Introduction

The WTvisitorOpinions API Documentation provides an easy way to connect your App to Back4App and use all the backend resources provided by Back4App Platform.

The API closely follows REST semantics, uses JSON to encode objects, relies on standard HTTP codes to signal operation outcomes and are mostly generated through Parse Server.

The API documentation below is specifically generated for your app WTvisitorOpinions by Back4App Platform. If you make changes to your app's backend schema (using Parse Server Dashboard, Back4App CLI or even these APIs), the API interface for those fields will change correspondingly, and those changes will be reflected here.

Getting Started

Since these are RESTful APIs, you are free to use your preferred technology to perform HTTP requests according to this documentation. Although we highly encourage you to use them through one of the Parse SDKs for a faster and better development experience. Please find below how to install each of the supported SDKs.

There is not a specific Parse SDK for cURL and you just need to make sure that cURL is installed in your machine. Most of the Operational Systems bring cURL installed by default, but you can also download the latest version of it in cURL Official Web-Site.

Please choose below one of the Parse SDKs and learn how to install it using the instructions that you can find in the right panel of this documentation.

It is not necessary to install any Parse SDK to call the API using cURL commands. To make sure that cURL is available at your machine, open a terminal window and execute a simple cURL command. If you do not have cURL available or want to use an updated version please visit their official website.

Installing JavaScript SDK

Installing Android SDK

Installing iOS SDK (Swift)

Installing iOS SDK (Objective-C)

Installing PHP SDK

Installing .NET SDK

Initializing Parse SDK

You do not need to do any kind of initialization when using cURL but you must send your app's credentials in the headers of all requests that you'll do. Example:

```
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
https://parseapi.back4app.com/serverInfo
```

You should not use the REST API Key in client apps. If you need to send a RETS API directly from your client-side code, you must use the Parse Client Key for your currently client-side platform (e.g. Client Key for iOS/Android, or .NET Key for Windows/Xamarin/Unity).

After installing the Parse SDK, you have to initialize the SDK using your App keys. You can copy them below from the right panel code snippets. At any time you can also find your App keys on your App dashboard under the Security & Keys menu. You don't need to initialize the SDK if you are using cURL, but you must send your app's credentials in the headers of any requests that you'll do.

APPLICATION ID

`ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

CLIENT KEY

`VRC33K9LgoYhFATorwKoZ6rrZxKerM60lEVlyCtM`

JAVASCRIPT KEY

`MiTtarrfIDmcQtRQkSirsXb7jp8AJDCxpsR0MawT`

.NET KEY

`0I7xpT7YJJextZ77ukqnrcJseAwDODkX8AVosu3n`

REST API KEY

`YjUjvxtx6OS9h8l8kKzeT05BMFWVkei76yGRvbK5`

FILE KEY

`69092a08-4065-426e-964e-9cea0b424509`

MASTER KEY

`vkCvGvWw3Sl70EOVrLyclgk2Cusrkf88jzyiiTMS`

Objects API

Example Object:

```
{
  "objectId": "4BwpMWdCnm",
  "myCustomKey1Name": "myCustomKey1Value",
  "myCustomKey2Name": "myCustomKey2Value",
  "createdAt": "2018-11-06T00:52:01.520Z",
  "updatedAt": "2018-11-06T00:52:04.713Z"
}
```

The special keys objectId, createdAt and updatedAt are default and always automatically created by the platform.

Performing CRUD (create, read, update and delete) operations through Back4App - Parse Server Hosting - is really simple and can be done using the Objects API. Each object consists of a set of key-value pairs that are transacted through the API as a JSON document. The keys must be alphanumeric strings and the values must be anything that can be JSON-encoded. The objects are organized in classes so you can distinguish different sorts of data.

The data that represents an object is schemaless which means that you don't need to specify ahead of time a new custom class schema. You just need to send the data and Parse will learn from it. But if for any reason you prefer to specify your class schema before sending the data, you can do that using the `Create a class` button in the Database Browser of your app's Parse Dashboard.

For each new custom class that you create in your app's schema (either through API or just sending the data), a new endpoint is generated at the address below through which you can perform your CRUD operations:

`https://parseapi.back4app.com/classes/MyCustomClassName`

In addition, a new section in this documentation is automatically generated for each new custom class that you create through which you can learn how to perform the CRUD operations specifically to one class' objects.

