1 Authentication

Taboola uses OAuth2 for authentication.
The idea is simple - request an Access Token from the Authorization Server, then attach the obtained Access Token as an HTTP header when making requests to the API.
All requests to the API must include an `Authorization` HTTP header, with its value containing the retrieved Access Token.

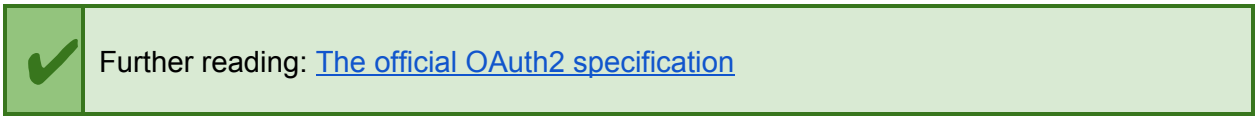
There are four possible ways to retrieve an Access Token:

1. Client Credentials: The client submits a Client ID + Client Secret pair to the token-endpoint, and in exchange receives an Access Token.
   This scheme is only appropriate when a *confidential* app performs server-to-server communication.

2. Password Authentication: The client submits a valid Username + Password pair to the token-endpoint, and in exchange receives an Access Token and a Refresh Token.
   This scheme is more appropriate for *non-confidential* client apps (e.g. web-apps, mobile apps).

3. Implicit Grant: The client redirects the user to a specific URL on Taboola Backstage, where the user authenticates with their username/password and approves access of the client app to their Taboola account. After confirmation, the user is redirected to a specific URL the client app supplied, with their Access Token attached to the fragment part of the URL.
   This scheme is more appropriate for *non-confidential* client apps (e.g. web-apps, mobile apps).

4. Authorization Code: The client redirects the user to a specific URL on Taboola Backstage, where the user authenticates with their username/password and approves access of the client app to their Taboola account. After confirmation, the user is redirected to a specific URL the client app supplied, with an Authorization Code appended as a query-parameter. The client-app then submits this Authorization Code to the token-endpoint and in exchange receives an Access Token and a Refresh Token.
   This scheme is more appropriate for *non-confidential* client apps (e.g. web-apps, mobile apps).



1.1 Prerequisites

1.1.1 Client ID

Before your app can send requests to the Authorization Server, you must obtain a Client ID from your Taboola contact person. Without a valid Client ID you won't be able to receive Access Tokens from the Authorization server.

The Client ID must be added to any request made at the Authorization Server, so it can know which 3rd-party is making this request.

1.1.2 Client Secret

If you are using the Client Credentials authentication scheme, you should also request a Client Secret.

The Client Secret is *confidential* and *should be kept a secret*.

1.2 Client Credentials

The Client Credentials scheme involves the client-app sending a client_id and client_secret in order to obtain an appropriate Access Token.

1.2.1 Request Structure

```
POST /backstage/oauth/token/
Host: https://backstage.taboola.com
Content-Type: application/x-www-form-urlencoded
```

```
?client_id=[client-id]&
    client_secret=[client-secret]&
    grant_type=client_credentials
```

1.2.2 Example Response

```
200 OK
```

```
{
  "access_token": "ab4Tk<saw\feaXcp53wF2ksasq12",
  "token_type": "bearer",
  "expires_in": 3600
}
```

When using the Client Credentials scheme, no `refresh_token` is supplied[+].

1.3 Password Authentication

The Password Authentication scheme involves the client-app sending a username and password in order to obtain an appropriate Access Token.

1.3.1 Request Structure

```
POST /backstage/oauth/token/
Host: https://backstage.taboola.com
Content-Type: application/x-www-form-urlencoded
```

```
?client_id=[client-id]&
    client_secret=[client-secret]&
    username=[username]&
    password=[password]&
    grant_type=password
```

1.3.2 Example Response

```
200 OK
```

```
{
  "access_token": "ab4Tk<saw\feaXcp53wF2ksasq12",
  "refresh_token": "dkjsERT\fck37dSFD<skjw@sddso",
  "token_type": "bearer",
  "expires_in": 3600
}
```

A failed attempt to login (wrong username/password) will result in a `400 Bad Request` error response, as per the OAuth2 spec.

1.4 Implicit Grant

The Implicit Grant scheme involves the client-app redirecting the user to the Authorization Server where they will be prompted with a confirmation screen. Upon confirmation, the user will again be redirected, this time to the `redirect_uri` parameter supplied by the client-app in the initial request. In addition, the URI fragment will contain the appropriate Access Token.

1.4.1 Flow

Redirect your users to the following URL:

```
GET /backstage/oauth/authorize/?
        client_id=[client-id]&
        redirect_uri=[redirect-uri]&
        response_type=token
Host: https://backstage.taboola.com
```

The user will be prompted to login if they are not already logged in to Taboola Backstage. Once authenticated, they will be redirected to a confirmation screen, asking them to allow the client-app access to their Taboola account.

