Server API
Jamespot.pro

V2.0

Documentation version 2.0.27

Table of contents

Introduction

The Jamespot.pro platform provides organisations with a private social network, where they can share and comment on information relating to their interests, creating and then broadening a community of people around these subjects.

This document describes the Jamespot.pro API. This API allows main read/write operations to be executed. It offers a selection of functions classified by general types of object:

Object	Functions
User	create() get() - getByField() contacts() – spots() update() list() delete() activity() - notify()
Spot	create() get()- getByField() update() list() delete() setUserLevel() - getUserLevel() members() - articles() activity() - notify()
Article	create() get()- getByField() update() list() delete() uploadFile(), deleteFile() comments(), addComment(), deleteComment()
Link	create(), list(), delete()
Global	translation() profiles() version()

Principles

All API functions can be called up through a single point of access:

`https://[platform].jamespot.pro/api/api.php?<paramList>`

The list of parameters <paramList> defines the characteristics of the API call.

The following table summarises the potential parameters in this list.

The parameters and values are case sensitive.

1.1. Mandatory parameters

Object	Functions
object	**Description**: Object which concerns the API function call **Alias**: o **Examples of values:** spot, article, user, spot_user
function	**Description**: Requested function for the object **Alias**: f **Examples of values:** get, create, delete, update
version	**Description**: Version of the API used **Alias**: v **Example**: 2.0 **Value by default:** 1.0 **WARNING**: to use version 2.0 of the API, this parameter must be provided.

To authenticate API Calls, two options are available :

- Add http headers to the call :
  X-Com-Jamespot-Module-Name : Application ID
  X-Com-Jamespot-Module-Key : Key value associated with the application

- Add parameters to the call

Object	Functions
module	**Description**: Application ID declared on the platform. **Alias**: m **Examples of value**: EXT-applicationName

Object	Functions
date	**Description**: The call's validity date. YYYY-MM-DDThh:mm:ssTZD **Alias**: d **Examples of value**: 2010-10-26T11:00:00Z OR 2010-10-26T12:00:00+01:00 Php generation: date('Y-m-d\TH:i:sP);
key	**Description**: The call's security key. The value is obtained by the following function: md5(applicationID '-' date '-' cléModule) **Alias**: k **Examples of value**: For md5(EXT-applicationName-2009-11-05T13:15:30-cledesecurite), k= 3d210275380b3431b6accfaeadbb8612

1.2. Call examples

The parameters can be provided in GET or in POST.

Example with GET:

To obtain the json description of the spot with the idspot 3, for the 'foo' platform, call up the URL:

https://foo.jamespot.pro/api/api.php?&k= d41d8cd98f00b204e9800998ecf8427e&d=2010-07-07T00:00:00&a=EXT-application&o=spot&f=get&idspot=3

1.3. Return format

The JSON return value is always an object comprising three fields: RC (Returned Code), STATS (Statistics) and VAL (Value).

RC is an object containing:

- The entire return code: During a correct execution, the return code is 0. In other cases, it is anything but 0.
- The error message. This message is empty for a correct execution.

STATS is an object containing:

- TIME The execution time
- And other optional parameters which depend on the API function

VAL is the object returned by the function's execution.

Examples

Correct return	{ "RC":{"CODE":0,"MSG":""}, "STATS":{"TIME":74}, "VAL":{"idSpot":"1234",[...]}}
Return error	{ "RC":{"CODE":10004,"MSG":"Missing parameter:IdSpot"}, "STATS":{"TIME":0}, "VAL":"" }

API reference

API is organised by objects which concern the function to be executed.

Possible objects are:

- user
- spot
- article
- global

For each object there is a description of the available functions, their input parameters, the associated processing and returned parameter.

1.4. User

1.4.1. Get

Parameter	Value
object	user
function	get
Optional parameters	
idUser	<idUser> Alias: idUser Examples of value 15
email	<email> Description: User email (used to connect to their account in the user interface). Example of value: username@live.com

At least one of the idUser/email parameters must be provided.
This function returns information on the user whose email or ID is used as a parameter.

For example, we can have:

```
{ 'RC' : '[…]', 'STATS' : '[…]',
        'VAL' : {idUser: 1.
        email: 'user1@mail.com', …
        user name: 'user1'
        img: 'http://imgUser1.png'
}
}
```

1.4.2. GetByField

Parameter	Value
object	user
function	getByField
name	Name of the field which concerns the request.
value	Value of the requested field Example of value: username@live.com

This function returns information on the user whose 'name' field is the 'value' value. If several users can fulfil this constraint, the user with the highest idUser is returned.