Please note that the User class' objects is a special case and has its own API whose documentation you can find here and endpoint is the one that you can find below:

`https://parseapi.back4app.com/users`

Creating Objects

Example Request:

```
curl -X POST \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
```

TRY ON SWAGGER

Example Response:

```
{
  "objectId": "4BwpMWdCnm",
  "createdAt": "2018-11-06T00:52:01.520Z"
}
```

To create a new object, you'll need to send a `POST` request to its class endpoint with your app's credentials in the headers and the object's data in the body. You can include as many key-value pairs as you want. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/classes/MyCustomClassName`

METHOD

`POST`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`Content-Type: application/json`

BODY

A JSON document with the key-value pairs that represent your object's data.

SUCCESS RESPONSE

STATUS

`201 Created`

HEADERS

`Location: https://parseapi.back4app.com/classes/MyCustomClassName/MyNewObjectId`

The `Location` header will contain the endpoint of the newly-created object.

BODY

A JSON document with the objectId and createdAt fields of the newly-created object.

ERROR RESPONSE

Please check the Errors section.

Reading Objects

```
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-G \
--data-urlencode "where={\"myCustomKey1Name\":\"myCustomKey1Value\"}" \
https://parseapi.back4app.com/classes/MyCustomClassName
```

TRY ON SWAGGER

Example Response:


```
{
  "results": [
    {
      "objectId": "zJxVP17mTi",
      "createdAt": "2018-10-31T14:16:13.616Z",
      "updatedAt": "2018-11-07T12:12:20.758Z",
      "myCustomKey1Name": "myCustomKey1Value",
      "myCustomKey2Name": ""
    },
    {
      "objectId": "yDbv0gKGJR",
      "createdAt": "2018-10-31T14:16:42.811Z",
      "updatedAt": "2018-11-07T12:12:18.543Z",
      "myCustomKey1Name": "myCustomKey1Value",
      "myCustomKey2Name": "myCustomKey2Value"
    },
    {
      "objectId": "xKue915KBG",
      "myCustomKey1Name": "myCustomKey1Value",
      "createdAt": "2018-11-07T12:11:58.533Z",
      "updatedAt": "2018-11-07T12:12:15.413Z",
      "myCustomKey2Name": "myCustomKey2Value"
    }
  ]
}
```


Without any URL parameters, this simply lists all objects in the class.

Learn more about query parameters in queries section.

To retrieve an object, you'll need to send a `GET` request to its class endpoint with your app's credentials in the headers and the query parameters in the URL parameters. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/classes/MyCustomClassName`

METHOD

`GET`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

PARAMETERS

A `where` URL parameter constraining the value for keys. It should be encoded JSON.

SUCCESS RESPONSE

HEADERS

`content-type: application/json;`

BODY

a JSON object that contains a results field with a JSON array that lists the objects.

ERROR RESPONSE

Please check the Errors section.

Updating Objects

Example Request:

```
curl -X PUT \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scx1RO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "Content-Type: application/json" \
-d '{"myCustomKeyName":"newValue"}' \
https://parseapi.back4app.com/classes/MyCustomClassName/Ed1nuqPvcm
```

TRY ON SWAGGER

Example Response:

```
{
  "updatedAt": "2011-08-21T18:02:52.248Z"
}
```

You can delete a single field from an object by using the **Delete** operation:

Example Request:

```
curl -X PUT \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scx1RO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "Content-Type: application/json" \
-d '{"myCustomKeyName":{"__op":"Delete"}}' \
https://parseapi.back4app.com/classes/MyCustomClassName/
```

Example Response:

```
{
  "updatedAt": "2011-08-21T18:02:52.248Z"
}
```

To update data on an object that already exists, send a `PUT` request to this object endpoint with your app's credentials in the headers and the query parameters in the body. Any keys you don't specify will remain unchanged, so you can update just a subset of the object's data. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

`https://parseapi.back4app.com/classes/MyCustomClassName/MyCurrentObjectId`

METHOD

`PUT`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`Content-Type: application/json`

BODY

A JSON document with the key-value pairs that represent the object's new data.

SUCCESS RESPONSE

STATUS

`200 OK`

HEADERS

`content-type: application/json;`

BODY

a JSON object that contains a updatedAt field with the timestamp of the update.

ERROR RESPONSE

Please check the Errors section.

Deleting Objects

Example Request:

```
curl -X DELETE \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
https://parseapi.back4app.com/classes/MyCustomClassName/Ed1nuqPvcm
```

TRY ON SWAGGER

Example Response:

```
{}
```

To delete an object send a `DELETE` request to this object endpoint with your app's credentials in the headers. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

`https://parseapi.back4app.com/classes/MyCustomClassName/MyCurrentObjectId`

METHOD

`DELETE`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

SUCCESS RESPONSE

STATUS

`200 OK`

HEADERS

`content-type: application/json;`

BODY

An empty JSON object.

