Self Service Advertiser API – Technical Documentation

Version 1.0

Last Updated: 17-APRIL-2019

By: Intango LTD

General

This document describes the general contract of the SelfAdvertiser web app API.

It contains technical specification for the use of the API in 2 modes:

1. A session-less mode, where the user invokes the API using his real token.
2. A Sessioned mode, where the user invokes the API during his logged session, from within the execution environment inside the webapp. The execution environment serves as a hands-on client real client to perform actions , and as a documentation center for the API.

The API is based on REST architecture and principles.

Security

User validation is applied using a token.

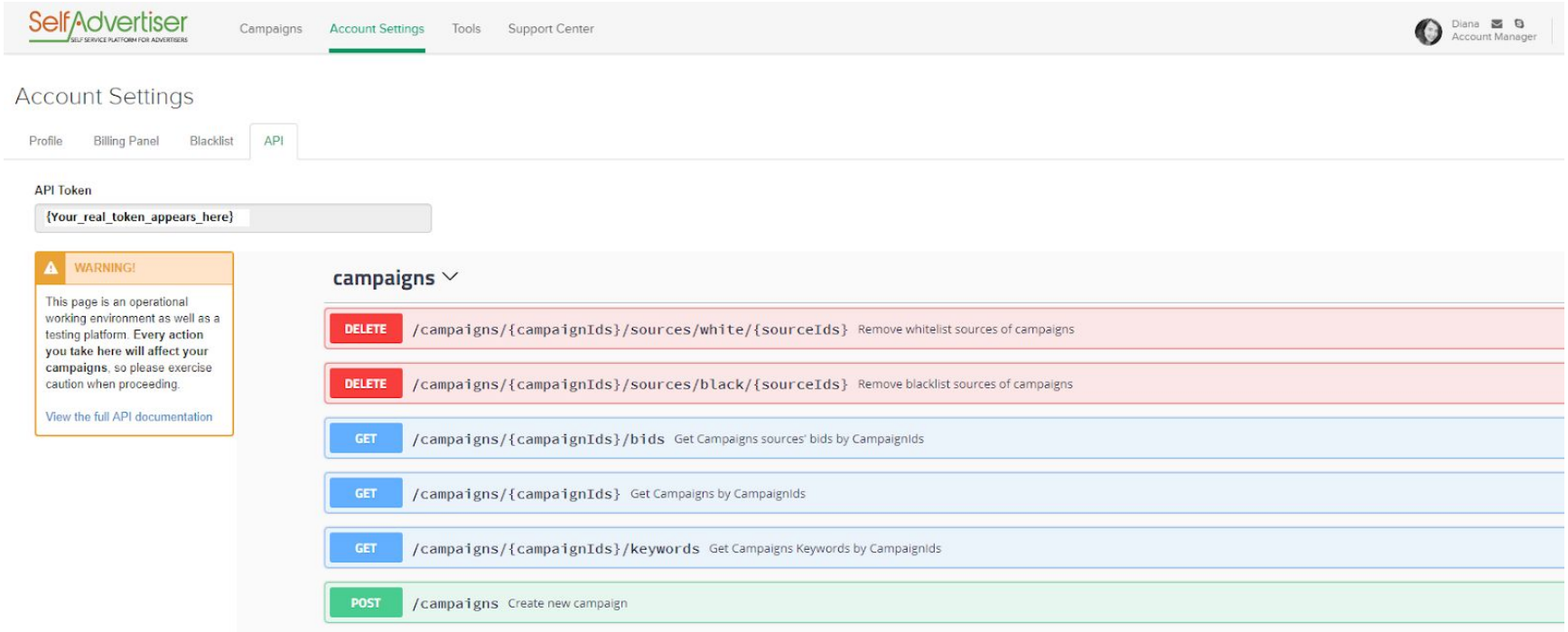
The token is needed when using session-less requests.
The token is NOT needed when using sessioned requests using the execution environment from within the webapp , it is automatically embedded using this approach, as described later.

The token string is predefined and unique for each advertiser, and available in a new tab in the SelfAdvertiser App.

To get the token for requests other outside the webapp (for the use of your own REST client):

Log into the application

Go to ***Account Settings*** tab -> ***API*** tab



Please notice the API token value - that value should be used with each API execution that is outside the execution environment.
Please recognize, on that same tab, embedded execution environment.

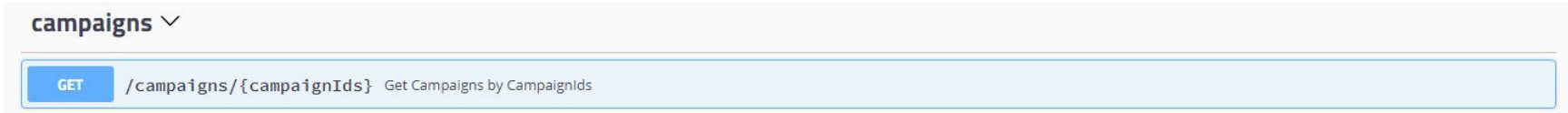
Execution Environment

The execution environment is a web interface where you can execute **real API requests live on your account.**
This environment serves both as a performing RESTful web client, and a documentation source.