1.4.3. Create

Parameter	Value
object	user
function	create
Mail	<email> Email of the user to be created
Password	<pwd> Password of the user to be created
Image	<image> Attached image file
_image_base64	<_image_base64> base64 encoded Image file
User name	<user name> User name of the user to be created
Language	<language> User's language, TLD format in two letters Examples of values: en for English, fr for French
Country	<country> The user's country Examples of value: en, fr, es
Role	<role> User's role Examples of value: Administrator, User, External
Optional parameters	
CSV import fields	All of the usable fields in the CSV import can be used with the same column names, including *Profile,* or *property_xxx to provide additional user properties*.
activity	<Boolean> Generate the creation activity on the wall
welcomeMail	<Boolean> Sends an email to the user with this account information
connectUsers	<Boolean> Add all users as contacts
property_xxx	<property value> Add the property "xxx" with the associated value to the created user.

Communities	<idCommunity> Add a user to a list of communities Ex: 8,10,3

This function creates a new user.

If the user name or email already exists, the API will return an error.

The return is equivalent to user.get()

1.4.4. Update

Parameter	Value
object	user
function	update
Mail	<email> User email to be modified
Optional parameters	
CSV import fields	All of the fields in the CSV import can be used with the same column names.

The update only provides the parameters to be modified.

The return is equivalent to user.get().

1.4.5. List

Parameter	Value
object	user
function	list
Optional parameters	
name	<name> Example of value: email
id	<id> User id from which the limitation applies. The returned user ids will be strictly superior. Example of value: 15
types[]	<array> List of user types to be returned. Example: types[]=user&types[]=user1
limit	<limit> Number of elements to be returned (default 50)

Returns a list of user ids and fields defined by the argument "name" if it is provided.

Example:

{ 'RC' : '[…]', 'STATS' : '[…]',

```
        'VAL': [
          {"idUser":"1","user name":"user1"},
          {"idUser":"2","user name":"user2"}
          ]
}
```


1.4.6. Delete

Parameter	Value
object	user
function	delete
Optional parameters	
idUser **or** **Mail**	<idUser>

Deletes a user and their Spots, articles and comments.

Only returns the standard error code, with code = 0 when the execution is correct.

1.4.7. Activity

Parameter	Value
object	user
function	activity
idUser	<idUser>
wording **or** **act_resource**	<wording> direct activity chains <act_resource> call for a resourceKey in the user's language
Optional parameters	
json_params	<json> Parameters to be inserted in the resource
act_url	url of the return link to the activity's sender page.
act_title	Title of the link

Adds an activity linked to the user and distributed to their followers.

1.4.8. Notify

Parameter	Value
object	user
function	notify
idUser	<idUser> sender
idUserTo	<idUser> recipient
Resources	<json> Json containing 3 entries (value = resourceKey): -subject => email subject -content => email content -contentDelayed => email content delayed

Sends an email to the user.

The contents are in HTML format.

Once user alerts are configured, the value 'content' or 'contentDelayed' will be used:

- If the user has configured a real-time alert, the 'content' value is used
- If the user has configured a daily or weekly delayed alert, the 'contentDelayed' is used.

1.4.9. Contacts

Parameter	Value
object	user
function	contacts
idUser	<idUser>
Optionnels	
limit	<limit> Number of contacts per page. Default to 50
page	<page> Page to be returned

Sends the paginated contact list.

1.4.10. Spots

Parameter	Value
object	user
function	spots
idUser	<idUser>
Optionnels	
limit	<limit> Number of spots per page. Default to 50
page	<page> Page to be returned

Returns a list of users' spot, and for each spot, the membership level of this user.

1.5. Spot

1.5.1. Get

Parameter	Value
object	spot
function	get
idSpot	(complete) <idSpot> Spot ID.

Returns the description of a spot: IdSpot, Title, Edito, Image, Skin, Public.

1.5.2. GetByField

Parameter	Value
object	spot
function	getByField
name	Name of the field which concerns the request.
value	Value of the requested field Example of value: username@live.com

This function returns information on the Spot whose 'name' field is the 'value' value. If several Spots can fulfil this constraint, the Spot with the highest idSpot is returned.

1.5.3. Create

Parameter	Value
object	spot
function	create
idUser	<idUser>: The Spot creator
category	<category> Spot category. If the platform does not use categories, the value 1 must be used.
title	<Spot title>
Optional parameters	
public	Boolean (0, 1) By default: 1
edito	<Editorial text>
lang	<language> (in 2 characters) The user's language is used by default.

Creates a new Spot in the category, with the given title and the default skin configuration. The 'user' is the Spot creator.

Returns an identical object to spot.get with getArticles=0 and getMembers=0.

1.5.4. Update

Parameter	Value
object	spot
function	update
idSpot	<idSpot>
Optional parameters	
category	<category>
title	<Spot title>
public	Boolean (0, 1) By default: 1
idUser	<idUser>
edito	<Editorial text>

Updates the Spot.

Returns an identical object to spot.get with getArticles=0 and getMembers=0.