ERROR RESPONSE

Please check the Errors section.

Understanding Data Types

Parse supports many types of data, from Strings to relations with other Parse Objects, users are able to store their data in our servers with great availability and low latency. Check the list aside with examples of how to define them and use on your application.

Examples:

```
{
  \"stringExample\": \"A string\",
  \"numberExample\": 1,
  \"booleanExample\": true,
  \"arrayExample\": [ 1, \"a string\" ],
  \"objectExample\": { \"foo\": \"bar\" },
  \"dateExample\": { \"__type\": \"Date\", \"iso\": \"2018-11-06T18:02:52.249Z\" },
  \"fileExample\": { \"__type\": \"File\", \"name\": \"resume.txt\" },
  \"pointerExample\": { \"__type\": \"Pointer\", \"className\": \"<YOUR_CLASS_NAME>\", \"objectId\": \"<THE_REFERENCED_OBJECT_ID>\" },
  \"relationExample\": { \"__type\": \"Relation\", \"className\": \"<YOUR_CLASS_NAME>\" },
  \"geopointExample\": { \"__type\": \"GeoPoint\", \"latitude\": 40.0, \"longitude\": -30.0 },
  \"polygonExample\": {\"__type\":\"Polygon\",\"coordinates\":[[0,0],[0,1],[1,0]]}
}
```

---

opinions Class

Example JSON:

```
{
  "name": "A string",
  "comment": "A string",
```

opinions is a custom class that was created and is specific for WTvisitorOpinions. Please use the following documentation to learn how to perform CRUD (create, read, update and delete) operations to this specific class. A new endpoint was automatically generated at the address below to which you can send your requests:

`https://parseapi.back4app.com/classes/opinions`

The following fields are supported by this class' schema and can be used in the operations:

Name	Type	Example
name	String	"A string"
comment	String	"A string"
willReturn	Boolean	true

Creating Objects

Example Request:

```
curl -X POST \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "Content-Type: application/json" \
-d "{ \"name\":\"A string\",\"comment\":\"A string\",\"willReturn\":true }" \
https://parseapi.back4app.com/classes/opinions
```

TRY ON SWAGGER

Example Response:

```
{
  "objectId": "4BwpMWdCnm",
  "createdAt": "2018-11-06T00:52:01.520Z"
}
```

To create a new object of the opinions class, you'll need to send a `POST` request to the opinions class' endpoint with your app's credentials in the headers and the object's data in the body. You can include as many key-value pairs of the supported fields as you want. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/classes/opinions`

METHOD

`POST`

HEADERS

`Content-Type: application/json`

BODY

A JSON document with the key-value pairs that represent your object's data according to the supported fields.

SUCCESS RESPONSE

STATUS

`201 Created`

HEADERS

`Location: https://parseapi.back4app.com/classes/opinions/MyNewObjectId`

The `Location` header will contain the endpoint of the newly-created object.

BODY

A JSON document with the objectId and createdAt fields of the newly-created object.

ERROR RESPONSE

Please check the Errors section.

Reading Objects

Example Request:

```
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-G \
--data-urlencode "where={ \"name\":\"A string\",\"comment\":\"A string\",\"willReturn\":true }" \
https://parseapi.back4app.com/classes/opinions
```

TRY ON SWAGGER

Example Response:

```
{
  "results": [
    {
      "objectId": "zJxVP17mTi",
      "createdAt": "2018-10-31T14:16:13.616Z",
      "updatedAt": "2018-11-07T12:12:20.758Z",
      "name": \"A string\","comment": \"A string\","willReturn": true
    },
    {
      "objectId": "yDbv0gKGJR",
      "createdAt": "2018-10-31T14:16:42.811Z",
      "updatedAt": "2018-11-07T12:12:18.543Z",
      "name": \"A string\","comment": \"A string\","willReturn": true
    },
    {
      "objectId": "xKue915KBG",
      "createdAt": "2018-11-07T12:11:58.533Z",
      "updatedAt": "2018-11-07T12:12:15.413Z",
      "name": \"A string\","comment": \"A string\","willReturn": true
    }
```

Without any URL parameters, this simply lists all objects in the class.

Learn more about query parameters in queries section.

