physically on hand, or otherwise ready to collect or dispatch.
current_qty_available	integer	The current quantity of the item available (taking into consideration prizes that have been awarded).
current_qty_on_hand	integer	The current quantity of the item on hand (taking into consideration prizes that have been awarded).
campaign_id	integer or null	The ID of the campaign record that the item belongs to, or null if none is set.
vpu	decimal	Value per unit (in default currency) of the item.
sku	string	SKU (stock-keeping unit) which is not used by Audata, but can be set to help identify your item in a third-party system.
ad_hoc_enabled	boolean	Whether or not the prize can be awarded on an ad-hoc, or unscheduled, basis.
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record

CREATE AN INVENTORY ITEM

Create an Inventory Item following the same RESTful pattern as other resources.

```
POST  /inventory_items
```

The below is an example of a request to create an Inventory Item.

```
$ curl https://promo.audata.io/api/v1/inventory_items \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X POST \
  -d inventory_item[description]=“Test Concert Tickets” \
  -d inventory_item[qty_available]=“20” \
  -d inventory_item[qty_on_hand]=“20”
```

RETRIEVE AN INVENTORY ITEM

You can retrieve details for an Inventory Item using a GET call...

```
GET /inventory_items/inventory_item_id
```

UPDATE AN INVENTORY ITEM

```
PUT /inventory_items/inventory_item_id
```

```
$ curl https://promo.audata.io/api/v1/inventory_items/INVENTORY_ITEM_ID \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X PUT \
  -d inventory_item[sku]=“ABC123”
```

You can update details for an Inventory Item using a PUT or PATCH request...

DELETE AN INVENTORY ITEM

To delete an Inventory Item permanently, you can call the URL for the resource with a DELETE HTTP verb.

```
DELETE /inventory_items/inventory_item_id
```

However, if the inventory item has prizes associated with it, this call will fail. Instead, you can archive the inventory item.

ARCHIVE AN INVENTORY ITEM

You can archive inventory items by updating the “archived_at” attribute to a date either now or in the past. You can set a future date, and it will be archived automatically on that date. Once archived, the inventory item will still be searchable, and will be accessible via the API, but will no longer appear in the web application.

An example of an archive call via curl is below...

```
$ curl https://promo.audata.io/api/v1/inventory_items/INVENTORY_ITEM_ID \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X PUT \
  -d inventory_item[archived_at]=“2019-01-01”
```

SEARCH INVENTORY ITEMS

You can search Inventory Items using a string query, and the API will return a JSON array of results. These results are paginated.

```
GET  /inventory_items/search?query=your_query
```

13. Campaigns

THE CAMPAIGN OBJECT

A Campaign in Audata represents a promotion or contest on-air, and helps to organise and group related records together, such as inventory items and prizes.

id	integer	The primary key for the record
name	string	The name of the campaign
client_id	integer or null	The ID of the client associated with this campaign
budget	decimal	The budget (in default currency) for the campaign
code	string	This is a code you can reference in Liquid templates for notifications delivered by Audata
io_code	string	An IO code or cost centre for the campaign which will in turn be applied to any related prizes and payments
gl_code	string	A GL code for finance which will in turn be applied to any related prizes and payments.
owner_email	string	Email address of the user who owns the campaign.
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record

CREATE A CAMPAIGN

Create a Campaign following the same RESTful pattern as other resources.

`POST` `/campaigns`

The below is an example of a curl request to create a Campaign.

```
$ curl https://promo.audata.io/api/v1/campaigns \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X POST \
  -d campaign[name]=“Some Campaign Name”
  -d campaign[owner_email]=“adrian@audata.io”
```

RETRIEVE A CAMPAIGN

You can retrieve details for a Campaign using a GET call...

```
GET  /campaigns/campaign_id
```

UPDATE A CAMPAIGN

You can update details for a Campaign using a PUT or PATCH request...

```
PUT  /campaigns/campaign_id
```

```
$ curl https://promo.audata.io/api/v1/campaigns/CAMPAIGN_ID \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X PUT \
  -d campaign[budget]=“25000”
```

DELETE A CAMPAIGN

To delete a Campaign permanently, you can call the URL for the resource with a DELETE HTTP verb.

```
DELETE  /inventory_items/inventory_item_id
```

ARCHIVE A CAMPAIGN

You can archive Campaigns in the same manner as Inventory Items, by updating the “archived_at” attribute. Below is an example curl request from your terminal.

```
$ curl https://promo.audata.io/api/v1/campaigns/CAMPAIGN_ID \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X PUT \
  -d campaign[archived_at]=“2019-01-01”
```

SEARCH CAMPAIGNS

You can search Campaigns using a string query, and the API will return a JSON array of results. These results are paginated.

```
GET  /campaigns/search?query=your_query
```

14. Clients

THE CLIENT OBJECT

The Client object in Audata doesn't contain useful data about the client itself, as organisations usually prefer to store that information in an external CRM system or similar. In Audata, clients can be assigned to campaigns, so that all promotional activity for the same Client can be identified.