Each line represents an API which is described along with its HTTP verb, URL entrypoint and short description.

Example:

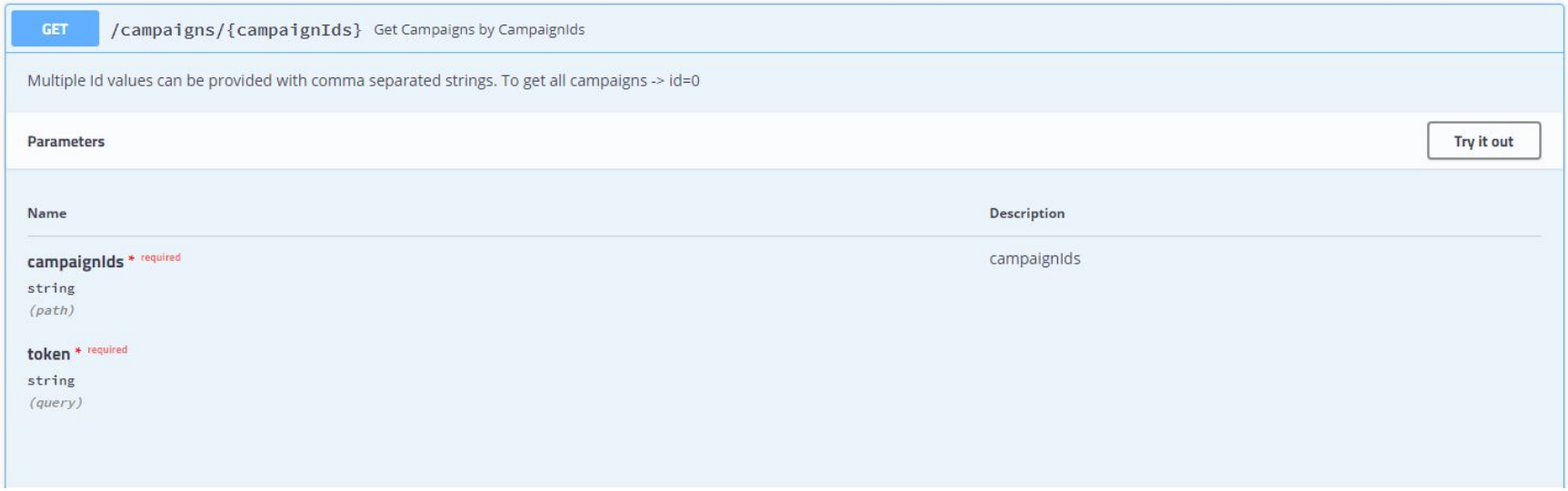
Under the ‘campaigns’ section, we can see the api:



This first level description tells us there's available API using the HTTP **GET** command, to the resource **/campaigns/{campaignIds}** which will return the wanted campaigns information, please notice the path parameter **{campaignIds} - which will be discussed later**.
If we click on that line, the description expands.
We can see there are 2 related parameters for this API: the ***campaignIds (this is a path parameter, please notice)***, and the ***token***, followed by the different responses returned by that API and the ability to switch the response type between json and xml.

campaignIds - this is a PATH parameter,
as described further more ("Multiple id values can be provided with comma separated strings to get all campaigns -> id=0"),
it represents the campaigns we want to get information about.
To get all the campaigns information, the value 0 should be used.

token - the token for the specific advertiser for authentication.



Responses Response content type: application/json

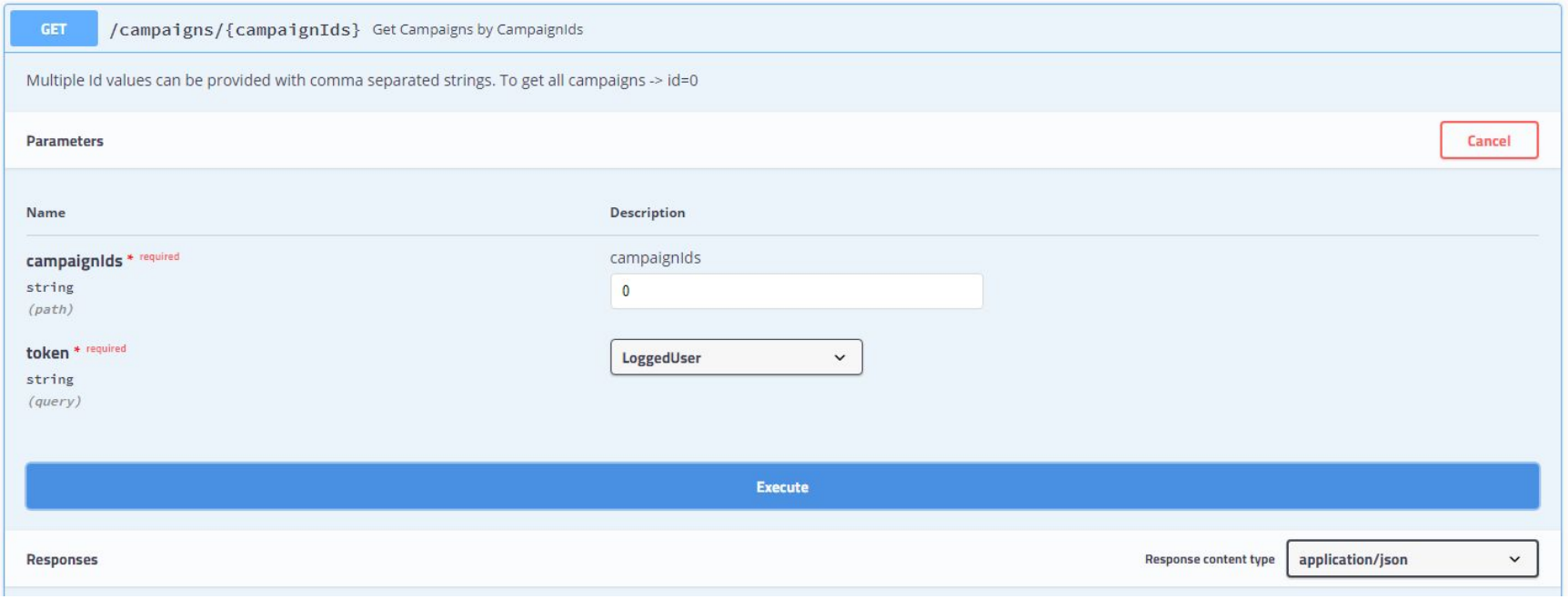
Code	Description
200	successful operation