To retrieve an object, you'll need to send a `GET` request to its class endpoint with your app's credentials in the headers and the query parameters in the URL parameters. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/classes/opinions`

METHOD

`GET`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

PARAMETERS

A `where` URL parameter constraining the value for keys. It should be encoded JSON.

SUCCESS RESPONSE

STATUS

`200 OK`

HEADERS

`content-type: application/json;`

BODY

a JSON object that contains a results field with a JSON array that lists the objects.

ERROR RESPONSE

Please check the Errors section.

Updating Objects

Example Request:

```
# Don't forget to set the OBJECT_ID parameter
curl -X PUT \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "Content-Type: application/json" \
-d "{ \"name\": \"A string\",\"comment\": \"A string\",\"willReturn\": true }" \
https://parseapi.back4app.com/classes/opinions/<OBJECT_ID>
```

TRY ON SWAGGER

```
{
  "updatedAt": "2011-08-21T18:02:52.248Z"
}
```

You can delete a single field from an object by using the **Delete** operation:

Example Request:

```
curl -X PUT \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "Content-Type: application/json" \
-d '{ "name": {"__op":"Delete"},"comment": {"__op":"Delete"},"willReturn": {"__op":"Delete"} }' \
https://parseapi.back4app.com/classes/opinions
```

TRY ON SWAGGER

Example Response:

```
{
  "updatedAt": "2011-08-21T18:02:52.248Z"
}
```

To update data on an object that already exists, send a `PUT` request to this object endpoint with your app's credentials in the headers and the query parameters in the body. Any keys you don't specify will remain unchanged, so you can update just a subset of the object's data. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/classes/opinions/MyCurrentObjectId`

METHOD

`PUT`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`Content-Type: application/json`

BODY

A JSON document with the key-value pairs that represent the object's new data.

SUCCESS RESPONSE

STATUS

`200 OK`

HEADERS

a JSON object that contains a updatedAt field with the timestamp of the update.

ERROR RESPONSE

Please check the Errors section.

Deleting Objects

Example Request:

```
# Don't forget to set the OBJECT_ID parameter
curl -X DELETE \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
https://parseapi.back4app.com/classes/opinions/<OBJECT_ID>
```

TRY ON SWAGGER

Example Response:

```
{}
```

To delete an object send a `DELETE` request to this object endpoint with your app's credentials in the headers. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/classes/opinions/MyCurrentObjectId`

METHOD

`DELETE`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

SUCCESS RESPONSE

STATUS

`200 OK`

HEADERS

`content-type: application/json;`

BODY

An empty JSON object.

User API

Example Object:

```
{
  "objectId": "4BwpMWdCnm",
  "username": "A string",
  "email": "A string",
  "password": "#Password123",
  "createdAt": "2018-11-06T00:52:01.520Z",
  "updatedAt": "2018-11-06T00:52:04.713Z"
}
```

The special keys objectId, createdAt and updatedAt are default and always automatically created by the platform.

User is a subclass of Object, what meanings that has the same properties and methods as a generic object. The difference is that Parse.User has some special additions specific to user accounts.

As others objects, users have a flexible schema, the differences are that the `username` and `password` fields are required and `username` and `email` must be unique per user.

For each new custom class that you create in your app's schema (either through API or just sending the data), a new endpoint is generated at the address below through which you can perform your CRUD operations:

`https://parseapi.back4app.com/user/MyCurrentUserObjectId`

Signing Up

Example Request:

```
# Don't forget to set your own User data!
curl -X POST \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "X-Parse-Revocable-Session: 1" \
-H "Content-Type: application/json" \
-d "{ \"password\":\"#Password123\", \"username\": \"A string\",\"email\": \"A string\" }" \
https://parseapi.back4app.com/users
```

TRY ON SWAGGER

Example Response:

```
{ "objectId":"nr7hAYS43a","createdAt":"2018-11-08T13:08:42.914Z","sessionToken":"r:35c2ae1c1def6c38a469e41ce671cb7e" }
```

To signing up a new user, you'll need to send a `POST` request to `/users` endpoint with your app's credentials in the headers and the object's data in the body. You can include as many key-value pairs as you want. This task can be easily accomplished just by calling the

To sign up a new user is a little bit different from creating a simple Parse Object in that the username and password fields are required and the last one must contain a capital letter, lowercase letter, a number and be at least 8 characters long.

REQUEST

URL

`https://parseapi.back4app.com/users`

METHOD

`POST`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`X-Parse-Revocable-Session: 1`

`Content-Type: application/json`

BODY

A JSON document with the key-value pairs that represent your object's data. It must contain a valid username and password.

