Onside

API Documentation

Version 1.0

The API has been designed to be accessed over the internet using HTTP and has its calls organized similarly to REST-based API systems.

The API is designed around a number of objects known as services. Each service supports a number of specific calls that perform specific functions.

In order for the API’s backend database to be fully functioning create/read/update/delete (CRUD) calls are available for every service.

Although the API follows closely the commonly known REST design there are a number of differences that need noting. The API don’t support any PUT requests for creating objects instead the URI called for POST requests is used to differentiate between creating and updating objects.

The following HTTP request methods are supported.

GET – Get a specific / list of objects

POST – create / update specific object

DELETE – delete specific object

In order to maintain security an additional header is require with every call.

OnsideAuthorization: XXX

TOKEN/SESSION – depends on javascript crypt libraries

All responses from the API will be JSON encoded and will be formatted in the same way, this way parsing all responses should be handled in the same way.

Every response object will contain 2 common variables:

service – the service that has been called

count – the number of objects in the resultset

Additional to these 2 optional arrays will be returned:

resultset – A list of all service objects found

errorset – A list of all errors

Example successful “channel” call response made by making a GET request to the “/channel”

***{***

***"service":"Channel",***

***"count":2,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test channel 1",***

***"description":"this is a test channel 1",***

***"sport":"football",***

***"type":"team",***

***"level":"1",***

***"geolat":null,***

***"geolng":null***

***}***

***},***

***{***

***"result":***

***{***

***"id":"2",***

***"name":"test channel 2",***

***"description":"test channel 2",***

***"sport":"tennis",***

***"type":"individual",***

***"level":"1",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

**Error Handling**

Errors are reported back using a combination of HTTP status codes and the “errorset” array in responses.

200 Ok

201 Created

202 Accepted

400 Bad Request

401 Unauthorized

403 Forbidden

404 Not Found

405 Method Not Allowed

500 Internal Server Error

501 Not Implemented

503 Service Unavailable

The HTTP status code should always be checked before the response as not all errors will return a response body.

All errors returned in the “errorset” array in the body of the response can support variable text. Variable text will be quoted using single quotes.

These errors have grouped by connection and client errors.

**Connection Errors**

101 Missing required header

102 Unknown service '?'

103 Invalid action '?' for service '?'

104 Service '?' not implemented

105 Client login required for this call

**Client Errors**

201 Missing required field '?'

202 Invalid field '?', accepted format ‘?’

203 Record failed to be created

204 Record failed to be updated

205 Record failed to be deleted

Article – Service Calls

Each channel contains the following fields of information:

* **id** – Numeric ID that uniquely identifies this object
* **title** – Title of the object can hold up to 100 characters
* **images** – Image(s)
* **videos** – Video(s)
* **author** – Author of the article can hold up to 100 characters
* **source** – The original source can hold up to 100 characters
* **extended** – Extended information of article
* **publish** – The original publication date / time
* **keywords** – A comma separated list of keywords used by search

Currently all calls return the full objects.

To read data from this service there are 2 calls that can be made.

Method: GET

URL: /article

Response code: 200

Returns a list of articles no filters are currently accepted

Example response:

***{***

***"service":"Article",***

***"count":2,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"title":"test article 1",***

***"source":"this is a test article 1",***

***"publish":"2011-11-06 14:18:07"***

***}***

***},***

***{***

***"result":***

***{***

***"id":"2",***

***"title":"test article 2",***

***"source":"this is a test article 2",***

***"publish":"1975-04-12 15:00:02"***

***}***

***}***

***]***

***}***

Method: GET

URL: /article/{ID}

Response code: 200

Returns the full details for article identified by its {ID}

Example response:

***{***

***"service":"Article",***

***"count":2,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"title":"test article 1",***

***"source":"this is a test article 1",***

***"publish":"2011-11-06 14:18:07"***

***}***

***}***

***]***

***}***

To alter data using this service there are 5 calls that can be made.

Method: POST

URL: /article

Response code: 201

Create a new article object and return a list of objects containing a single object that has been created

Example response:

***{***

***"service":"Article",***

***"count":2,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"title":"test article 1",***

***"source":"this is a test article 1",***

***"publish":"2011-11-06 14:18:07"***

***}***

***}***

***]***

***}***

Method: POST

URL: /article/{ID}

Response code: 200

Update details for article object identified by its {ID} and return a list of objects containing a single object that has been created

Example response:

***{***

***"service":"Article",***

***"count":2,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"title":"test article 1",***

***"source":"this is a test article 1",***

***"publish":"2011-11-06 14:18:07"***

***}***

***}***

***]***

***}***

Method: DELETE

URL: /article/{ID}

Response code:

Delete article object identified by its {ID} and returns an empty list of objects

Channel – Service Calls

Each channel object contains the following fields of information:

* **id** – Numeric ID that uniquely identifies this object
* **name** – Name of the object can hold up to 100 characters
* **description** – Full description can hold any amount of data
* **sport** – The sport this object is about can hold up to 50 characters
* **type** – The type this object is can hold up to 50 characters
* **level** – The level this object is can hold a single integer (0-9)
* **keywords** – A comma separated list of keywords used by search
* **geolat** – The decimal representation of geo-location latitude
* **geolng** – The decimal representation of geo-location longitude

Currently all calls return the full objects.

To read data from this service there are 2 calls that can be made.

