KTrax UDP and JSON interface

KISS
Technologies

Zürich, January 2019

Contents

1 Revisions

Date	Version	Author	Description
June 10, 2015	1.00	gw	Initial release
June 12, 2015	1.03	gw	Added disclose_id, name3, flags Clarifications and units Updated JSON interface
June 22, 2015	1.04	gw	Document formatting
July 3, 2015	1.05	gw	Added JSON fields: Turn rate, stealth
July 15, 2015	1.06	gw	Added CGI params `max_results`, `undisclosed`, `static`
September 14, 2015	1.07	gw	`map_orientation` clarifications
October 27, 2015	1.08	gw	JSON altitude: MSL, `ktrax_info`
November 4, 2015	1.10	gw	Added JSON/sortie interface
December 3, 2015	1.22	gw	Added SOS notification, JSON filter parameter
January 25, 2016	1.24	gw	Added `turn_rate`
February 29, 2016	1.26	gw	Added `name4`
May 22, 2016	1.30.4	gw	Added `max_age`; Clarifications; examples; troubleshooting
July 16, 2018	2.2.5	gw	Logbook additions and corrections
September 18, 2018	2.2.6	gw	Deprecate relative dates for logbook
January 28,2019	2.2.6	gw	Correct outdated links, focus on aviation

2 Introduction

KTrax is a high performance realtime tracking solution developed by KISS Technologies GmbH. Applications range from Air Traffic Control over ride sharing apps to asset tracking or law enforcment. KTrax is available for client installation or as Sofware as a Service (SaaS). For more information, please see https://www.kisstech.ch/pdf/datasheets/ktrax-en.pdf or visit our homepage at http://www.kisstech.ch/.

This document describes:

- The KTrax native tracking UDP protocol
- The KTrax JSON service

3 Existing KTrax servers

The following servers are currently active:

- UDP/ktrax.kisstech.ch:48890: Aviation

4 Licensing

Data from KTrax servers is available for for free for private and personal use only. For other use cases, please contact us for licensing arrangements. Please do not scrape data.

Licensing of the KTrax software is also possible. Please contact us for details.

5 The KTrax native tracking protocol (UDP)

Note: See https://github.com/gewesp/ktrax-mobile for an open-source Android application which implements the native KTrax UDP protocol.

The native tracking protocol allows mobile clients to track their position on KTrax. It has the following characteristics:

- UDP based
- Packet structure: Blank-separated ASCII
- Millisecond values must be integers (no decimal point allowed), other values may have decimal places.
- Client sends ktrax_pos to server, server replies by ktrax_update and occasional special purpose packets
- Client → server: ktrax_pos, ktrax_hello, ktrax_bye
- Server → client: ktrax_update, ktrax_url, ktrax_bye
- Loss of isolated packets is tolerated

Note: Binary packet encoding will be added in the future.

A single UDP source (IP address/port combination) can transmit data for at most 5 targets.

5.1 Periodic packets

5.1.1 The ktrax_pos packet (client to server)

Usage: Aviation

Periodicity:

- 0.1 to 1 packets/s for aviation tracking (indicative value)

Summary:

```
ktrax_pos <id> <seqnr> <systime>
          <n_received> <loss_pct> <rtt_lat> <tdraw>
          <nettype> <cellid> <roam> <batt> <maghdg>
          <fix_systime> <vehicle_type_and_flags>
          <gpstime> <lat> <lon> <alt>
          <speed> <course> <vertical_speed> <accuracy>
          <test_systime>
          [ <turn_rate> ]
```

Optional parameters are listed in [brackets].

id: Client ID, e.g. andr:a5f3e8d04d2f9f28

seqnr: Sequence number of this ktrax_pos packet, should start with 0 and increase by 1 for each packet

systime: System time when packet was sent [ms], any fixed system time reference is OK

n_received: Number of ktrax_update packets received

Informational values, not necessary for operation:

loss_pct: Packet loss [%], uses `n_received`

rtt_lat: Round trip latency as determined by client [ms]

tdraw: Radar drawing time [ms]

nettype: Current network type [integer], see below for values

cellid: Current network ID if cellular network [integer]

roam: 1 when roaming, 0 otherwise

batt: Battery level [%]

maghdg: Magnetic heading [degrees], 1800 if not available

Mandatory values:

fix_systime: System time when fix was obtained [ms]; same time reference as `systime`

vehicle_type_and_flags: Target type [integer], see section 7.