SUCCESS RESPONSE

STATUS

`201 Created`

HEADERS

`Location: https://parseapi.back4app.com/users/MyNewUserId`

The `Location` header will contain the endpoint of the newly-created user.

BODY

A JSON document with the objectId, createdAt and sessionToken fields of the newly-created user.

ERROR RESPONSE

Please check the Errors section.

Logging In

Example Request:

```
# Don't forget to set the USERNAME and PASSWORD parameters!
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "X-Parse-Revocable-Session: 1" \
-G \
--data-urlencode 'username=<USERNAME>' \
--data-urlencode 'password=<PASSWORD>' \
https://parseapi.back4app.com/login
```

Example Response:

```
{
  "objectId":"AHRLeYvh0d",
  "username":"newUserName",
  "createdAt":"2018-11-08T13:50:56.843Z",
  "updatedAt":"2018-11-08T13:50:56.843Z",
  "sessionToken":"r:8d975a0f207fab1211752da3be0a3c81"
}
```

The response could contain another custom fields.

To log in into a user account you'll need to send a `GET` request to `/login` endpoint with your app's credentials in the headers and the username and password as URL parameters. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/login`

METHOD

`GET`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`X-Parse-Revocable-Session: 1`

PARAMETERS

Pass the `username` and `password` in URL parameters. It should be encoded JSON.

SUCCESS RESPONSE

STATUS

`200 OK`

BODY

A JSON document with the user's fields.

ERROR RESPONSE

Please check the Errors section.

Verifying Email

Example Request:

```
# Don't forget to set the target e-mail!
curl -X POST \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
```

TRY ON SWAGGER

Example Response:

```
{}
```

Enable email verification in server settings provides the app to reserve part of its experience just to users with confirmed email addresses. Email verification adds the emailVerified key to the Parse.User object. When a Parse.User's email is set or modified, emailVerified is set to false and an email is automatically sent to user email. After the validation process, the emailVerified is set to true.

Access Server Settings -> Verification Emails to enable automatic verification emails.

You can request a verification email to be sent by sending a `POST` request to `/verificationEmailRequest` endpoint with email in the body of the request:

Note that a verification email will not be sent if the email has already been successfully verified.

SIGN UP

To implement Sign Up with Email Verification, you will use the same method which you used to implement the basic user registration. But this time, instead of making an intent to the next page, you will ask the user to verify his or her email to login

Once the sign up process is completed, the user is automatically added to Parse Dashboard and its emailVerified Boolean attribute is set as true. Email verification is must as only you'll be allowed to let your users access your app entirely.

LOG IN

To implement Log In with Email Verification, you will use the same method which you used to implement the basic user registration. But this time, you will check the emailVerified boolean before granting further access to the user.

Requesting Password Reset

Example Request:

```
# Don't forget to set the target e-mail!
curl -X POST \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scx1RO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "Content-Type: application/json" \
-d '{"email":"email@example.com"}' \
https://parseapi.back4app.com/requestPasswordReset
```

TRY ON SWAGGER

Example Response:

You can initiate password resets for users who have emails associated with their account. To do this, send a `POST` request to `/requestPasswordReset` endpoint with email in the body of the request: This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/requestPasswordReset`

METHOD

`POST`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`Content-Type: application/json`

BODY

A JSON document with the user's `email`.

SUCCESS RESPONSE

STATUS

`200 OK`

BODY

An empty JSON document.

ERROR RESPONSE

Please check the Errors section.

Retrieving Current User

Example Request:

```
# Don't forget to set the session token, received
# when you logged in or signed up
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "X-Parse-Session-Token: <SESSION_TOKEN>" \
https://parseapi.back4app.com/users/me
```

TRY ON SWAGGER

Example Response:

```
{
  "username": "myuser123",
```

```
}
```

To retrieve the current user, you'll need to send a `GET` request to `users/me` endpoint with your app's credentials and the session token in the headers. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

REQUEST

URL

`https://parseapi.back4app.com/users/me`

METHOD

`GET`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`X-Parse-Session-Token: r:03a4c2d87a63a020a7d737c6fc64fd4c`

SUCCESS RESPONSE

STATUS

`200 OK`

BODY

A JSON document with all the user-provided fields, except password.

ERROR RESPONSE

Please check the Errors section.