Example Value | Model


```
{
  "returnStatusStruct": {
    "returnCode": 0,
    "returnMessage": "string",
    "requestId": "string",
    "requestUri": "string",
    "queryString": "string"
  },
  "entity": {},
  "meta": {
    "startRow": 0,
    "endRow": 0,
    "totalRows": 0,
    "fromDate": "string",
    "toDate": "string",
    "customInfo": "string"
  },
  "list": {
    "list": [
      {}
    ]
  }
}
```


400	Invalid params supplied
404	Campaign(s) not found
500	Internal server error
503	Service not available

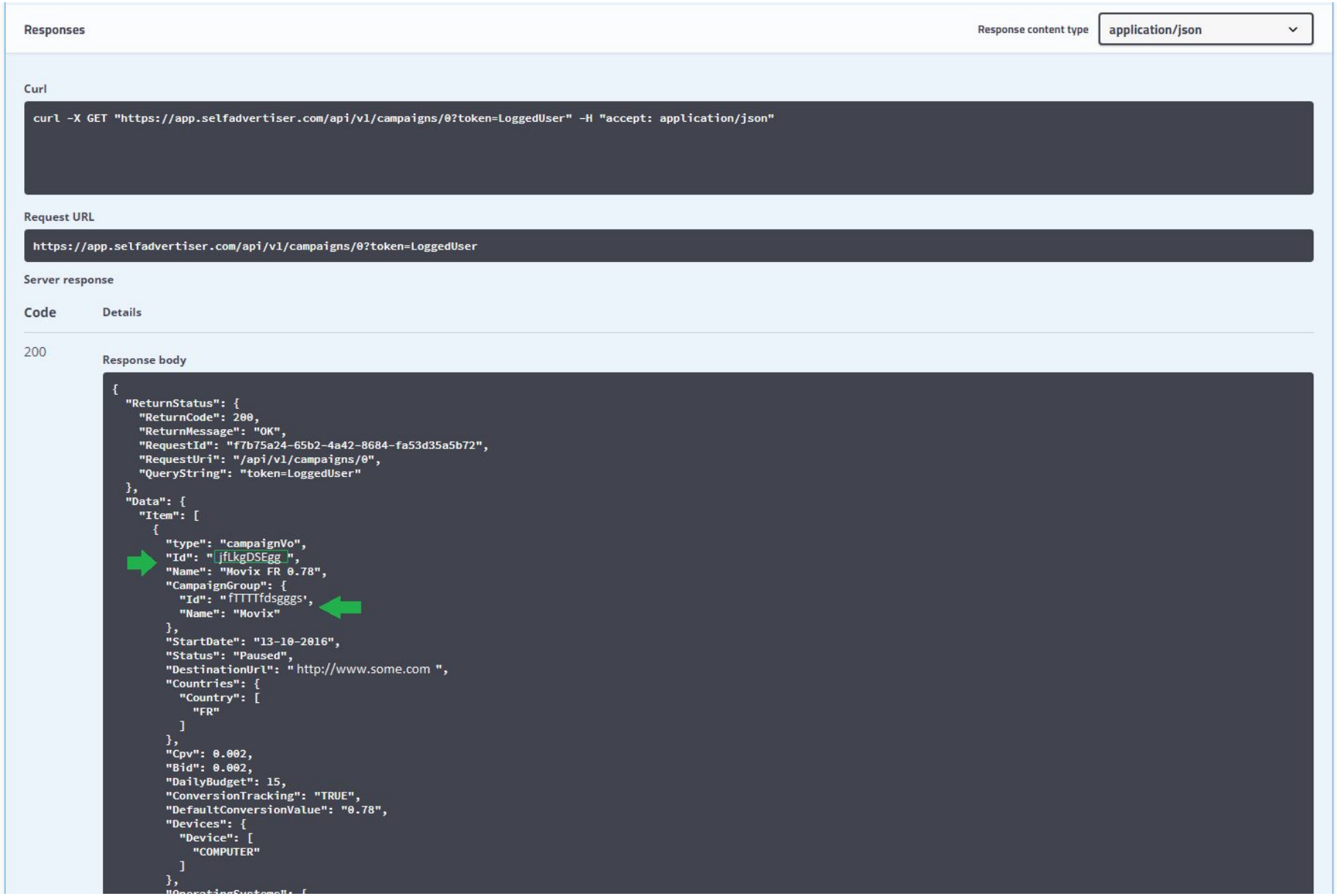
Up until now, we had the formal description of this specific get campaigns API.
Clicking on "**Try it out**" button will reveal the embedded and **fully live operational REST CLIENT**:



Please notice **campaignIds** parameter input field - you can use this web interface to input this parameter.
Remember, for this example "0" means all campaigns, other than this option , you can specify the specific campaign ids separated by comma.
Please notice **token** parameter is already embedded for this execution environment, and the real token is not needed to execute in this environment.

Please hit the "**Execute**" button with campaignIds set to 0, and wait for the result, the loading progress bar will appear upon execution:

Let’s take a look at an example response:



Responses

Response content type: application/json

Curl

```
curl -X GET "https://app.selfadvertiser.com/api/v1/campaigns/0?token=LoggedUser" -H "accept: application/json"
```

Request URL

```
https://app.selfadvertiser.com/api/v1/campaigns/0?token=LoggedUser
```

Server response

Code	Details
200	Response body


```
{
  "ReturnStatus": {
    "ReturnCode": 200,
    "ReturnMessage": "OK",
    "RequestId": "f7b75a24-65b2-4a42-8684-fa53d35a5b72",
    "RequestUri": "/api/v1/campaigns/0",
    "QueryString": "token=LoggedUser"
  },
  "Data": {
    "Item": [
      {
        "type": "campaignVo",
        "Id": " jfLkgDSEgg ",
        "Name": "Movix FR 0.78",
        "CampaignGroup": {
          "Id": "fTTTTfdsgggs",
          "Name": "Movix"
        },
        "StartDate": "13-10-2016",
        "Status": "Paused",
        "DestinationUrl": " http://www.some.com ",
        "Countries": {
          "Country": [
            "FR"
          ]
        },
        "Cpv": 0.002,
        "Bid": 0.002,
        "DailyBudget": 15,
        "ConversionTracking": "TRUE",
        "DefaultConversionValue": "0.78",
        "Devices": {
          "Device": [
            "COMPUTER"
          ]
        },
```

We can see in this demo example that we have received back **HTTP 200** response containing **JSON response** in the body. We can notice the name and correlating **id of the campaign** and the **campaign group name and id** which the campaign is related to.
Scrolling further down will show us all the remaining details for that specific campaign, and the other campaigns as well.
Please remember we wanted to get the information for all the campaigns by indicating "0" in the campaignIds input text.

If we wanted to get the information for this specific campaign, we needed to enter its exact id (instead of 0) in the campaignIds input text.

We can also see examples of the appropriate curl command and http command for this running:

```
curl -X GET "https://app.selfadvertiser.com/api/v1/campaigns/0?token=LoggedUser" -H "accept: application/json"
```

```
https://app.selfadvertiser.com/api/v1/campaigns/0?token=LoggedUser
```

Important note:

Executing curl OR the URL address alternatives, will work outside of this execution environment,
depending that the **token parameter is replaced with your real token,** described earlier**.**

Non-working example for a specific campaign information invocation, outside of the execution environment:

https://app.selfadvertiser.com/api/v1/campaigns/jlfkgDSEgg&token=65436534c3rrrd4844466b57eac601500

Existing API Appendix

As described earlier, the main entry point URL is https://app.selfadvertiser.com/api/vx/ where vx represents requested version of service.

Each request should be sent using an appropriate HTTP method.
According to RESTful API conventions, GET for resource retrieval, PUT for resource updating, POST for resource creating

To work with specific campaign(s) (info, stats or update), first get all campaigns ids by calling either GET /campaigns/ , or
GET /groups/ (holds both campaigns and campaign groups).

To work with specific group(s) (info, stats or update), first get all group ids by calling GET groups/ API.