1.5.5. List

Parameter	Value
object	spot
function	list
Optional parameters	
name	<name> Example of value: Description
id	<id> Description: Spot id from which the limitation applies. (The returned ids are strictly greater) Examples of value 15
types[]	<array> List of the spot types to be returned Example: types[]=spot&types[]=thinkTank
limit	<limit> Number of elements to be returned (default 50)

Returns a list of Spot ids and fields defined by the argument "name" if it is provided.

See return example for the user list method.

1.5.6. SetUserLevel

Parameter	Value
object	spot
function	setUserLevel
idSpot	<idSpot>
idUser	<idUser>
Level	<Level> "Level" is an integer : CREATOR=0; ADMINISTRATOR=1; CONTRIBUTOR=2; SUBSCRIBER=3; VIEWER=4; VISITOR=5;

Adds a user to the Spot.

Only returns the standard error code, with code = 0 when the execution is correct.

1.5.7. GetUserLevel

Parameter	Value
object	spot
function	getUserLevel
idSpot	<idSpot>
idUser	<idUser>

Returns the role of a user in a Spot.

1.5.8. Members

Parameter	Value
object	spot
function	members
idSpot	<idSpot>

Returns the list of users who are members of a Spot, with their role.

1.5.9. Articles

Parameter	Value
Object	spot
function	articles
idSpot	<idSpot>
Optional parameters	
limit	<limit> Number of elements to be returned (default 50)
id	<id> Article id from which the limitation applies. The returned article ids will be strictly superior. Example of value: 15

Returns the list of articles (all types) for a Spot, in order of the idArticle. The value of the dateOrder is used to classify the articles in the Spot.

1.5.10. Delete

Parameter	Value
object	spot
function	delete
idSpot	<idSpot>

Deletes a Spot.

Only returns the standard error code, with code = 0 when the execution is correct.

1.5.11. Activity

Parameter	Value
object	spot
function	activity
idUser	<idUser>
idSpot	<idSpot>
wording **or** **act_resource**	<wording> direct activity chains <act_resource> call for a resourceKey in the user's language
Optional parameters	
json_params	<json> Parameters to be inserted in the resource
act_url	url of the link to the sender
act_title	Title of the link

Adds an activity associated with the Spot and distributed according to the logic of the Spot.

1.5.12. Notify

Parameter	Value
object	spot
function	notify
idUser	<idUser>
idSpot	<idSpot>
ressources	<json> Json containing 3 entries (value = resourceKey): -subject => email subject -content => email content -contentDelayed => delayed email content

Sends an email to all members of a Spot.

1.6. Article

1.6.1. Get

Parameter	Value
object	article
function	get
idArticle	complete <idarticle>

Returns all article fields and custom attributes, as well as the attached files.

1.6.2. GetByField

Parameter	Value
object	article
function	getByField
name	Name of the field which concerns the request.
value	Value of the requested field Example of value: username@live.com
type	Value's type of the requested field. *name*, *bigint*, *tinyint*, *text*, *longtext*, *float*, *date* *timestamp*

This function returns information on the article whose 'name' field is the 'value' value. If several articles can fulfil this constraint, the article with the highest idArticle is returned.

1.6.3. Create

Parameter	Value
object	article
function	create
idUser or **email**	<idUser> or User's <email>
publishTo	<uris> Example : spot/1, spot/2, user/3
type	<Type of article> "mpArticle" is the default type.
title	<Article title>
description	<article content>
Optional parameters	
url	<url>
edito	<article editorial>
image	<string> : « url »
UrlImage	<img_url>
tags	Article <tags> separated by commas
Sendalert	<string> "on" Sends an email to all members of the Spot
xxx	Where 'xxx' is the name of the supplementary attribute, depending on the article type.

Adds an article to the requested Spot. The User becomes the author. Returns the equivalent of Article.get()

1.6.4. Update

Parameter	Value
object	article
function	update
idArticle	complete <idarticle>.
Optional parameters	
title	<Article title>
text	<article content>
idUser	<idUser>
idSpot	<idSpot>
url	<url>
tags	<tags>
xxx	Where 'xxx' is the name of the supplementary attribute, depending on the type of article.

Modifies the fields provided for the requested article. At least one of the Title, Content, IdSpot, IdUser, Url or Img fields must be provided.

Returns the equivalent of article.get().

1.6.5. List

Parameter	Value
object	article
function	list
Optional parameters	
id	<id> Description: Article id from which the limitation applies. (The returned ids are strictly greater) Examples of value 15
limit	<limit> Number of elements to be returned (default 50)
idUser	<idUser> Articles author
idSpot	<idSpot> Spot containing the article
type	<type> Articles type.
itemFormat	<format> Format of articles returned in the list. Possibles values ; 'header', 'article' Defaults to 'header'

Returns a list of articles formatted according to itemFormat, fullfilling the conditions provided.