The following flags can be OR'ed with the value: 0x100: Allow target to appear on tracking site 0x200: Emergency/SOS

gpstime: GPS time [ms] since 1970 epoch, UTC, -1 for invalid fix

lat, lon: GPS latitude/longitude [degree]

alt: GPS altitude over the WGS84 ellipsoid [m]

speed: Horizontal speed [m/s]

course: Course over ground [degree]

vertical_speed: Vertical speed [m/s], up = positive

accuracy: GPS accuracy estimate [m]

System test:

test_systime: Last time TEST button was pressed ([ms], same reference as `systime`)

Optional parameters:

turn_rate: Turn rate [degree/s], positive: turning right

Notes:

- Longitude in [-180, 180], latitude in [-90, 90].
- Not all parameters are mandatory. Not applicable values should be set to -1 unless noted otherwise.

5.1.2 The ktrax_hello packet (client to server)

Usage: Aviation

Periodicity:

- 1 packet/30 minutes, starting at beginning of session
- A new hello packet each time the client configuration changes (e.g., networking status)
- It is recommended to repeat the packet a few times.

Summary:

```
ktrax_hello <id> <systime>
            <clientversion>
            <devicemodel> <systemversion>
            <network_operator> <network_country_iso>
```

The fields `id` and `systime` are as for `ktrax_pos`.

clientversion: Client software version [string]

devicemodel: Client device model [string]

systemversion: Client device Operating System version [string]

network_operator: Network operator name (if available)

network_country_iso: Network country ISO code (if available)

5.1.3 The ktrax_info/ktrax_info2 packet (client to server)

Usage: Aviation

Periodicity:

- Sent in conjunction with `ktrax_hello`.

Summary:

```
ktrax_info <id> <systime>
           <name1> <name2> <name3> ...
ktrax_info2 <id> <systime>
           <name1> <name2> <name3> <name4> ...
```

The fields `id`, and `systime` are as for `ktrax_pos`.

name1: Callsign [string]

name2: Short (alternative) callsign [string]

name3: Reserved

name4: Only `ktrax_info2`: Symbol color (e.g. `'#a1efd2'`)

`name1` and `name2` must contain only upper-case letters, numbers, hyphen or underscore.

Additional fields may be added in the future.

5.1.4 The ktrax_data packet (client to server)

Summary:

```
ktrax_data <id> <seqnr> <systime> <channel> <keep>
           { JSON data }
```

- Transmits arbitrary user-defined data ("user data") to the server.
- Data are sent in JSON objects.
- Data are in named `channels` and each channel will keep `keep` back items in a circular buffer.

The fields `id`, `seqnr` and `systime` are as for `ktrax_pos`. The same sequence number counter should be used for `ktrax_pos` and `ktrax_data`.

channel: ...

keep: ...

5.1.5 The ktrax_update packet (server to client)

Usage: Aviation

Summary:

```
ktrax_update <sent_systime> <n_clients> <map_orientation> <self_moving>
             <notification> <n_targets>
             [ A0 <gpstime> <target_id> <vehicle_type>
                <distance> <bearing> <relative_altitude>
                <speed> <display_course_degree> <vertical_speed> <heading>
                <tCPA> <dCPA> <danger> <notification> ]
             [ A1 ... ] ... [ A4 ... ]
```

sent_systime: `systime` of ktrax_pos packet this packet replies to

n_clients: Current total number of clients

map_orientation: 'UP' direction on map [degrees]. Can be heading, track or just North.

self_moving: This client moving? 1 : 0

notification: Audio alarm level 0 ... 3, 4 for SOS/emergency, error notification 101, ...

n_targets: 0–5, Number of following targets A0 ... A4

A0 ... A4: Markers for target data block For each target, the marker is followed by the following block:

gpstime: GPS time of last fix from this target

target_id: Target ID, e.g. andr:a5f3e8d04d2f9f28

vehicle_type: Target type, e.g. 101 for boat, 151 for buoy etc.

distance: Distance own ship → target [m]

bearing: True bearing from own ship → target [degrees]

relative_altitude: Target minus own ship altitude [m]

speed: Target speed [m/s]

display_course_degree: Target course over ground, 1800 if not moving

vertical_speed: Target vertical speed [m/s], positive = climbing

heading: Target heading [degree] (future extension)

tCPA: Time to closest approach [s]

dCPA: Distance at closest approach [m]

danger: Danger level (currently unused)

notification: Notification for this target, for radar symbol colors

The `ktrax_update` packet is:

- Sent only as reply to ktrax_pos packets to the source IP address/port.
- Contains server information, map orientation, audio notification and up to 5 targets

5.2 Special purpose packets (messages)

Usage: Aviation

Messages are used for one-time actions like opening URLs or termination of the client.

