FoodZaps Open APIs

Getting Started

The REST API lets you interact with FoodZaps Server from anything that can send an HTTP request. There are many things you can do with the REST API. For example:

- A website can access FoodZaps Server data from JavaScript.
- A web server can show data from FoodZaps Server on a website.
- You can download recent data to run your own custom analytics.

All API access is over HTTPS, and accessed via the `https://api.foodzaps.com` domain. The relative path prefix `/1/` indicates that we are currently using version 1 of the API.

URL	HTTP Verb	Functionality
/1/login	GET	Logging In
/1/logout	POST	Logging Out
/1/classes/order	GET	Get Transaction Details

Logging In

Before can start to query the data, user session is required which is through the Logging In.

A **sessionToken** will be created once logging in successfully.

Request

```
curl -X GET \
 -H "X-FoodZaps-Application-Id: ${APPLICATION_ID}" \
 -H "X-FoodZaps-REST-API-Key: ${REST_API_KEY}" \
 -H "X-FoodZaps-Revocable-Session: 1" \
 -G \
 --data-urlencode 'username=<username>' \
 --data-urlencode 'password=<password>' \
 https://api.foodzaps.com/1/login
```

Response

Key	Global
objectId	**USER_ID**
sessionToken	**SESSION_TOKEN**

Get Transaction Details

Request

Query the lastest 10 transaction

```
curl -X GET \
 -H "X-FoodZaps-Application-Id: ${APPLICATION_ID}" \
 -H "X-FoodZaps-REST-API-Key: ${REST_API_KEY}" \
 -H "X-FoodZaps-Session-Token: ${SESSION_TOKEN}"
    -G \
 --data-urlencode 'where={owner:${USER_ID}}' \
 --data-urlencode 'limit=10' \
    --data-urlencode 'order=-updatedAt' \
 --data-urlencode
'keys=create,modify,dish_name,status,order,global_id,controller,device' \
 https://api.foodzaps.com/1/classes/order
```

Response

JSON Arrays

Key	Type	Comment
create	Integer	timestamp in millisecond time which the order is created
modify	Integer	timestamp in millisecond time which the order is last updated
dish_name	Sting	Name of the dish
status	Integer	0 - Waiting for Confirmation 1 - Waiting for Cooking 2 - Cooking 3 - Waiting for Delivery 4 - Delivering 5 - Delivered 6 - Billed 7 - Paid 8 - Cancel 9 - Refund
order	Integer	Order/Receipt transaction number

global_id	Integer	Individual item transaction number
controller	String	Control Station ID
device	String	Device ID which the order is created

Request Format

For POST and PUT requests, the request body must be JSON, with the `Content-Type` header set to `application/json`.

Authentication is done via HTTP headers. The `X-FoodZaps-Application-Id` header identifies which application you are accessing, and the `X-FoodZaps-REST-API-Key` header authenticates the endpoint.

Name	Value
X-FoodZaps-Application-Id	e94ee179fd0ac94291bc950ebaa526c7cbe94c9b3aedf2ea50d91f48112b8d58
X-FoodZaps-REST-API-Key	yO9bEG0tVsg7xw9ed9CStpOy7uoiG0KHIFltxsnK

Response Format

The response format for all requests is a JSON object.

Whether a request succeeded is indicated by the HTTP status code. A 2xx status code indicates success, whereas a 4xx status code indicates failure. When a request fails, the response body is still JSON, but always contains the fields `code` and `error` which you can inspect to use for debugging.

For example, trying to save an object with invalid keys will return the message:

```
{
 "code": 105,
 "error": "invalid field name: lo^e"
}
```

Data Format

Data is built around a JSON encoding, schemaless and key-value pairs. Keys must be alphanumeric strings. Values can be anything that can be JSON-encoded. When you retrieve objects from Parse, some fields are automatically added: `createdAt`, `updatedAt`, and `objectId`. These field names are reserved. `createdAt` and `updatedAt` are UTC timestamps stored in ISO 8601 format with millisecond precision: `YYYY-MM-DDTHH:MM:SS.MMMZ`. `objectId` is a string unique to this class that identifies this object.

```
{
  "__type": "Date",
  "iso": "2011-08-21T18:02:52.249Z"
}
```