Reading Users

Example Request:

```
# Don't forget to set the USER_OBJECT_ID parameter
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
https://parseapi.back4app.com/users/<USER_OBJECT_ID>
```

TRY ON SWAGGER

Example Response:

```
{
  "username":\"A string\",
  "email":\"A string\"

  "createdAt": "2018-11-07T20:58:34.448Z",
```

To retrieve users is almost the same way as retrieving a generic Parse.Object, you'll need to send a `GET` request to its `users` endpoint with your app's credentials in the headers. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

All the queries properties are applicable to retrieve Parse.User

REQUEST

URL

`https://parseapi.back4app.com/users/myCurrentUserId`

METHOD

`GET`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

SUCCESS RESPONSE

STATUS

`200 OK`

BODY

A JSON document with all the user-provided fields except password.

ERROR RESPONSE

Please check the Errors section.

Updating Users

Example Request:

```
# Don't forget to set the SESSION_TOKEN and USER_OBJECT_ID parameters
curl -X PUT \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "X-Parse-Session-Token: <SESSION_TOKEN>" \
-H "Content-Type: application/json" \
-d "{ \"username\": \"A string\",\"email\": \"A string\" }" \
https://parseapi.back4app.com/users/<USER_OBJECT_ID>
```

TRY ON SWAGGER

Example Response:

```
{
  "updatedAt": "2011-08-21T18:02:52.248Z"
```

You must provide the X-Parse-Session-Token header to authenticate.

To update data on an user that already exists, send a `PUT` request to this user endpoint with your app's credentials in the headers and the query parameters in the body. Any keys you don't specify will remain unchanged, so you can update just a subset of the object's data. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

> The Parse.User class is secured by default, you are not able to invoke `save` method unless the Parse.User was obtained using an authenticated method, like `logIn`, `signUp` or `current`

REQUEST

URL

`https://parseapi.back4app.com/users/MyUserObjectId`

METHOD

`PUT`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`X-Parse-Session-Token: myCurrentSessionToken`

`Content-Type: application/json`

BODY

A JSON document with the key-value pairs that represent the user's new data.

SUCCESS RESPONSE

STATUS

`200 OK`

HEADERS

`content-type: application/json;`

BODY

a JSON object that contains a updatedAt field with the timestamp of the update.

ERROR RESPONSE

Please check the Errors section.

Deleting Users

Example Request:

```
# Don't forget to set the SESSION_TOKEN and USER_OBJECT_ID parameters
curl -X DELETE \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
```

Example Response:

```
{}
```

You must provide the X-Parse-Session-Token header to authenticate.

To delete an user send a `DELETE` request to this user endpoint with your app's credentials in the headers. This task can be easily accomplished just by calling the appropriated method of your preferred Parse SDK. Please check how to do it in the right panel of this documentation.

> The Parse.User class is secured by default, you are not able to invoke `destroy` method unless the Parse.User was obtained using an authenticated method, like `logIn`, `signUp` or `current`

REQUEST

URL

`https://parseapi.back4app.com/users/MyCurrentObjectId`

METHOD

`DELETE`

HEADERS

`X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy`

`X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY`

`X-Parse-Session-Token: myCurrentSessionToken`

SUCCESS RESPONSE

STATUS

`200 OK`

HEADERS

`content-type: application/json;`

BODY

An empty JSON object.

ERROR RESPONSE

Please check the Errors section.

Queries

```
curl -X GET \
  -H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
  -H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
  https://parseapi.back4app.com/classes/MyCustomClass
```

Sometimes you will find that the **get** method from the Parse Object is not enough to retrieve the exact data that you want. In this section, we will explore the **Query** capabilities, which allows us to fetch an array of objects, apply different constraints to filter our query results, or even get unique results given a specific field.

Counting Objects

Counting objects is as simple as that:

```
curl -X GET \
  -H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
  -H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
  -G --data-urlencode "count=1" \
  https://parseapi.back4app.com/classes/MyCustomClass
```

If you do not need to retrieve any data from the objects, but count them, we have it covered.

Distinct Results

Example Request:

```
curl -X GET \
  -H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
  -H "X-Parse-Master-Key: vkCvGvWw3Sl70EOVrLyclgk2Cusrkf88jzyiiTMS" \
  -H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
  -G --data-urlencode 'distinct=likes' \
  https://parseapi.back4app.com/aggregate/Post
```

The query results should contain only unique emails

It is also possible to fetch unique results for a specific field, like so:

Constraints

Example Request:

```
# Finds the Posts with title equal to "My post title" and
# with likes greater than 100
curl -X GET \
  -H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
  -H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
  -G \
  --data-urlencode 'where={"title": "My post title", "likes": { "$gt": 100 }}' \
  https://parseapi.back4app.com/classes/Post
```

There are other operators than the $gt used above. Check the list below:

```
# $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
# $text - Performs a full text search on indexed fields

# Example request using multiple operators
# Finds all Posts with likes between 50 and 400
# and with title starting with "Parse"
curl -X GET \
  -H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
  -H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
  -G \
  --data-urlencode 'where={"likes":{"$gte":50,"$lte":400}, "title":{"$regex": "^Parse"}}' \
  https://parseapi.back4app.com/classes/Post
```

You can apply multiple constraints on a query, which will filter all the objects that do not match the conditions. It will be like an AND operator of constraints.

