Livedata REST API [beta]

The API combines data from XContest livetracking server and XContest flight uploads into single data source.
Data of (both live and uploaded) **flights** are returned up to **48 hours** into the past.

There are two endpoints - **Live-users** and **Live-track**.

Live-users retrieves basic info about last known location and available flights of user(s). Live-users endpoint may be requested at regular intervals. **Minimal secure interval** between consecutive calls is **1 minute** for a single client.

Live-track retrieves a flight with a full tracklog (typically for an uploaded flight), or the last increment of the tracklog after the *lastfixtime* (typically for a live flight). The endpoint should be requested based on the previous *Live-users* endpoint call, only if detailed tracklog is needed.
There are two situations, when *Live-track* endpoint is called:

- there is a new flight, both live and uploaded, of a user:
    - *Live-track* endpoint may be called without *lastfixtime* parameter
    - It returns a flight data with the whole available tracklog
- there is a new tracklog increment of a live flight:
    - *Live-track* endpoint must be called with *lastfixtime* parameter
    - *lastfixtime* must contain the time of the last gpx fix, which client application already retrieved before
    - It returns a flight data with the tracklog increment after the *lastfixtime*

Normally, the following request on *Live-users* enpoint with the same parameters should retrieve the same data, as retrieved by the previous call, or with increments (new users/flights). But clients should be able to handle **users/flights removal** as well (for example in case of too short live flights under 2 minutes, deleted/deactivated/disapproved uploaded flights…). If the user or flight is missing in the consecutive api response, the data of the user/flight should be discarded on the client side as well.

Authorization

Authorization is ensured by basic access authorization.
All requests on the API endpoints must contain HTTP Header:

```
Authorization: Bearer {your JWT token}
```

You can generate your JWT token on XContest website or contact us on info@xcontest.org and let us know:

- name of your app which will process XContest livedata
- purpose of your app
- IP address from which API requests will be sent

Live-users endpoint

Returns the data about users (pilots), their **last known position** (*lastLoc*) and available **flights** (*flights*) within defined timeframe.
The response doesn't contain track data (GPS fixes), only first and last gps fix of each flight.
To get a tracklog of a particular flight, see Live-track endpoint.

URL of the Live-users endpoint: https://api.xcontest.org/livedata/users

Request examples

```
{live-users-endpoint}?entity=contest:ranacup2022&opentime=2022-03-31T00:00:00Z&closetime=2022-04-01T00:00:00Z

{live-users-endpoint}?entity=group:friendly-event-2022&opentime=2022-03-31T00:00:00Z&closetime=2022-04-01T00:00:00Z

{live-users-endpoint}?entity=club:1214&opentime=2022-03-31T00:00:00Z&closetime=2022-04-01T00:00:00Z

{live-users-endpoint}?entity=user:kUgsHrVb3TSwVp23o9P1LsEaIWPZ&opentime=2022-03-31T00:00:00Z&closetime=2022-04-01T00:00:00Z&source=live
```

Request parameters

- **entity** - identification of the requested entity in form *type:identifier*
  - required
  - **type** - type of requested entity
    - **contest** - get all available flights of users from the contest (for contest organizers/scorers)
    - **group** - get all available flights of users from defined group
    - **club** - get all available flights of users from the club. A club is always associated with a (usually national) contest - a pilot must be registered in the current season of the contest and not disapproved here.
    - **user** - get all available flights of a user (for individual users to get their own data)
  - **identifier** - identifier of the entity
    - if type == contest: identifier of the contest season on XContest
    - if type == group: identifier of the group
    - if type == club: ID of the club
    - if type == user: UUID idenfier of the XContest user
- **opentime** - iso8601 UTC datetime of the "task opening"
  - required
  - data from tracklogs started before *opentime* are not returned
  - must not be set in the future (later than current time)
  - maximum allowed difference between current time a *opentime* is +48 hours