Data Type

The following types are allowed for each field in the object:

- String
- Number
- Boolean
- Arrays
- JSON Objects
- DateTime
- File
- Pointer to anther Parse Object
- Null

Query Constraints

There are several ways to put constraints on the objects found, using the **where** URL parameter. The value of the **where** parameter should be encoded JSON. Thus, if you look at the actual URL requested, it would be JSON-encoded, then URL-encoded. The simplest use of the **where** par ameter is constraining the value for keys. The **where** parameter supports the following options:

Key	Operation
$lt	Less Than
$lte	Less Than Or Equal To
$gt	Greater Than
$gte	Greater Than Or Equal To
$ne	Not Equal To
$in	Contained In
$nin	Not Contained in
$exists	A value is set for the key
$select	This matches a value for a key in the result of a different query
$dontSelect	Requires that a key's value not match a value for a key in the result of a different query
$all	Contains all of the given values
$regex	Requires that a key's value match a regular expression

In additional to **where**, there are several parameters you can use to configure what types of results are returned by the query.

Parameter	Use
order	Specify a field to sort by
limit	Limit the number of objects returned by the query
skip	Use with limit to paginate through results
keys	Restrict the fields returned by the query
include	Use on Pointer columns to return the full object

You can also use the **order** parameter to specify a field to sort by. Prefixing with a negative sign reverses the order.

You can use the **`limit`** and **`skip`** parameters for pagination. **`limit`** defaults to 100, but anything from 1 to 1000 is a valid limit.

You can restrict the fields returned by passing `keys` a comma-separated list.

Error Codes

Name

	Code	Description
`UserInvalidLoginParams`	101	Invalid login parameters. Check error message for more details.
`ObjectNotFound`	101	The specified object or session doesn't exist or could not be found. Can also indicate that you do not have the necessary permissions to read or write this object. Check error message for more details.
`InvalidQuery`	102	There is a problem with the parameters used to construct this query. This could be an invalid field name or an invalid field type for a specific constraint. Check error message for more details.
`InvalidClassName`	103	Missing or invalid classname. Classnames are case-sensitive. They must start with a letter, and a-zA-Z0-9_ are the only valid characters.
`MissingObjectId`	104	An unspecified object id.
`InvalidFieldName`	105	An invalid field name. Keys are case-sensitive. They must start with a letter, and a-zA-Z0-9_ are the only valid characters. Some field names may be reserved. Check error message for more details.
`InvalidPointer`	106	A malformed pointer was used. You would typically only see this if you have modified a client SDK.
`InvalidJSON`	107	Badly formed JSON was received upstream. This either indicates you have done something unusual with modifying how things encode to JSON, or the network is failing badly. Can also indicate an invalid utf-8 string or use of multiple form encoded values. Check error message for more details.
`CommandUnavailable`	108	The feature you tried to access is only available internally for testing purposes.
`NotInitialized`	109	You must call Parse.initialize before using the Parse library. Check the Quick Start guide for your platform.
`ObjectTooLarge`	116	The object is too large. Parse Objectss have a max size of 128 kilobytes.
`ExceededConfigParamsError`	116	You have reached the limit of 100 config parameters.
`InvalidLimitError`	117	An invalid value was set for the limit. Check error message for more details.
`InvalidSkipError`	118	An invalid value was set for skip. Check error message for more details.
`OperationForbidden`	119	The operation isn't allowed for clients due to class-level permissions. Check error message for more details.
`CacheMiss`	120	The result was not found in the cache.
`InvalidNestedKey`	121	An invalid key was used in a nested JSONObject. Check error message for more details.
`InvalidACL`	123	An invalid ACL was provided.
`InvalidEmailAddress`	125	The email address was invalid.
`DuplicateValue`	137	Unique field was given a value that is already taken.
`InvalidRoleName`	139	Role's name is invalid.
`ReservedValue`	139	Field value is reserved.
`ExceededCollectionQuota`	140	You have reached the quota on the number of classes in your app. Please delete some classes if you need to add a new class.
`ScriptFailed`	141	Cloud Code script failed. Usually points to a JavaScript error. Check error message for more details.