If confirmed, the user will be redirected to the specified `redirect_uri`, with the Access Token attached to the Fragment. For example:

```
GET [redirect-uri]#access_token=[access-token]&expires_in=[expires-in]
```

When using the Implicit Grant scheme, no `refresh_token` is supplied[+].

When using the Implicit Grant scheme, whenever the current Access Token is invalidated, the user must go through the entire authentication flow again.

1.5 Authorization Code

The Authorization Code scheme is similar in flow to the Implicit Grant, with the difference that instead of directly obtaining an Access Token, the client-app receives an Authorization Code, which it can then use to obtain an Access Token and a Refresh Token from the token-endpoint.

Once generated, an Authorization Code has a 10-minutes expiry period. If 10 minutes pass and the Code was not used, it is revoked and cannot be used to obtain an Access Token.

1.5.1 Flow

Redirect your users to the following URL:

```
GET /backstage/oauth/authorize/?
        client_id=[client-id]&
        redirect_uri=[redirect-uri]&
        response_type=code
Host: https://backstage.taboola.com
```

The user will be prompted to login if they are not already logged in to Taboola Backstage. Once authenticated, they will be redirected to a confirmation screen, asking them to allow the client-app access to their Taboola account.

If confirmed, the user will be redirected to the specified `redirect_uri`, with the Access Token attached to the Query:

```
GET [redirect-uri]?code=[authorization-code]
```

Once the Authorization Code is obtained, the client-app can submit the code to the token-endpoint in exchange for a valid Access Token:

```
POST /backstage/oauth/token/
Host: https://backstage.taboola.com
Content-Type: application/x-www-form-urlencoded
```

```
?client_id=[client-id]&
    code=[authorization-code]&
    redirect_uri=[redirect-uri]&
    grant_type=authorization_code
```

The response will look similar to the response of a Password request:

```
200 OK
```

```
{
  "access_token": "ab4Tk<saw\feaXcp53wF2ksasq12",
  "refresh_token": "dkjsERT\fck37dSFD<skjw@sddso",
  "token_type": "bearer",
  "expires_in": 3600
}
```

1.6 Using the Access Token

After fetching the Access Token, the client should use it when making requests to the API. The token should be sent as an HTTP header called `Authorization` with the prefix `Bearer` :

```
GET /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

Once generated, an Access Token has a 12-hour expiry period. After 12 hours the Access Token is revoked, and the client-app should request another one, either by submitting a Refresh Token or by initiating a new authentication flow.

The Access Token should be attached to any request made at the API. An API call which is missing the `Authorization` header will result in a `401 Unauthorized` error response.

If a request to the API contains an invalid or expired Access Token, the request will result in a `401 Unauthorized` error response.

1.7 Using the Refresh Token to Refresh an Access Token

The Refresh Token is a long-lived token, which enables a client-app to obtain a new, valid Access Token, in case the current Access Token expires, is revoked, or lost.

When requesting a new Access Token using a valid Refresh Token, it is necessary for the request to be authenticated against the Authorization Server. The requesting entity can authenticate in several ways:

1. **Client Credentials**: If the request includes the `client_id` and `client_secret` fields, it will be authenticated against those credentials.
2. **Backstage Session (Cookies)**: If the request is made via a browser, and the requesting user is currently authenticated against the Backstage application, then the attached Cookie will be used for authentication.

1.7.1 Using Client Credentials

To refresh an Access Token, send an HTTP *POST* request in the following form:

```
POST /backstage/oauth/token/
```

```
Host: https://backstage.taboola.com
Content-Type: application/x-www-form-urlencoded
```

```
?refresh_token=[refresh-token]&
    client_id=[client-id]&
    client_secret=[client-secret]&
    grant_type=refresh_token
```

1.7.2 Using Backstage Session (Cookies)

```
POST /backstage/oauth/token/
Host: https://backstage.taboola.com
Content-Type: application/x-www-form-urlencoded
Cookie: JSESSIONID=[jsession-id]
```

```
?refresh_token=[refresh-token]&
    grant_type=refresh_token
```

Cookies are usually added automatically by the browser, so you probably don't need to add these yourself.

No `client_id` is required when refreshing a token with a session-based authentication, because it is inferred from the `refresh_token`.

1.7.3 Example Response

The response will look similar to the response of a Password request:

```
200 OK
```

```
{
  "access_token": "ab4Tk<saw\feaXcp53wF2ksasq12",
  "refresh_token": "dkjsERT\fck37dSFD<skjw@sddso",
  "token_type": "bearer",
  "expires_in": 3600
}
```

2 General Request/Response Description

This API enables you to manage your resources in Taboola's platform. It allows access to several resources, and enables management of those resources.

> ⚠ **Access Token**
>
> Before making any requests to the API, the client-app must obtain an Access Token as described in the Authentication chapter. After retrieving an Access Token, the client-app must attach it to every request, as an HTTP header named `Authorization`.

> ⚠ **SSL**
>
> Any requests to either the Authorization Server or the API itself *should always use SSL* (https://). The API will not respond to non-secured requests.

2.1 Requests

> ⚠ Most requests made at the API will require an `[account-id]` path-variable representing the Account which the resource belongs to. This parameter can be, unless otherwise noted, an `account_id` of any existing Account which the requesting user has permission to read/modify.

The general `GET` request form is:

```
GET /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

The general `POST` request form is:

```
POST /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
Content-Type: application/json
```

```
{
  "name": [campaign-name],
  ...
}
```

The general `PUT` request form is:

```
PUT /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
Content-Type: application/json
```

```
{
  "name": [campaign-name],
  ...
}
```

The body of *POST* or *PUT* requests should always be a valid JSON object, and the `Content-Type` header should be `application/json`.

When updating resources, it is possible to use either the `POST` method, or the `PUT` method. The API will respect both, but the documentation only uses the `POST` approach in examples.

The general `DELETE` request form is:

```
DELETE /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

2.2 Responses

Valid requests will receive a response with status `200 OK`.
The response body will always be a valid JSON object.

When fetching a *list* of resources, the response body will contain a JSON object with a `results` field - an array of resource objects, containing all the relevant resources. The response *might* include additional metadata fields, such as the timezone of the results, last-updated timestamp, etc. These fields are not documented and should not be relied upon.

All resources in a specific request result set will have the same structure.

```
200 OK
```

```
{
   "results": [
       // ... List of resources
   ]
}
```

When a *single* resource is requested, the response body will contain a JSON object representing the resource.

```
200 OK
```

```
{
    "id": [campaign-id],
    "name": [campaign-name]
    // ... more fields
}
```

2.3 Errors

Whenever an error occurs, the server will reply with the appropriate HTTP error status code. Error responses will contain a JSON object with information regarding the error in their body. The object will contain the internal error-code, and a human-readable message.

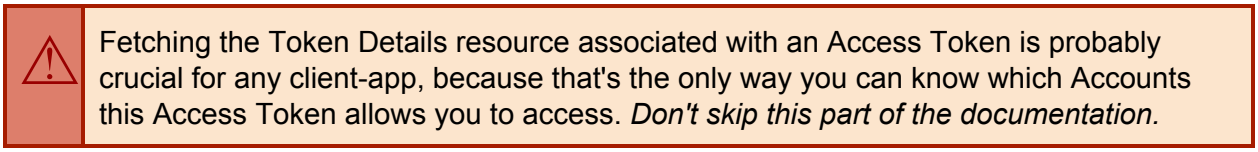
```
500 Internal Server Error
```

```
{
    "http_status": 500,
    "message": [error-message]
}
```

2.4 Dictionary

The dictionary allows to get from Taboola lists of possible values and their meaning in various contexts (enum's and their relevant codes). This is used in order to get predefined allowed values to be selected by the user. For example, if the user would like to target specific countries, the dictionary will allow to get the possible country values supported in Taboola and display them as selectable options to the user. The dictionary is covered in a specific chapter of the Taboola API.

3 Token Details



In order for a client-app to start fetching resources, it must obtain some basic details about the Token Holder (the user with which a specific Access Token is associated), namely the `[account-id]` of the Account the Token Holder belongs to. This is crucial for accessing the API, since all requested resources are requested for a specific Account, as described in General Description chapter.

The API only exposes a *read* route for fetching the Token Details associated with an Access Token. There's no way to fetch a *list* of different Token Details, and no way to *create*, *update*, or *delete* them - only *read*.

3.1 Token Details Resource

> The Token Details resource is *read-only*, meaning you can only fetch Token Details resources, but you cannot modify them via the API.

3.1.1 Fields List

Name	Type	Description
username	String	The username of the User associated with this Access Token
account_id	String	The account_id of the Account the associated User belongs to
full_name	String	The user full name
expires_in	Integer	Number of seconds left until this Access Token expires

3.2 Reading

To fetch a Token Details resource associated with a specific Access Token, send an HTTP *GET* request in the following form:

```
GET /backstage/api/1.0/token-details/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

3.2.1 Response

```
200 OK
```

```
{
    "expires_in": 43180,
    "username": "example@taboola.com",
    "account_id": "taboola-demo-account",
    "full_name": "Demo User Name",
}
```