The available methods are described below, along with their description and relevant parameters

(please notice next section "**Supported QueryString Parameters**"):

METHOD	DESCRIPTION	RELEVANT PARAMS
GET campaigns/{campaignIds}	All campaigns details no stats. option to filter by campaign status	**token** **campaignIds (in the path)** **status**
GET campaigns/{campaignIds}/keywords	List of keywords for specific campaign(s)	**token** **campaignIds (in the path)**
GET campaigns/{campaignIds}/bids	Bid per sources for specific campaign(s)	**token** **campaignIds (in the path)**
GET groups/{campaignGroupIds}	List all groups (optional campaign details - with(BASIC or FULL) / without (NONE))	**token** **campaignGroupIds (in path)** **details**
GET stats/campaigns/{campaignIds}	Campaigns stats. option to filter by campaign id, from/to date, min/max impressions, cost, or conversions	**token** **campaignIds (in the path)** **fromDate** **toDate** **minImpressions** **maxImpressions** **minCost** **maxCost** **minConversions** **maxConversions**
GET stats/groups/{campaignGroupIds}	All campaigns stats for group. option to filter by campaign group id, from/to date, min/max impressions, cost, or conversions	**token** **campaignGroupIds (in the path)** **fromDate** **toDate**

		minImpressions **maxImpressions** **minCost** **maxCost** **minConversions** **maxConversions**
GET stats/campaigns/{campaignids}/sources	Single campaign sources stats. option to filter by campaign id, from/to date, min/max impressions, cost, or conversions	**token** **campaignIds (in the path)** **fromDate** **toDate** **minImpressions** **maxImpressions** **minCost** **maxCost** **minConversions** **maxConversions**
GET stats/campaigns/{campaignids}/keywords	Single campaign keywords stats. option to filter by from/to date, min/max impressions, cost, or conversions	**token** **campaignIds (in the path)** **fromDate** **toDate** **minImpressions** **maxImpressions** **minCost** **maxCost** **minConversions** **maxConversions**
PUT campaigns/{campaignids}/action/{action}	Update status for single campaign, using action(various actions in **bold**) - followed by description of campaign state transition: **Resume** : PAUSED => ACTIVE	**token** **campaignIds (in the path)** **action (in the path)**

	REJECTED => NEW **Pause**: ACTIVE => PAUSED **Remove**: ? => DELETED	
PUT campaigns/{campaignids}/bid/{newbid}	Update bid for single campaign	**token** **campaignIds (in the path)** **newbid (in the path)**
PUT campaigns/{campaignids}/url/?url ={newurl}	Update url for single campaign	**token** **campaignIds (in the path)** **newurl (in the path)**
PUT /campaigns/{campaignIds}/sources/bids	Update bid per source for campaigns	**token** **campaignIds (in the path)** **source & bid in JSON (request body).** **Example:** **{** **"123456": "0.001",** **"456789": "0.002"** **}** **Please note the difference from create campaign scenario using 'data' request parameter , which accepts a different format than json.**
PUT /campaigns/{campaignIds}/daily-budget/{daily-budget}	Update campaigns daily budget	**token** **campaignIds (in the path)**

		daily-budget (in the path)
PUT /campaigns/optimized/{campaignIds}/core	Update optimized campaign core date: multiplier cpa_goal default_conversion_value	**token** **core data in JSON (request body).** **Example:** **{** **"multiplier": "1.0",** **"cpa_goal": "2.5",** **“default_conversion_value:”1.0”** **}**
DELETE campaigns/{campaignids}/sources/white/ {sourceIds}	Remove a source for specific campaigns from their whitelist of sources **(*1)**	**token** **campaignIds (in the path)** **sourceids (in the path)**
DELETE campaigns/{campaignids}/sources/black/ {sourceIds}	Remove a source for specific campaigns from their blicklist of sources	**token** **campaignIds (in the path)** **sourceids (in the path)**
DELETE /campaigns/{campaignIds}/keywords/active/{keywords}	Remove keywords from the campaigns' active keywords list	**token** **campaignIds (in the path)** **keywords (in the path)**
DELETE /campaigns/{campaignIds}/keywords/blocked/{keywords}	Remove keywords from the campaigns' blocked keywords list	**token** **campaignIds (in the path)** **keywords (in the path)**
POST /campaigns	Create new campaign. **(*2)**	**token** **data**
POST /campaigns/optimized	Create new optimized campaign **(*2)**	**token** **data**

POST /campaigns/{campaignIds}/keywords/active/{keywords}	Add keywords to campaigns' active keywords list	**token** **campaignIds (in the path)** **keywords (in the path)**
POST /campaigns/{campaignIds}/keywords/blocked/{keywords}	Add keywords to campaigns' blocked keywords list	**token** **campaignIds (in the path)** **keywords (in the path)**
POST /campaigns/{campaignIds}/sources/white/{sourceIds}	Add sources to campaigns' whitelist sources	**token** **campaignIds (in the path)** **sourceIds(in the path)**
POST /campaigns/{campaignIds}/sources/black/{sourceIds}	Add sources to campaigns' blacklist sources **(*1)**	**token** **campaignIds (in the path)** **sourceIds(in the path)**

(*1)

Adding a source to blacklist or removing a source from whitelist, will delete its specific bid (in case bid per source was defined).

(*2)

'data' parameter can be accepted in the request body as a json, in additional to the existing API.
Please review an example of a proper json structure as described below in '**Request Examples**' section.
Each field and matching values are explained in '**Supported QueryString Parameters**' section.
"404 - Campaign(s) not found" status is irrelevant for this API.