`FunctionNotFound`	141	Cloud function not found. Check that the specified Cloud function is present in your Cloud Code script and has been deployed.
`JobNotFound`	141	Background job not found. Check that the specified job is present in your Cloud Code script and has been deployed.
`SuccessErrorNotCalled`	141	success/error was not called. A cloud function will return once response.success() or response.error() is called. A background job will similarly finish execution once status.success() or status.error() is called. If a function or job never reaches either of the success/error methods, this error will be returned. This may happen when a function does not handle an error response correctly, preventing code execution from reaching the success() method call.
`MultupleSuccessErrorCalls`	141	Can't call success/error multiple times. A cloud function will return once response.success() or response.error() is called. A background job will similarly finish execution once status.success() or status.error() is called. If a function or job calls success() and/or error() more than once in a single execution path, this error will be returned.
`ValidationFailed`	142	Cloud Code validation failed.
`WebhookError`	143	Webhook error.
`InvalidImageData`	150	Invalid image data.
`UnsavedFileError`	151	An unsaved file.
`InvalidPushTimeError`	152	An invalid push time was specified.
`HostingError`	158	Hosting error.
`InvalidEventName`	160	The provided analytics event name is invalid.
`ClassNotEmpty`	255	Class is not empty and cannot be dropped.
`AppNameInvalid`	256	App name is invalid.
`MissingAPIKeyError`	902	The request is missing an API key.
`InvalidAPIKeyError`	903	The request is using an invalid API key.

Logging In

Request Method

GET

Enter request URL here and Input Parameters

https://api.foodzaps.com/1/login?username=**<username>**&password=**<password>**

Header & Value

Operation	Header	Value
Application ID	X-FoodZaps-Application-Id	e94ee179fd0ac94291bc950ebaa526c7cbe94c9b3aedf2ea50d91f48112b8d58
REST API Key	X-FoodZaps-REST-API-Key	yO9bEG0tVsg7xw9ed9CStpOy7uoiG0KHIFltxsnK
Revocable Session	X-FoodZaps-Revocable-Session	1

Response

{
"ANDROID_ID": "c3ee449392ce05ba",
"IMEI": "355306068785387",
"appVer": 9438,
"controller": "72030ac3-addb-45d7-a708-dd7d8afdd76b",
"createdAt": "2016-09-13T08:42:40.733Z",
"deviceLocale": "GB",
"email": "demo@foodzaps.com",
"emailVerified": true,
"fullName": "Bai Weiye",
"macAddress": "48:5A:3F:75:69:2B",
"objectId": "Mx2qGjqhHi",
"other": "{\"feedback\":\"Existing Business;Bar\"}",
"promoCode": "trial",
"reportHourOfDay": 0,
"reportMin": 0,
"sessionToken": "r:6lZZwpCHtjyVfvrtZ1wDp4gL3",
"transactionTotal": 1,
"updatedAt": "2016-09-13T09:37:18.139Z",
"username": "demo@foodzaps.com"
}
	Select **"objectId"** and **"sessionToken"**

Transaction Details API

Request Method

GET

Enter request URL here and Input Parameters

https://api.foodzaps.com/1/classes/order?where={owner:**<objectId>**}

Header & Value

Name	Header	Value
Application ID	X-FoodZaps-Application-Id	e94ee179fd0ac94291bc950ebaa526c7cbe94c9b3aedf2ea50d91f48112b8d58
REST API Key	X-FoodZaps-REST-API-Key	yO9bEG0tVsg7xw9ed9CStpOy7uoiG0KHIFltxsnK
Session Token	X-FoodZaps-Session-Token	**e.g.: r:6lZZwpCHtjyVfvrtZ1wDp4gL3** **(Please copy at Login API-> Response)**

Query Constraint (where{})

Key	Operation
$lt	Less Than
$lte	Less Than Or Equal To
$gt	Greater Than
$gte	Greater Than Or Equal To

$ne	Not Equal To
$in	Contained In
$nin	Not Contained in
$exists	A value is set for the key
$select	This matches a value for a key in the result of a different query
$dontSelect	Requires that a key's value not match a value for a key in the result of a different query
$all	Contains all of the given values
$regex	Requires that a key's value match a regular expression