Method: GET

URL: /channel

Response code: 200

Returns a list of channels accepted filters (user/?public/private?)

Example response:

***{***

***"service":"Channel",***

***"count":2,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test channel 1",***

***"description":"this is a test channel 1",***

***"sport":"football",***

***"type":"team",***

***"level":"1",***

***"geolat":null,***

***"geolng":null***

***}***

***},***

***{***

***"result":***

***{***

***"id":"2",***

***"name":"test channel 2",***

***"description":"test channel 2",***

***"sport":"tennis",***

***"type":"individual",***

***"level":"1",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

Method: GET

URL: /channel/{ID}

Response code: 200

Returns the full details for channel identified by its {ID}

Example response:

***{***

***"service":"Channel",***

***"count":1,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test channel 1",***

***"description":"this is a test channel 1",***

***"sport":"football",***

***"type":"team",***

***"level":"1",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

To alter data using this service there are 5 calls that can be made.

Method: POST

URL: /channel

Response code: 201

Create a new channel object and return a list of objects containing a single object that has been created

Example response:

***{***

***"service":"Channel",***

***"count":1,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test channel 1",***

***"description":"this is a test channel 1",***

***"sport":"football",***

***"type":"team",***

***"level":"1",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

Method: POST

URL: /channel/{ID}

Response code: 200

Update details for channel object identified by its {ID} and return a list of objects containing a single object that has been created

Example response:

***{***

***"service":"Channel",***

***"count":1,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test channel 1",***

***"description":"this is a test channel 1",***

***"sport":"football",***

***"type":"team",***

***"level":"1",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

Method: DELETE

URL: /channel/{ID}

Response code:

Delete channel object identified by its {ID} and returns an empty list of objects

This service also supports additional calls shown below.

Method: POST

URL: /channel/follow

Response code:

Assign a user to follow a channel, required parameters (user ID / channel ID)

Method: POST

URL: /channel/unfollow

Response code:

Unassign a user from following a channel, required parameters (user ID / channel ID)

Discussion – Service Calls

TBD

Event - Service Calls

Each event object contains the following fields of information:

* **id** – Numeric ID that uniquely identifies this object
* **name** – Name of the object can hold up to 100 characters
* **sport** – The sport this object is about can hold up to 50 characters
* **type** – The type this object is can hold up to 50 characters
* **keywords** – A comma separated list of keywords used by search
* **participants** – A list of participant objects that are sport specific
* **time** – An object that is sport specific
* **parent** –
* **location** –
* **geolat** – The decimal representation of geo-location latitude
* **geolng** – The decimal representation of geo-location longitude

Currently all calls return the full objects.

To read data from this service there are 2 calls that can be made.

Method: GET

URL: /event

Response code: 200

Returns a list of event accepted filters (channel/discussion/user)

Example response:

***{***

***"service":"Event",***

***"count":2,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test event 1",***

***"sport":"football",***

***"type":"team",***

***"geolat":null,***

***"geolng":null***

***}***

***},***

***{***

***"result":***

***{***

***"id":"2",***

***"name":"test event 2",***

***"sport":"tennis",***

***"type":"individual",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

Method: GET

URL: /event/{ID}

Response code: 200

Returns the full details for event identified by its {ID}

Example response:

***{***

***"service":"Event",***

***"count":1,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test event 1",***

***"sport":"football",***

***"type":"team",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

To alter data using this service there are 5 calls that can be made.

Method: POST

URL: /event

Response code: 201

Create a new event object and return a list of objects containing a single object that has been created

Example response:

***{***

***"service":"Event",***

***"count":1,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test event 1",***

***"sport":"football",***

***"type":"team",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

Method: POST

URL: /event/{ID}

Response code: 200

Update details for event object identified by its {ID} and return a list of objects containing a single object that has been created

Example response:

***{***

***"service":"Event",***

***"count":1,***

***"resultset":***

***[***

***{***

***"result":***

***{***

***"id":"1",***

***"name":"test event 1",***

***"sport":"football",***

***"type":"team",***

***"geolat":null,***

***"geolng":null***

***}***

***}***

***]***

***}***

Method: DELETE

URL: /event/{ID}

Response code:

Delete event object identified by its {ID} and returns an empty list of objects

This service also supports additional calls shown below.

Method: POST

URL: /event/assign

Response code:

Assign an event to a channel, required parameters (event ID / channel ID)

Method: POST

URL: /event/unassign

Response code:

Unassign an event from following a channel, required parameters (event ID / channel ID)

User – Service Calls

Each full user object contains the following fields of information:

* **id** –
* **name** –
* **email** –
* **password** –
* **facebook –**
* **twitter** –
* **google** –
* **linkedin** –
* **token** -

Method: GET

URL: /user/{ID}

Response code: 200

Method: POST

URL: /user/{ID}

Response code: 200

Method: POST

URL: /user/register

Response code: 201

Create a new event object and return a list of objects containing a single object that has been created

Method: POST

URL: /user/login

Response code: 200 / 204?

Attempts to login as a registered user if successful an API session is started for this user and a token is generated and returned with the user object

Search – Service Calls

This service supports a single call that performs a text search of other objects (channels / events / articles) that match the search criteria and returns 3 arrays in the response resultset that contains all matched objects.

Method: GET

URL: /search

Response code: 200

Returns a list of all objects matching the search criteria, accepted filters (type)