    - must not be set before contest begin time (if entity type == contest), which is at contest begin date at 00:00:00 UTC
    - recommendations:
        - for general use/free XC flying:
            - local application (within one country/timezone): beginning of the current day (local time midnight or sunrise of the current day etc.)
            - global application (all around the world): *current time - 24 hours*
        - for competitions events/tasks: beginning of the launch time window or shortly before
        - use more specific time if it's important to filter out irrelevant activity before the predefined time window
- **closetime** - iso8601 UTC datetime of the "task closing"
    - optional
    - data from tracklogs started after *closetime* are not returned
    - if not specified, it's internally set to *opentime* + 24 hours (for example, if *opentime* is 2022-03-30T10:00:00Z, *closetime* is set to 2022-03-31T10:00:00Z)
    - difference between *closetime* and *opentime* must be > 0
    - maximum allowed difference between *closetime* and *opentime* is +24 hours
    - maximum allowed difference between current time a *closetime* is +24 hours
    - must not be set after the contest end time (if entity type == contest), which is at contest end date + 24 hours (if the contest ends at 30th September, contest end time is 1st October 00:00:00 UTC).
    - recommendations:
        - for general use you can safely ignore the explicit value (because by default it's set to opentime + 24 hours)
        - use explicit specific time if it's important to filter out irrelevant activity after the predefined time window
- **source** - if needed, specify the source of flights (*live* or *upload*)
    - optional
    - if not specified, flights/users from both livetracking and XContest uploads are returned
    - **live** - only flights from XContest livetracking server are returned
    - **upload** - only flights uploaded to XContest conventionally are returned
    - in case of *source=live*: keep in mind, that those flights may be originated on XContest livetracking server and then automatically uploaded to XContest

Response

JSON data:

```
{
      "users": {
            "kUgsHrVb3TSwVp23o9P1LsEaIWPZ": {
                  "username": "XContest-user",
                  "fullname": "John von XContest",
                  "lastLoc": {
                        "type": "Feature",
                        "geometry": {
                              "type": "Point",
                              "coordinates": [
                                    7.602978,
                                    46.154038,
                                    210,
                                    {
                                          "t": "2022-04-04T16:30:40Z",
                                          "b": 201
                                    }
                              ]
                        },
                        "properties": {
                              "source": "live",
                              "landed": true
                        }
                  },
                  "flights":[
                        {flight},
                        {flight}
                  ]
            }
      }
}
```

- key in *users* object is **uuid** of user on XContest
- *lastLoc* is the last know location in form of GeoJSON Point Feature
- *lastLoc.geometry.coordinates* is the value of the *lastFix* of the last flight
- *lastLoc.properties.source* is the value of the source of the last flight (if the position is from the livetracking of from the flight uploaded normally)
- *flights* are ordered by start time in descending order (oldest first)

{flight} format:

```
{

      "uuid":"0a869d43-0b43-4831-a885-649f1a891f04",
      "firstFix": [
            7.489853,
            46.206353,
            2614,
            {
                  "t": "2022-04-04T14:19:31Z",
                  "b": 2578
            }
      ],
      "lastFix": [
            7.602978,
            46.154038,
            210,
            {
                  "t": "2022-04-04T16:30:40Z",
                  "b": 201
            }
      ],
      "source": "live",
      "landed": false,
      "clientId": "xctrack",
      "simulation": false,
      "glider": "SKY PARAGLIDERS Apollo 2",
      "faiClass": "FAI-3 (PG)",
}
```

- *uuid* - string; a flight unique identifier
- format of **fix** (lastLoc.geometry.coordinates / firstFix / lastFix) array: `[`
  ```
  x/longitude dd.dddddd [float (-180;180)],
  y/latitude dd.dddddd [float (-90;90)],
  gps altitude m,
  {
        "t": iso8601 UTC datetime,
        "b": baro altitude m
  }
  ]
  ```
- *source* - string; one of:
  - *live* - the data from the livetracking
  - *upload* - flight has been uploaded/claimed to XContest server
- *landed* - boolean
  - if true, flight has been already finished
  - if false, it's ongoing flight (expect further tracklog data)
  - uploaded flights (*source: upload*) are always landed

- *faiClass* - string; one of:
    - 'FAI-3 (PG)',
    - 'FAI-1 (HG)',
    - 'FAI-5 (Rigid wing)',
    - 'FAI-2 (Rigid wing)',
    - 'RPF (PPG Foot-launched)',
    - 'RPL (PPG Trike)',
    - 'RAL (Landplane/Movable aerodynamic control)',
    - 'RWL (Landplane/Weight-shift control)',
    - 'RWF (Foot-launched/Weight-shift control)',
    - 'RGL (Landplane/autogyro)',