Create Campaign Using API

The API allows you to create campaigns without the need to go through the platform.

General Concept

In the API, each creation call creates only one campaign.

It's important to note that the creation API supports JSON format in which value pairs are separated by a comma {"key1":”value1”,”key2”:value2”}, a structure that is not compatible with that format will result in data error.

Column Description and Valid Values

Below is the list of attributes in the JSON structure. Fields marked in * are mandatory.

Field Name	Type
ID	A unique ID which is issued by the system, in order to uniquely identify the campaign. Leave this field empty, the system will create a new campaign id following to the creation call.
Group Name*	This is the campaign’s group name. You can create a new group, by writing in this field a group name that does not yet exist in the system.
Name* (A.K.A. Campaign name)	The name of the campaign.
Status+	This field should be empty.
Action+	This field should be empty.
Active Keywords	Current Keywords or Keywords you want to add. Each Keyword should be separated by a pipe (“|”) Example: casino|casino offer|online casino

Blocked Keywords	Current Keywords or Keywords you want to block. Each keyword should be separated by a pipe (“|”) Example: casino|casino offer|online casino
Daily Budget*	The daily budget per campaign, as a positive whole number in USD.
Daily Budget per Source	The daily budget you want to spend on a single source. The Daily Budget per Source should be greater than 10$ per day, or '-1' to disable this property.
CPV*	The bid per view/Impression, in USD.
Full page PPV*	Can be TRUE/FALSE
Domain Redirect*	Can be TRUE/FALSE (mutually exclusive with Full Page PPV)
Destination URL*	The URL of your Landing page. You can use the @@SOURCE@@ or @@CAMPAIGN-KEYWORD@@ tokens in your URLs, and we'll provide you the source (site) ID or the campaign Keyword that was matched to your landing page for easy tracking. If conversion tracking is enabled, the URL must also include the @@CLICK-ID@@ token.
Track Conversions*	Can be TRUE/FALSE. If you want to track conversions, select TRUE, alternatively select FALSE. If you are tracking conversion, make sure to implement the @@CLICK-ID@@ token in your destination URL.
Default Conversion Value	In case you wish to track conversions, please insert the value of conversion. If you are not tracking conversions, leave it blank.
Black List	Add the source IDs you wish to block. Each source should be separated by a pipe (“|”) Example: 4228595|6828797|7426531
White List	Add specific sources (Whitelist) you want the campaign to run on. Each source should be separated by a pipe (“|”) Example: 4228595|6828797|7426531
Bid per source	Assign a bid to a specific source ID. Each pair should be separated by a pipe (“|”). Example: 123:0.5|456:1.5

Countries	A two-letter country code (ISO Alpha 2) For multiple GEOs, use a pipe (“|”) as separator. To target all GEOs, you can either leave the field empty or type ALL. Example: MX|PL|SG|TR|ZA
Desktop	Can be TRUE/FALSE
Tablet	Can be TRUE/FALSE
Mobile	Can be TRUE/FALSE
Windows	Type ALL to target all Windows versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid versions: WINDOWS_10, WINDOWS_81, WINDOWS_RT, WINDOWS_8, WINDOWS_7, WINDOWS_VISTA, WINDOWS_2000, WINDOWS_XP, WINDOWS_10_MOBILE, WINDOWS_PHONE10, WINDOWS_PHONE8_1, WINDOWS_PHONE8, WINDOWS_MOBILE7, WINDOWS_PHONE, WINDOWS_MOBILE, WINDOWS_98, XBOX_OS For example: "ALL|!WINDOWS_10" or "WINDOWS_81+"
Mac	Type ALL to target all Mac versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid versions: MAC_OS_X10_14, MAC_OS_X10_13, MAC_OS_X10_12, MAC_OS_X10_11, MAC_OS_X10_10, MAC_OS_X10_09, MAC_OS_X10_08, MAC_OS_X10_07, MAC_OS_X10_06, MAC_OS_X10_05 For example: "ALL|!MAC_OS_X10_14" or "MAC_OS_X10_13+"
Linux	Type ALL to target all Linux versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid versions:

	UBUNTU
Android	Type ALL to target all Android versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: ANDROID9, ANDROID8, ANDROID7, ANDROID6, ANDROID5, ANDROID4, ANDROID3, ANDROID2, ANDROID1 For example: "ALL|!ANDROID9" or "ANDROID8+"
Chrome OS	Write ALL to select this option or leave it blank to detarget it.
iOS	Type ALL to target all iOS versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: iOS12, iOS11, iOS10, iOS9, iOS8, iOS7, iOS6, iOS5, iOS4, iOS3_IPHONE, MAC_OS_X_IPAD For example: "ALL|!iOS12" or "iOS11+"
Other OS	Write ALL to select this option or leave it blank to detarget it.
Apple WebKit	Type ALL to target all Apple WebKit versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: APPLE_WEB_KIT_6, APPLE_WEB_KIT_5, APPLE_WEB_KIT_4 For example: "ALL|!APPLE_WEB_KIT_6" or "APPLE_WEB_KIT_5+"