A common use case for the Client API resource is to bring information about clients Audata activity into a CRM system.

id	integer	The primary key for the record
name	string	The name of the client
account_id	string	An account ID you would use to identify the client in an external system (e.g. CRM)
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record

CREATE A CLIENT

Create a Client following the same RESTful pattern as other resources.

`POST` `/clients`

The below is an example of a curl request to create a Client.

```
$ curl https://promo.audata.io/api/v1/clients \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -X PUT \
  -d client[name]="XYZ Media Buyers"
  -d client[account_id]="XYX001"
```

RETRIEVE A CLIENT

You can retrieve details for a Client using a GET call...

`GET` `/clients/client_id`

UPDATE A CLIENT

You can update details for a Client using a PUT or PATCH request...

`PUT` `/clients/client_id`

```
$ curl https://promo.audata.io/api/v1/clients/CLIENT_ID \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X PUT \
  -d client[name]=“New Client Name”
```

DELETE A CLIENT

To delete a Campaign permanently, you can call the URL for the resource with a DELETE HTTP verb.

`DELETE` `/clients/client_id`

LIST CLIENTS

You can list clients ordered by most recently created with a simple GET request...
This query supports pagination.

`GET` `/clients`

15. Payments

THE PAYMENT OBJECT

Payments in Audata Promo represent a cash prize that needs to be paid (or has been paid) to a winner. Payments are different to other resources in that they can not be manually created or deleted, even via the API. Payments are automatically created by the system whenever a cash prize is awarded. If the prize is deleted, the associated payment is also removed within 24 hours.

You can update payments, but only the “paid_at” attribute to record a payment as having been paid. To change other details for a payment, you must instead delete the associated Prize and then create a new one.

id	integer	The primary key for the record
bank_account_id	integer or null	The ID of the Bank Account record that the payee has nominated to receive their payment into
listener_id	integer	The ID of the Listener record who has won the cash prize
prize_id	integer or null	The ID of the associated Prize record for this payment. If null, the prize has been deleted and this payment is also pending automatic deletion.
amount	decimal	The amount of the prize in default currency
paid_at	String or null	The time the payment was marked as paid as a ISO 8601 string. If null, indicates the payment has not yet been paid.
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record

LIST PAYMENTS

You can retrieve a list of recent payments using the below request
This request supports pagination.

```
GET  /finance/payments
```

RETRIEVE A PAYMENT

To retrieve information about a specific payment, use the Payment ID and the below request.

```
GET  /finance/payments/payment_id
```

UPDATING A PAYMENT

Update payment records to mark them as paid using a PUT request.

```
PUT  /finance/payments/payment_id
```

```
$ curl https://promo.audata.io/api/v1/payments/PAYMENT_ID \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X PUT \
  -d payment[paid_at]=“2019-01-01”
```

NOTE

You can only update the paid_at attribute for payments. To make other changes, remove your prize and create a new one.

Payments can not be created or deleted. They are automatically managed by the Prize record that created them.

GET BANK ACCOUNT DETAILS FOR PAYMENT

As well as retrieving bank account details by querying the bank_account_id, you can retrieve the bank account for a specific payment using the format below:

```
GET  /finance/payments/payment_id/bank_account
```

This will return the bank account details if one has been nominated for the payment or an error and the status 404 if not.

16. Bank Accounts

THE BANK ACCOUNT OBJECT

Bank Accounts in Audata represent a payee's bank account where a Payment for a cash prize should be deposited. Listeners are prompted to create and save their Bank Account details the first time they win a cash prize.

id	integer	The primary key for the record
listener_id	integer	The ID of the Listener record who the bank account belongs to
bsb	string or null	The BSB or other routing code for the account
account_name	string or null	The name on the bank account
institution	string or null	In Australia, the 3-letter ABA code of the bank account institution
location	string or null	In Australia, the location of the bank accounts branch
number	string	The account number of the bank account
payments_default	boolean or null	Whether or not this is the default payments account for this listener
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record

RETRIEVING A BANK ACCOUNT

You can retrieve an individual bank account by its ID, using the format below...

```
GET  /finance/bank_accounts/bank_account_id
```

RETRIEVING BANK ACCOUNTS FOR A LISTENER

You can retrieve a list of listeners bank accounts using...

```
GET  /listeners/listener_id/bank_accounts
```

CREATE A BANK ACCOUNT

Create a Bank Account following the same RESTful pattern as other resources.

`POST` `/finance/bank_accounts`

The below is an example of a curl request to create a Bank Account.

```
$ curl https://promo.audata.io/api/v1/finance/bank_accounts \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X POST \
  -d bank_account[listener_id]=“1234”
  -d bank_account[bsb]=“063123”
  -d bank_account[number]=“123456789”
  -d bank_account[payments_default]=“true”
  -d bank_account[account_name]=“JOHN EXAMPLE”
```

DELETE A BANK ACCOUNT

To delete a Bank Account permanently, you can call the URL for the resource with a DELETE HTTP verb.