- Message packets are sent only as reply to ktrax_pos (in addition to ktrax_update)
- Message packets are repeated at least 5 times after issuance by the server
- Each message has a `code` serving as a unique integer identifying the message. Clients ignore subsequent packets with the same code.

5.2.1 Bye message

Server to client

`ktrax_bye <code> <reason>`

- Terminates client
- Can be used e.g. to auto-shut down client on inactivity or leaving the lake.
- `reason` Short string shown to the user.

Client to server

`ktrax_bye <id> <systime>`

The client indicates to the server that it's terminating and will no longer send packets. See `ktrax_pos` for `ID` and `systime`.

5.2.2 URL message

Summary:

`ktrax_url <code> <reason>`

- Opens a browser on the given URL.
- Used to inform users about updates, system status etc.
- **Use sparingly!**

5.3 Examples

5.3.1 Gliding

A glider G-CJLO agreeing to be tracked (type 1 OR'ed with 256, thus 257), with a purple symbol (color ff00ff) might send the following position and info packets:

```
ktrax_pos demo:client1 0 0 0 0 0 0 -1 -1 0 100 90 0 257
1418635398000 47 8 0 100 90 0 5 -1000000000
ktrax_info2 demo:client1 0 G-CJLO LO - #ff00ff
ktrax_pos demo:client1 1 1000 0 0 0 0 -1 -1 0 100 90 1000 257
1418635399000 47 8.001317181 0 100 90 0 5 -1000000000
ktrax_info2 demo:client1 1000 G-CJLO LO - #ff00ff
ktrax_pos demo:client1 2 2000 0 0 0 0 -1 -1 0 100 90 2000 257
1418635400000 47 8.00263436201 0 100 90 0 5 -1000000000
```

Notice that the `ktrax_info2` packet is repeated to ensure reliable delivery.

6 Network type codes

6.1 Mobile networks

For an up-to-date list, see https://developer.android.com/reference/android/telephony/TelephonyManager.html

- 0: Unknown/WiFi
- 1: GPRS
- 2: EDGE
- 3: UMTS
- 8: HSDPA
- 10: HSPA
- 13: LTE
- 15: HSPAP

6.2 Special purpose networks

- 201: Open Glider Network
- 202: Airware

7 Vehicle types

7.1 Aviation

See the FLARM Data Port specification.

- 1: Glider
- 2: Tow plane
- 3: Helicopter
- 4: Parachute
- 5: Drop plane
- 6: Hang glider (delta)
- 7: Paraglider
- 8: Power plane
- 9: Jet
- 10: Unidentified Flying Object
- 11: Balloon
- 12: Airship
- 13: UAV
- 14: not assigned
- 15: Static; e.g. captive balloon

8 KTrax JSON service: Tracking

Summary: The JSON interface allows web clients to get all targets with active tracking flag in a specific rectangle.

URL:

```
https://ktrax.kisstech.ch/cgi-bin/ktrax.cgi?db=aviation&<cgi_parameters>
```

Example query: https://ktrax.kisstech.ch/cgi-bin/ktrax.cgi?db=aviation&sw_lat=41.12&sw_lon=4&ne_lat=45&ne_lon=10.1&max_results=1000&max_age=2000&ktrax_id=icao%3a4B1307

CGI parameters:

- Database selection: db=aviation
- Bounding box: `sw_lat=<lat>&ne_lat=<lat>&sw_lon=<lon>&ne_lon=<lon>`
- Return specific ID (in addition to all in bounding box): ktrax_id=id
- Maximum number of results: `max_results=<n>` (Default: 300; ktrax_id will always be delivered, number is currently only approximate)
- Maximum age (time since last update) of targets in seconds: `max_age=<t>` (Default: Infinity, i.e. return all targets regardless of age)
- Include targets with undisclosed ID: `undisclosed=<0|1>` (Default: 1)
- Pre-filter targets: `filter=<all|fixed|moving|sos>` `fixed`/`moving` filters only static/moving targets, `sos` filters only SOS targets. Notice that "static" in this context means targets that are fixed to the earth, e.g. buoys or navigation aids. That is, parked aircraft are classified as `moving`, not `fixed`. FLARM devices with vehicle type "static" are `fixed`.
- DEPRECATED (use `filter` instead): Include static targets: `static=<0|1>` (Default: 1)