Chrome	Type ALL to target all Chrome versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: CHROME74, CHROME73, ..., CHROME8 For example: "ALL|!CHROME74" or "CHROME73+"
Microsoft Edge	Type ALL to target all Microsoft Edge versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: EDGE18, EDGE17, EDGE16, EDGE15, EDGE14, EDGE13, EDGE12 For example: "ALL|!EDGE18" or "EDGE17+"
Firefox	Type ALL to target all Firefox versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: FIREFOX66, FIREFOX65, ..., FIREFOX3, FIREFOX2, FIREFOX1_5 For example: "ALL|!FIREFOX66" or "FIREFOX65+"
Internet Explorer	Type ALL to target all Internet Explorer versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: IE11, IE10, IE9, IE8, IE7, IE6, IE5_5, IE5 For example: "ALL|!IE11" or "IE10+"

Opera	Type ALL to target all Opera versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: OPERA_MINI, OPERA59, OPERA58, ..., OPERA9 For example: "ALL|!OPERA59" or "OPERA58+"
Safari	Type ALL to target all Safari versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: BLACKBERRY10, MOBILE_SAFARI, SAFARI12, SAFARI11, ..., SAFARI4 For example: "ALL|!SAFARI12" or "SAFARI11+"
Samsung Internet	Type ALL to target all Samsung Internet versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: SAMSUNG_BROWSER9_0, SAMSUNG_BROWSER8_4, SAMSUNG_BROWSER8_2, SAMSUNG_BROWSER8_0, SAMSUNG_BROWSER7_4, SAMSUNG_BROWSER7_2, SAMSUNG_BROWSER7_1, SAMSUNG_BROWSER7_0, SAMSUNG_BROWSER6_4, SAMSUNG_BROWSER6_3, SAMSUNG_BROWSER6_2, SAMSUNG_BROWSER6_0, SAMSUNG_BROWSER5_4, SAMSUNG_BROWSER5_2, SAMSUNG_BROWSER5_0, SAMSUNG_BROWSER4_2, SAMSUNG_BROWSER4_0, SAMSUNG_BROWSER3_5, SAMSUNG_BROWSER3_4, SAMSUNG_BROWSER3_3, SAMSUNG_BROWSER3_2, SAMSUNG_BROWSER3_0, SAMSUNG_BROWSER2_1, SAMSUNG_BROWSER2_0, SAMSUNG_BROWSER1_1, SAMSUNG_BROWSER1_0 For example: "ALL|!SAMSUNG_BROWSER9_0" or "SAMSUNG_BROWSER8_4+"

UC Browser	Type ALL to target all UC Browser versions, or type a specific version. Values should be separated by a pipe ("|"). To detarget this option, leave this field blank. To target a specific versions and higher versions add '+' after the OS name. To detarget a specific option add '!' after the OS name. List of valid values: UCBROWSER12, UCBROWSER11, UCBROWSER10, UCBROWSER9, UCBROWSER8 For example: "ALL|!UCBROWSER12" or "UCBROWSER11+"
Frequency Capping	Control how often your ad will be displayed to the same visitor. A visitor is defined using a combination of IP address and user agent. Valid values are: 144 (1 Visit / 10 Minutes) 48 (1 Visit / 30 Minutes (Recommended-default value) 24 (1 Visit / 1 Hour) 12 (1 Visit / 2 Hours) 8 (1 Visit / 3 Hours) 6 (1 Visit / 4 Hours) 4 (1 Visit / 6 Hours) 3 (1 Visit / 8 Hours) 2 (1 Visit / 12 Hours) 1 (1 Visit / 24 Hours) 0 (Unlimited)
Traffic Distribution	Valid values can be EVENLY (Evenly throughout the day) or EAGER (As soon as possible)
After Approval	Valid values can be START or PAUSED

Supported QueryString Parameters

PARAMETER	DEFAULT	Description
token=<<token>>	-	Security token, will be used to identify the advertiser. Will be sent in every request. **Required**
format=<<XML|JSON>>	JSON	The return values format
fromDate=<<yyyy-MM-dd>>	NOW-7 DAYS	fromDate for stats - optional
toDate=<<yyyy-MM-dd>>	NOW	toDate for stats - optional
minImpressions=<<integer>>	0	minimum impressions for campaign, inclusive - optional
maxImpressions=<<integer>>	infinity	maximum impressions for campaign, inclusive - optional
minCost=<<number>>	0	minimum cost of campaign, inclusive - optional
maxCost=<<number>>	infinity	maximum cost of campaign, inclusive. must be bigger than 0, otherwise will be ignored - optional
minConversions=<<integer>>	0	minimum conversions for campaign, inclusive - optional
maxConversions=<<integer>>	infinity	maximum conversions for campaign, inclusive - optional
details=<<NONE|BASIC|FULL>>	BASIC	detail level - relevant to group info and group stats
data	no default value	A JSON formatted string , passed as a parameter in the query request, containing all the information for a campaign creation. Please notice the structure format: {“KEY1”:”VALUE1”,”KEY2”:”VALUE2”,.....} Each key-value pair defines the campaign attribute and its value,

		as defined in **"Column Description and Valid Values"** section. Please see example in "**Request Examples**".