`DELETE` `/finance/bank_accounts/`**`bank_account_id`**

NOTE

Bank Accounts can not be edited. If you need to amend a bank account, delete it and create a new one. Setting payments_default as true will automatically assign the Bank Account to any payments that do not have one nominated already.

17. Lists

Lists in Audata allow you to create a list of Listeners, which is often used for creating “prize draws” rather than awarding a prize upfront, however you could create Lists for a range of other purposes.

THE LIST OBJECT

id	integer	The primary key for the record
name	string	The name of the list
allow_duplicates	boolean or null	Whether or not the same listener can be added to the list more than once
created_at	string	The time the record was created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record

RETRIEVE LISTS

You can browse all your lists using a top-level query
This query supports pagination.

```
GET  /lists
```

RETRIEVE A SINGLE LIST

You can use a List ID to retrieve a single list

```
GET  /lists/list_id
```

RETRIEVE CONTENTS OF A LIST

While the above query will retrieve the details of a List, it will not return the actuall Listeners that belong to it. However, you can easily retrieve these using the below query.

```
GET  /lists/list_id/listeners
```

Results of this query are paginated, so you can use **page** and **per_page** URL parameters to select what range of results you want to return.

ADD LISTENERS TO A LIST

To add a Listener to a list, do a POST request to the below URL along with the Listener IDs. You can add a single Listener, or multiple in one request using an array of IDs.

```
POST  /lists/list_id/add_listeners
```

Here's an example request:

```
$ curl https://promo.audata.io/api/v1/lists/LIST_ID/add_listeners \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -X POST \
  -d list[listener_ids]="10,23,44"
```

CREATE A LIST

You can create a List with a POST request like this one

```
POST  /lists
```

```
$ curl https://promo.audata.io/api/v1/lists \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -X POST \
  -d list[name]="Competition Entry Draw"
```

UPDATE A LIST

You can update a List like any other resource with a PUT request like the below. However, you can not use this query to add or remove Listeners from the list. See above for the appropriate way to do that.

```
PUT  /lists/list_id
```

```
$ curl https://promo.audata.io/api/v1/lists \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -X POST \
  -d list[name]="Some Updated Name"
```

DELETE A LIST

You can delete the List (and its contents) using a DELETE query. Deleting a List does not delete the Listener records themselves.

```
DELETE  /lists/list_id
```

18. Forms

Audata Promo allows you to create custom Web Forms that can be embedded in your web sites and mobile apps for your audience to enter contests or otherwise collect their details.

THE FORM OBJECT

id	integer	The primary key for the record
name	string	The name of the form
campaign_id	integer	The ID of the campaign the form is associated with
allow_duplicates	boolean	Whether or not a single listener can register using the form more than once
list_id	integer	The ID of a list that you want to add form respondents to
redirect_url	string	The URL to redirect users to after completing the form
message	string	A message to display to users after completing the form
created_at	string	The time the record was first created as a ISO 8601 string
updated_at	string	The last time the record was modified as a ISO 8601 string
uuid	string	The UUID of the record

RETRIEVE FORMS

You can browse all your Forms using a top-level query
This query supports pagination.

`GET` `/forms`

RETRIEVE A SINGLE FORM

You can use a Form ID to retrieve a single Form

`GET` `/forms/form_id`

CREATE A FORM

`POST` `/forms`

You can create a Form with a POST request like this one

```
$ curl https://promo.audata.io/api/v1/forms \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X POST \
  -d form[name]=“Breakfast Contest Registrations”
  -d form[message]=“Thank you for registering!”
```

RETRIEVE FORM RESPONSES

`GET` `/forms/**form_id**/responses`

You can use the API to access the responses to a Form using the endpoint below:
By default you’ll only see the 20 most recent responses with this query, you need to use pagination parameters (see section 7 of this document) to access more.

Note that every Form response automatically creates a Listener record with the details the user provided. The **listener_id** as well as an object representing the Listener data will be included in the Form Response.

RETRIEVE SINGLE FORM RESPONSE

If you want to select just a single response and you have its ID, you can do so using a query like this:

`GET` `/forms/**form_id**/responses/**response_id**`

UPDATE A FORM

You can update a Form like any other resource with a PUT request like the below.

`PUT` `/forms/**form_id**`

```
$ curl https://promo.audata.io/api/v1/forms \
  -H “Authorization: Bearer YOUR_API_KEY” \
  -X POST \
  -d form[name]=“Some Updated Name”
```

DELETE A FORM

You can delete a Form using the standard query like the below. This will also delete any responses attached to the Form.

```
DELETE  /forms/form_id
```