# Technical note for tracking app

## App functionality

We are considering two different operational modes for the app. One is a high-performance mode to be used for competition flying, and the other is a casual mode with she is used for flight tracking when not being part of a contest. The two modes differ only in the location reporting settings, but some mode specific functionality is required in the interface.

### Basic common functionality

The base interface should be kept as simple as possible:

1. On/off switch
   1. Optional with slider/time field to set the duration until auto stop tracking. Automatic fallback value of 12 hours if nothing has been set by the user.
   2. May be two separate place which is, one for competition and one for casual
2. Connection status. Green when we have successfully transmitted a position within the last 5 to 10 seconds, Red together with elapsed time since last success when more than 10 seconds (insert appropriate times).
3. Onboarding functionality as described in section “App login”.
4. Link to the main competition tracking site <https://tracking.airsports.no/>

### Location reporting settings

1. Server URL is <https://traccarclient.airsports.no/> (subject to verification)
2. Wake lock (if applicable) should be on
3. Off-line buffering should be on

#### Competition reporting settings

1. Location accuracy should be always High, so no need for control
2. Location frequency should be 1 second
3. With such high tracking frequency, reporting distance and reporting angle (names from the android client) can be 0.

#### Casual reporting settings

The following settings should give a good balance between data volume and tracking accuracy. The website will not update positions more often than 30 seconds, and 1500 m is 30 seconds when flying in 100 kn. The five-minute location frequency is for when the aircraft is standing still.

1. Location accuracy should be always High, so no need for control
2. Location frequency should be 300 seconds
3. Reporting distance should be 1500 m
4. Reporting angle should be 10°

### Additional user interface for casual reporting

When doing casual flying the tracking app is linked to the authenticated person. We do not wish to display the person’s name on the global map, but it should be fine to display the aircraft registration. Therefore, whenever enabling casual reporting, the user needs to be able to update his aircraft registration. This is a field on the profile settings, same as will be updated as part of the registration process in section “App login”. This can be achieved for instance by popping up a modal when the user starts casual tracking that is pre-populated with the current aircraft registration. The user can then either continue directly to start tracking or update the aircraft registration which triggers a post to update the profile on the live tracking website. It is acceptable to start tracking before the profile is updated.

### Optional features

1. An optional feature is to store the tracking log off-line inside the device either as a series of NMEA sentences if that is available, or as a GPX file or something similar. If we do this, then it must be simple to access the file (listed on the front page of the application, a list of “unhandled track logs”) and share it through common sharing mechanisms (email, whatever). Once a log has been shared, it can be deleted. Old logs (more than a day or two old) should be deleted automatically.
2. Push notification. For instance, whenever a contest is starting, push this notification to the app with a link to the contest tracking map. Only receiving push notifications inside the app is required, pushing the actual notifications is handled by the tracking website.

## App login

The issue addressed in this section is to uniquely and securely identify the user of the application (both android and iOS) so that we can be reasonably certain that the user of the app matches the linked user profile in the tracking website (which in turn is linked to the contestant in a task).

The main concern is to protect the system from bad input data (as opposed to malicious input data), since with the open traccar platform it is impossible to secure the entire system. After all, with the open traccar client app it is easily possible to connect with whatever tracking ID the user desires. However, by not publishing the server address and keeping tracking IDs relatively long and hidden we can through obscurity buy ourselves a reasonable and sufficient security.

Malicious users can still subvert existing tracking IDs and mess up the live tracking by interfacing with the traccar server using other software, but that is beyond the scope of this solution (and is made difficult by random enough tracking IDs (and connecting to the server using https).

There are two main drawbacks with having the device generating a unique ID.

1. Is it possible to generate a globally unique ID?
2. A long unique ID is difficult to copy from the device screen to the web interface for the contest organiser. It is probable that typing errors will occur, leading to unexpected issues when the contestant does not appear on the tracking map as expected.

The following approach is perhaps a bit overkill, but it allows for a tighter integration between the live tracking website and the tracking app, avoiding the need for copying complex ID strings. This is an alternative to bullet point 2 of the basic user interface. I propose the following:

1. When starting the application for the first time, these are is presented with a screen to enter an email address and password.
2. When pressing a "submit" button, the app POSTs the email address and password to the Web server, and the Web server checks if a user is identified by this email address already exists (basically a login).
   1. If the email address exists and the password was accepted, the post responds with a 200 response code. No further action is required from the user.
      1. The response includes an authorisation token to be used for all further API requests. This should be stored internally in the application (it is not necessary to store the password).
         1. If token authentication fails it is probable that the authentication token has been changed on the server, in which case the user must login again through the app.
      2. The app retrieves the profile for the user which includes tracking ID and aircraft registration (which may be null).
   2. If the email address does not exist, the web server responds with a 400 response.
      1. The app should display an extended registration screen where the user can enter first name (required), last name (required), phone number (optional), email address (required, remembered from the initial screen and therefore perhaps not displayed to the user as an input option), password(required), and a profile image (optional).
         1. At a minimum, we need first name and last name which is required by the data model, as well as email and password.
      2. When the user has entered the required information and pressed “submit” the information is POSTed to the web server.
         1. When receiving the 200 response, the app should present a screen for the user to enter the received verification number.
         2. The app POSTs the verification number together with the email address to the Web server
         3. If the verification number is wrong, the Web server will respond with an appropriate status code (somewhere in the 400 range?) and the process is repeated from 2.
         4. If the verification number is correct, the application responds with 200 and continues to 2.a.
3. Once the user is authenticated (either as an existing user or as a new user) the app can GET the user profile information from the Web server using REST with token authentication. This profile information can be displayed on the home screen of the tracking app or something similar.

It is not my intent that the app should be used to perform profile maintenance since this can be easily achieved through the website, but maybe it is a nice touch to allow the user to update his profile image or a short biography through the app.

The user account can be created either beforehand through the website in which case the email address will be identified in point 2a, or through the application registration process. The user can access the website by clicking a “forgotten password” link on the website to receive an automatically generated password by email to access his profile through the website.

With this authenticated link between the website and that the app, it is easy to further extend the app in the future to provide services such as observation registration in flight.

The required REST end points have not yet been created. Once we decide on the implementation details it will be trivial to create the endpoints and deploy the update to the live server for integration testing.