## Logging in and obtaining an OAuth token

Log into the system with your email address and password. If you set token=true in the post entity, then a new OAuth token will be issued in the response.

This token can be presented on each subsequent HTTP request. The token has an expiry date, and will also be explicitly invalidated if presented in a Logout request.

POST <http://127.0.0.1:8080/rest/auth/login>

HEADER content-type = application/json

{

"username": "coach@acme.com",

"password": "secretstuff",

"token": true

}

The response is complex object with a lot of meta-data about the authenticating user and their account.

Look for;

|  |
| --- |
| "token": "9272f1b1-59ef-4275-a566-851f58568f5e",  "tokenExp": 1.481314878219E12, |

The token should be extracted and in all subsequent requests it should be added in as a HTTP header.

Ie the header in a subsequent request would be;

Authorization: Bearer 9272f1b1-59ef-4275-a566-851f58568f5e

The token will expire at the tokenExp time. This is a unix timestamp in milliseconds.

To logout and invalidate a token use;

GET <http://127.0.0.1:8080/rest/auth/logout> and present the token in the header.

## List my athletes and my associate athletes;

GET <http://127.0.0.1:8080/rest/users/delegates/users>

Response: List of user records

The response will include the authenticated user’s “user” record, and then all the user’s where this is an active relationship.

Fields to note include;

* Id - unique system ID for this user
* Relationship - the relationship / delegate access level with this user
* coach - will show the user’s “primary” coach user id. Ie if me.id == user.coach.id then I am their primary coach. If me.id != user.coach.id or user.coach == null, then the user is an associate athlete to me

Example VUser;

{

"id": 84.0,

"email": "peter@hotmail.com",

"md5": "98ae5642f4dd8d699d837bae7067e9f2",

"dob": 7.65072E11,

"firstname": "Peter",

"lastname": "Pumpkin",

"timezone": "Europe/Paris",

"lat": 43.70943,

"lon": 7.29562,

"relationship": "coach",

“coach”: { “id”: 67 },

"premiumExpiryTs": 9.223372036854776E18,

"isPremium": true,

"hasPwr": true,

"hasBpm": true,

"lastLogin": 1.474803231372E12,

"schedulePref": "prefer\_single",

"roles": [

"user",

"premium"

],

"attsMask": 270.0

},

## Query for recent activities

POST <http://127.0.0.1:8080/rest/users/activities/search/0/100>

HEADER content-type = application/json

{

"criteria": {

"fromTs": 1451566800000,

"toTs": 1496239200000,

"isNotNull": [

"fileId"

]

},

"fields": [

"name",

"training",

"distance",

“avgWatts”,

“avgBpm”,

],

"opts": 1

}

In the path of the request, the 0/100 is paginated search options. Ie start from offset 0 and give me 100 records.

In the entity body;

* criteria.fromTs - a unix timestamp in ms
* criteria.toTs - a unix timestamp in ms
* fields - include these fields in the result set (lots more available here if need be)
* opts - directive to disable verbose field logging in the response

The result will include;

* cnt - the total number of records which matched this search criteria
* result.results - an array of activity records

Note - in the result set, look for the fileId field. In some of our more recent API result sets this may also be called activityId.

# File upload API

## **Support file extensions**

|  |  |  |
| --- | --- | --- |
| Method | GET |  |
| Query path | /rest/files/supports |  |
| Response | List<DataFileType> | ie ["fit","ant","gpx","act","tcx","srm","pwx","json","bdx"] |

## **File upload**

|  |  |  |
| --- | --- | --- |
| Method | POST |  |
| Content type | multipart/form-data |  |
| Query path | /rest/files/upload |  |
| Form parms | * attachment - the file * filename - a unique filename with extension * json - optional params | ie filename - 201512010700.fit  Note - The filename can also be set in the json params |
| Response | VUserWorkoutFile | ie {"id":5038023,"ts":1429395794000 … } The id is the unique “fileId” which can be used for future file operations. |

The optional params is a DataFileUploadContext

json object. It can be use to set alternate values to be used when processing the file.

Sample values include;

|  |  |  |
| --- | --- | --- |
| workoutId | Long | UserWorkout.id - force this ride to be associated with this scheduled workout |
| eventId | Long | UserEvent.id - force this ride to be associated with this scheduled event |
| filename | String | Set the filename here rather than in the form attribute |
| thresholdWatts | int | The rider’s threshold power (FTP) |
| thresholdBpm | int | The rider’s threshold heart rate (AT) |
| equipment | Equipment enum | road, mtb, cx, tt, … etc - set the type of bike this ride was completed on |
| weight | double | The rider’s weight (kg) |
| name | String | An alternate name for this file. ie “Tuesday bunch ride” |
| userId | Long | The userId to upload the ride file to. ie for coach/team multi-rider access |

ie a sample json string could be;

{"filename":"2015-04-19-08-23-14.fit","thresholdWatts":350,"thresholdBpm":168,"weight":70.1}

## **File delete**

|  |  |  |
| --- | --- | --- |
| Method | GET |  |
| Query path | /rest/files/delete/<fileId> |  |
| Response | N/A |  |

## **File download**

This will download the original data file.

|  |  |  |
| --- | --- | --- |
| Method | GET |  |
| Query path | /rest/files/download/<fileId> |  |
| Response | Octet stream |  |

# 

# 

# 