1.6.6. Delete

Parameter	Value
object	article
function	delete
idArticle	<idarticle>

Deletes the requested article.

Only returns the standard error code, with code = 0 when the execution is correct.

1.6.7. UploadFile

Parameter	Value
object	article
function	uploadFile
idArticle	<idarticle>
upload	<file>

Adds an attached file to the idArticle article.
Returns the equivalent of an Article.get(idArticle) after having attached the files sent.

The call must follow the "multipart/form-data" call conventions
The screenshot below shows an example of a call trace

1.6.8. SetImage

Parameter	Value
object	article
function	setImage
idArticle	<idarticle>
image	<file>

Associate an image to this article.

1.6.9. DeleteFile

Parameter	Value
object	article
function	deleteFile
idFile	<idafile>

Deletes the file idFile.

1.6.10. Comments

Parameter	Value
Object	article
function	comments
idArticle	<IdArticle>

Returns the article comments list.

1.6.11. AddComment

Parameter	Value
Object	article
function	addComment
idUser or **mail**	<idUser> or user's <mail>
idArticle	<IdArticle>
text	Comment text to be added

Adds a comment for the article, for this user. Returns a comment structure.

1.6.12. DeleteComment

Parameter	Value
object	Article
function	deleteComment
idComment	Comment id to be deleted

Removes this comment.

1.7. Link

1.7.1. Create

Parameter	Value
object	link
function	create
type	<link type>
srcUri	Link's source Uri
targetUri	Link's target Uri
Optionnal	
value	Text value of associated to the link

Adds a typed link, from source to target. Uris used may reference contents, users or spots.

1.7.2. List

Parameter	Value
object	link
function	list
Optionnels	
type	<link type>
srcUri	Link's source Uri
targetUri	Link's target Uri

Returns the list of filtered links following the constraints.

One of Source or target URI must be provided.

1.7.3. Delete

Parameter	Value
object	link
function	delete
type	<link type>
idLink	Link identifier

Deletes this link.

1.8. Global

1.8.1. ActivityStream

Parameter	Value
object	global
function	activityStream
startId	<id> first Id activity to be returned.
limit	Number of activities to be returned

Returns the activityStream starting from startId, up to 'limit' activities.

1.8.2. Translation

Parameter	Value
Object	global
Function	translation
Lang	<language> Example of values: en, fr, pl
stringArray	<json> Json resourceKey table Example of values: ["GLOBAL_Spot","GLOBAL_Search"]

Returns the translations of the requested chains on the platform.

1.8.3. Profiles

Parameter	Value
Object	global
Function	profiles

Returns the list of profiles available on the platform. Useful for creating users.

1.8.4. Version

Parameter	Value
Object	global
function	version

Returns the API version used.
Useful for testing your connection and your settings.

1.8.5. UserSynchro

Parameter	Value
object	global
function	userSynchro
publishTo	<uri> Example of value : spot/5
title	<articleTitle>
idUser	<idUser> Author of the batch request
upload	<file> CSV file containing users to be synchronized. For details, see article->upload()

Returns a synchronization article, with its dentifier, and,containing the log of the request execution.

1.8.6. Search

Parameter	Valeur
object	global
function	search
adminsearch	<true | false> optional. Security context*
Mail | idUser	< email address | identifier > optional. Security context*
query	< requête solr > Required parameter

Return Solr query result according to configured settings. More at : http://lucene.apache.org/solr/.

1.9. Taxonomy

1.9.1. Get

Parameter	Value
object	taxonomy
function	get
uri	<uri> Taxonomy identifier

Return taxonomy description : uri, type, title, dateCreation, mandatory.

1.9.2. Create

Parameter	Value
object	taxonomy
function	create
idUser	<idUser>
title	<taxonomy title>
type	<type> taxonomy type : taxonomyOpen taxonomyClose taxonomySkos
mandatory	<0 || 1> Is taxonomy mandatory

Add a new taxonomy.

1.9.3. addConcept

Parameter	Value
object	taxonomy
function	addConcept
idUser	<idUser>
uri	<uri> Taxonomy identifier
title	< taxonomy title>

Add a new concept for existing taxonomy .

1.9.4. Import

Parameter	Value
object	taxonomy
function	import
idUser	<idUser>
uri	<uri> Taxonomy identifier
file	<file> File .rdf which include the tree to transfer. See Article.UploadFile for further details.

1.10. Community

1.10.1. Get

Parameter	Value
object	community
function	get
idCommunity	<idCommunity> Exemple de valeur 15

Return data from specified community : 'id', 'title', 'description', 'status'.

1.10.2. List

Parameter	Value
object	community
function	list
[technical]	<0|1|null>

Return a list of community depending of the optional parameter ‘technical’ : all, technical or not technical.