Request Examples

For a customer with token 1111:

1. Get all campaigns in XML format –

/GET https://app.selfadvertiser.com/api/v1/campaigns/?token=111&format=xml

2. Get sources bids of one campaign (id= aaa) in JSON format –/GET
https://app.selfadvertiser.com/api/v1/campaigns/aaa/bids/?token=111&format=json (format can be omitted, JSON is default)

3. Update bid of two campaigns (id=aaa and id=bbb) to 0.3

/PUT https://app.selfadvertiser.com/api/v1/campaigns/aaa,bbb/bid/0.3?token=111

When using optional Ids (e.g: {campaignids}, {groupids}):

1. To get single campaign, use Id - e.g. to get campaign id *abc* details, use campaigns/abc
2. To get multiple campaigns, use comma separated ids – e.g. to get campaign ids *abc & def* details, use campaigns/abc, def
3. To get all campaigns, the ids section can be left empty. if further path needed use 0 rather than empty
    a. to get all campaigns details: campaigns/
    b. to get all campaigns' keywords: campaigns/0/keywords

4. Create a new campaign
(Copying this example from the PDF document might not work properly, please refer to

https://ssa-api.selfadvertiser.com/APICustomerdoc_create_campaign_json_example.txt):

/POST https://app.selfadvertiser.com/api/v1/campaigns?data={"Id":"","GroupName":"My First Group","Name":"My first Campaign","Status":"","Action":"","Active Keywords":"kw1|kw2","Blocked Keywords":"","Daily Budget":"25","Daily Budget per Source":"20","CPV":"0.9","Full page PPV":"","Domain Redirect":"TRUE","Destination URL":"http://www.api.com?id=@@CLICK-ID@@","Track Conversions":"TRUE","Default Conversion Value":"5","Black List":"","White List":"","Bid per source":"","Countries":"US|GB","Desktop":"TRUE","Tablet":"FALSE","mobile":"TRUE","Windows":"WINDOWS_7+","Mac":"ALL","Linux":"","Android":"","ChromeOS":"","IOS":"","Other OS":"","Chrome":"","Microsoft Edge":"ALL","Firefox":"","Internet Explorer":"","Opera":"","Safari":"","ucbrowser":"","apple webkit":"","samsung internet":"","Other Browsers":"","After Approval":"PAUSED","Frequency Capping":"48","Traffic Distribution":"EVENLY"}&token=111

Please pay extra attention to the 'data' variable , which is in fact a json structure passed.

An accessible text example of data json structure can be found here:
https://ssa-api.selfadvertiser.com/APICustomerdoc_create_campaign_json_example.txt

5. Create a new optimized campaign
(Copying this example from the PDF document might not work properly, please refer to
https://ssa-api.selfadvertiser.com/APICustomerdoc_create_optimized_campaign_json_example.txt):

/POST https://app.selfadvertiser.com/api/v1/campaigns/optimized?data={"Id":"","GroupName":"DCPM campaigns","Name":"my first DCPM campaign","Status":"","Action":"","Active Keywords":"kw1|kw2","Blocked Keywords":"","Daily Budget":"25","Daily Budget per Source":"-1","Full page PPV":"","Domain Redirect":"TRUE","Destination URL":"http://www.api.com?id=@@CLICK-ID@@","Track Conversions":"TRUE","Default Conversion Value":"5","Black List":"","White List":"","Bid per source":"","Countries":"US","Desktop":"TRUE","Tablet":"FALSE","mobile":"TRUE","Windows":"WINDOWS_7+","Mac":"ALL","Linux":"","Android":"","ChromeOS":"","IOS":"","Other OS":"","Chrome":"","Microsoft Edge":"ALL","samsung internet":"","Firefox":"","Internet Explorer":"","Opera":"","Safari":"","ucbrowser":"","apple webkit":"","Other Browsers":"","After Approval":"PAUSED","Frequency Capping":"48","Traffic Distribution":"EVENLY","MULTIPLIER":"3.0","CORE BID":"1.1","CPA GOAL":"1.2"}&token=111

Please pay extra attention to the 'data' variable , which is in fact a json structure passed.

An accessible text example of data json structure can be found here:
https://ssa-api.selfadvertiser.com/APICustomerdoc_create_optimized_campaign_json_example.txt

Responses

Each response will consist of three parts as described below:

{ *ReturnStatus:*
{ ReturnCode:<<###>>,
ReturnMessage:<<XXXXXXXX>>,
RequestId:<<####>>,
RequestUri:<<XXXXXXX>>,
QueryString:{XXXXXXX}
}
Data:
{ [....]
return data always as array, empty if error or not found
}
Meta:
{ paging info & special messages (e.g – unable to update reasons)
}
}

Response Codes

Code	Description	Comments
200	Success	
403	No Token Supplied	
403	Invalid Token	

404	Resource not found	e.g.: requesting for statistics of a campaign that does not exist (wrong campaign id)
500	Internal server error	
503	Service Unavailable	API requests not available