Ordering

Example Request:

```
curl -X GET \
  -H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
  -H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
  -G \
  --data-urlencode 'order=likes' \
  https://parseapi.back4app.com/classes/Post
```

It is also possible to sort the query results by a specific field:

Limit and Skip

Code:

```
# Finds 10 Posts, skipping the first 100 results
curl -X GET \
  -H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
  -H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
  -G \
  --data-urlencode 'limit=10' \
  --data-urlencode 'skip=100' \
  https://parseapi.back4app.com/classes/Post
```

If you need to limit the number of results or skip the first N results (useful when implementing pagination), you can do like so:

Relational Queries

Using object reference as a constraint:

```
-H "Content-Type: application/json" \
-d '{"content":"First post ! Yay!"}' \
https://parseapi.back4app.com/classes/Post

# Creates the Comment
curl -X POST \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "Content-Type: application/json" \
-d '{"content":"Congrats!", "post":{"__type":"Pointer","className":"Post","objectId":"<OBJECT_ID>"}}' \
https://parseapi.back4app.com/classes/Comment

# Finds only the comments related to the newly created post
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-G \
--data-urlencode 'where={"post":{"__type":"Pointer","className":"Post","objectId":"<OBJECT_ID>"}}' \
https://parseapi.back4app.com/classes/Comment
```

Creating relational constraints:

```
# Fetches only the comments whose posts contains image
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-G \
--data-urlencode 'where={"post":{"$inQuery":{"where":{"image":{"$exists":true}},"className":"Post"}}}' \
https://parseapi.back4app.com/classes/Comment
```

Creating relational constraints (negation):

```
# Fetches only the comments whose posts does NOT contains more than 50 likes
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-G \
--data-urlencode 'where={"post":{"$notInQuery":{"where":{"likes":{"$gt":50}},"className":"Post"}}}' \
https://parseapi.back4app.com/classes/Comment
```

Initializing a related object:

```
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-G \
--data-urlencode 'include=post' \
https://parseapi.back4app.com/classes/Comment

# You can also do multi level includes using dot notation. ex: include=post.author
```

Query supports creating constraints that evaluates if a field matches a particular Parse Object. It allows you to fetch only the objects that contains a specific relation with another object.

There are cases when you need to evaluate some related object fields in order to build a query with refined results.

That can be done by creating a inner query whose constraints will be applied only on the related field you want to evaluate, those are known as relational constraints and are also demonstrated in this section.

Using AND / OR operators

OR operator:

```
-G \
--data-urlencode 'where={"$or":[{"likes":{"$gt":1000}},{"title":"My great post"}]}' \
https://parseapi.back4app.com/classes/Comment
```

AND operator:

```
# Fetches the posts with more than 1000 likes OR posts with the title "My great post",
# and a specific user is the author of the post
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-G \
--data-urlencode 'where={"$or":[{"likes":{"$gt":1000}}, {"title":"My great post"}], "author": {"__type":"Pointer","className":"_User","objectId":"kzunnPFh5i"}}' \
https://parseapi.back4app.com/classes/Post
```

In order to create more complex queries, you will often need to create multiple constraints and combining them using logical operators OR and AND.

By using both operators, Query enables us to combine multiple subqueries, thus applying different sets of constraints. This way, we can achieve a higher level of complexity but also get even more refined results from our queries.