Live-track endpoint

Returns the data (=> tracklog) of the indiviual flight, identified by flight uuid.
Flights uuids are retrieved from the Live-users endpoint.
You don't need to request Live-track endpoint unless you need incremental tracklog data of the flight. If you only need last known position of the pilot, it's retrieved by the Live-users endpoint directly.
In case you request live-track endpoint repeatedly to get the newest tracklog data, you **should always** call it with *lastfixtime* parameter whenever it's possible. In such case you get only the last increment of tracklog data (and not the whole tracklog each time you repeatedly call the endpoint). There are only two special cases when client should get whole tracklog data of the flight at once: by initial API call (the client has no tracklog data yet that day) and for the uploaded flight (flight which wasn't live before and has been just uploaded conventionally to XContest server).

A flight track is in form of GeoJSON Linestring. Gps fixes (tracklog) are contained in the *flight.geometry.coordinates* array.
By default, it returns all available gps fixes of the flight's tracklog. If *lastfixtime* is specified, only gps fixes beginning from the next immediately following after the lastfixtime are returned. If there are no available (new) fixes, the *coordinates* array may be empty.
Time values for each fix must be reconstructed iteratively (see {gpsFix} format) - that's why *flight.properties* object contains *firstFixTime* (it's needed) and *lastFixTime* (it's handy for checking - the reconstructed time value of the last fix should be equal).

URL of the Live-track endpoint: https://api.xcontest.org/livedata/track

Request examples

```
{live-track-endpoint}?entity=contest:ranacup2022&flight=0a869d43-0b43-4831-a885-649f1a891f04&lastfixtime=2022-04-04T16:30:40Z

{live-track-endpoint}?entity=group:friendly-event-2022&flight=0a869d43-0b43-4831-a885-649f1a891f04&lastfixtime=2022-04-04T16:30:40Z

{live-track-endpoint}?entity=club:1214&flight=0a869d43-0b43-4831-a885-649f1a891f04&lastfixtime=2022-04-04T16:30:40Z

{live-track-endpoint}?entity=user:kUgsHrVb3TSwVp23o9P1LsEaIWPZ&flight=0a869d43-0b43-4831-a885-649f1a891f04&lastfixtime=2022-04-04T16:30:40Z
```

Request parameters

- **entity** - same as on *Live-users* endpoint
- **flight** - uuid identifier of the flight

    - required
- **lastfixtime** - iso8601 UTC datetime of the last data received previously by the client
    - optional, but strongly recommended
    - if not specified, all available track data are returned
    - if specified, only track data after the *lastfixtime* are returned (part of the track data started on the closest following fix)

Response

JSON data

```
{
      "flight": {flight}
}
```

{flight} format:

GeoJSON object

```
{
      "type":"Feature",
      "geometry":{
            "type":"Linestring",
            "coordinates": [
                  {gpsFix},
                  {gpsFix}
            ],
      },
      "properties": {
            "uuid":"0a869d43-0b43-4831-a885-649f1a891f04",
            "firstFixTime": "2022-04-04T14:19:31Z",
            "lastFixTime": "2022-04-04T16:30:40Z"
      }
}
```

- *properties.uuid* is a flight unique identifier (same as flight.*uuid* in Live-users enpoint response)

{gpsFix} format:

(in *geometry.coordinates* array)

```
[
      x/longitude dd.dddddd [float (-180;180)],
      y/latitude dd.dddddd [float (-90;90)],
      z/gps altitude m [integer],
      {
            "dt": delta time in seconds from previous fix; default
      1sec (ommited in most cases),
            "b": baro altitude m; ommited if there is no barometric
      altitude
      }
]
```

- only first 3 elements (longitude, latitude, gps altitude) are always present
- 4th element (object) is ommited if *dt* (delta time) is default/omitted and no *b* (baro altitude) is present
    - *dt* (delta time) is the difference between current fix and previous fix in seconds; default value is 1 - in such case dt is completely omitted

Timestamps of gps fixes

Timestamp for each fix must be **restored by iteration** of *coordinates* array

- first fix has usually the timestamp of the *properties.firstFixTime*. That's why there is usually "dt" : 0
- following fixes usually add 1 sec (and thus *dt* is omitted, because 1 is default value).
- in other cases (delta time is not 1 second) dt is explicitly present. Simply add *dt* value (number of seconds) to the previous fix timestamp value