Aggregation

Grouping the results using a field value:

```
# Grouping the results by the number of likes.
# We need to use $ to indicate "content" is the name of the field
# we want to use as our objectId

curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "X-Parse-Master-Key: vkCvGvWw3Sl70EOVrLyclgk2Cusrkf88jzyiiTMS" \
-G \
--data-urlencode 'group={"objectId": "$content" }' \
https://parseapi.back4app.com/aggregate/Post

# The results are similar to the distinct feature
```

Aggregating values:

```
curl -X GET \
-H "X-Parse-Application-Id: ygvQv5SiQusJl5Ba5QIB6IFstE716eRGK3lfBOgy" \
-H "X-Parse-REST-API-Key: S7scxlRO6lBd4UECAIx2u9bvCyxGmjuADQvFOLSY" \
-H "X-Parse-Master-Key: vkCvGvWw3Sl70EOVrLyclgk2Cusrkf88jzyiiTMS" \
-G \
--data-urlencode 'group={"objectId":null,"total":{"$sum":"$likes"},"average":{"$avg": "$likes"},"maxLikes":{"$max": "$likes"},"minLikes":{"$min": "$likes"}}'
https://parseapi.back4app.com/aggregate/Post
```

For a more complete list of operators, please check the MongoDB documentation.

It is also possible to aggregate data into our queries. This is done by adding a pipeline of aggregations, where each aggregation can apply a transformation on the query results. In these aggregations, you can also apply calculations such as retrieving the average, minimum, maximum, and the sum of values.

Note that for the aggregate operation you must include the master key in the headers and send your request to `/aggregate/<YOUR-CLASS-NAME>`

Errors

This section provides a list of the Back4App API errors, with the error code and name, as well as brief details on how to properly handle errors. Make sure to inspect the returned error message for more specific information.

API Issues

Code	Name	Meaning
101	UserInvalidLoginParams	Invalid login parameters. Check error message for more details.
101	ObjectNotFound	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.
102	InvalidQuery	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.
105	InvalidFieldName	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.
107	InvalidJSON	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.
109	NotInitialized	You must call Parse.initialize before using the Parse library. Check the Quick Start guide for your platform.
116	ObjectTooLarge	The object is too large. Parse Objects have a max size of 128 kilobytes.
116	ExceededConfigParamsError	You have reached the limit of 100 config parameters.
117	InvalidLimitError	An invalid value was set for the limit. Check error message for more details.
118	InvalidSkipError	An invalid value was set for skip. Check error message for more details.
119	OperationForbidden	The operation isn't allowed for clients due to class-level permissions. Check error message for more details.
120	CacheMiss	The result was not found in the cache.
121	InvalidNestedKey	An invalid key was used in a nested JSONObject. Check error message for more details.
123	InvalidACL	An invalid ACL was provided.

125	InvalidEmailAddress	The email address was invalid.
137	DuplicateValue	Unique field was given a value that is already taken.
139	InvalidRoleName	Role's name is invalid.
139	ReservedValue	Field value is reserved.
140	ExceededCollectionQuota	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.
141	ScriptFailed	Cloud Code script failed. Usually points to a JavaScript error. Check error message for more details.
141	FunctionNotFound	Cloud function not found. Check that the specified Cloud function is present in your Cloud Code script and has been deployed.
141	JobNotFound	Background job not found. Check that the specified job is present in your Cloud Code script and has been deployed.
141	SuccessErrorNotCalled	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.
141	MultupleSuccessErrorCalls	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.
142	ValidationFailed	Cloud Code validation failed.
143	WebhookError	Webhook error.
150	InvalidImageData	Invalid image data.
151	UnsavedFileError	An unsaved file.
152	InvalidPushTimeError	An invalid push time was specified.
158	HostingError	Hosting error.
160	InvalidEventName	The provided analytics event name is invalid.
255	ClassNotEmpty	Class is not empty and cannot be dropped.
256	AppNameInvalid	App name is invalid.
902	MissingAPIKeyError	The request is missing an API key.
903	InvalidAPIKeyError	The request is using an invalid API key.

Code	Name	Meaning
200	UsernameMissing	Invalid login parameters. Check error message for more details.
201	PasswordMissing	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.
202	UsernameTaken	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.
203	UserEmailTaken	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.
204	UserEmailMissing	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.
205	UserWithEmailNotFound	You must call Parse.initialize before using the Parse library. Check the Quick Start guide for your platform.
206	SessionMissing	A user object without a valid session could not be altered.
207	MustCreateUserThroughSignup	A user can only be created through signup.
208	AccountAlreadyLinked	An account being linked is already linked to another user.
209	InvalidSessionToken	The device's session token is no longer valid. The application should ask the user to log in again.

General Issues

Code	Name	Meaning
-1	OtherCause	Invalid login parameters. Check error message for more details.
1	InternalServerError	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.
2	ServiceUnavailable	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.
4	ClientDisconnected	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.

Learning more

Since this documentation is not yet exhaustive and the team is still working to make it complete, you can find below a set of useful links through which you can learn about the APIs that are not yet documented here:

Back4App Documentation

Parse Open Source Documentation