Note 1: Use URI escaping, e.g. ogn:DD8EE2 becomes ogn%3aDD8EE2
Note 2: The db parameter must currently come first in the URI.
Note 3: To obtain SOS targets without hitting the maximum number of targets, use `filter=sos`.

Example reply:

```
{
"server_stats": { "n_mobile": 2, "n_tracking": 2,
                  "n_disclose_id": 2, "n_stealth" : 0,
                  "n_fixed":  1 },
"targets": [
  ["demo:client1", 101, "2014-12-15T09:23:18Z", 1, 47, 8, 0, 0, 5,
   1.4, 0, 0, 0, 0, 1, 0, "no_relay", ["", "", "", ""], 100, 0, 0],
  ["demo:client2", 101, "2014-12-15T09:23:18Z", 1, 47, 8, 0, 0, 5,
   1.3, 90, 0, 0, 0, 1, 0, "no_relay", ["", "", "", ""], 100, 0, 0],
  ["190_THEMSE_4", 182, "2014-12-15T09:23:18Z", 0,
   51.46235, -0.31757355, 0, 0, 5,
   0, 1800, 0, 0, 0, 1, 0, "no_relay", ["", "", "", ""], 100, 0, 0]
]
}
```

Subfields of server_stats

n_mobile: Total number of mobile clients

n_tracking: Number of mobile clients that can be disclosed over the JSON interface

n_disclose_id: Number of mobile clients that disclose their ID

n_fixed: Number of fixed (static) clients. These always disclose their ID.

n_sos: Number of clients currently in distress (SOS)

Note: Server statistics are only updated every 10 seconds.

Subfields of targets

```
[
  [
    id, type, gps_time, age, lat, lon, alt, RESERVED1, accuracy,
    speed, course, vertical_speed, turnrate, notification, disclose_id, stealth,
    relay,
    [ name1, name2, name3, name4, ... ]
    battery_level, packet_loss, sos,
    RESERVED ...
  ]
  // , ...
]
```

Note: A maximum of 300 targets are returned in the standard version of KTrax. Each row of the array represents a target as follows:

id: Unique identifier [string], should be qualified by "ogn:", "icao:" etc. Max. 21 characters.

type: Vehicle type [integer]

gps_time: Date/time of last known position [ASCII string].

age: Approximate time since last packet was received [s].

lat, lon: Geographic latitude/longitude [degrees]

alt: Altitude above MSL **(Note: Not WGS84)** [m]

accuracy: Horizontal position uncertainty [m]

speed: Horizontal speed [m/s]

course: Course over ground [degrees]

vertical_speed: Vertical speed [m/s, positive UP]

turnrate: Turn rate [degrees/s, positive RIGHT]

notification: Alarm level 0...3, 4 means emergency (SOS)

disclose_id: Whether or not this target allows the id to be disclosed [0/1]

stealth: Stealth mode, applicable to some airborne systems [0/1]

relay: Identifier of station that last received this target [string]

name1, name2, name3, name4: Additional information (vessel name/callsign, flight id, tail number, symbol color etc.) [string]

battery_level: Battery level (if available) [percent]

packet_loss: Packet loss (if available and applicable) [percent]

sos: Distress (SOS) status [0/1]

9 KTrax JSON service: Sortie logbook

Summary: The Sortie logbook JSON interface provides information about takeoff and landing times, location, and launch method of tracked aircraft.

URL:

`https://ktrax.kisstech.ch/cgi-bin/ktrax.cgi?db=sortie&<cgi_parameters>`

Example query (replace the date range!): https://ktrax.kisstech.ch/cgi-bin/ktrax.cgi?db=sortie&query_type=ap&id=LFMF&tz=1&dbeg=2019-01-20&dend=2019-01-30

This query returns all flights from airport LFMF in timezone UTC+1 during 10 days in January 2019.

CGI parameters:

- Database selection: Always `db=sortie`
- Index selection. Search for airport ID or name: `query_type=ap`; search for a callsign: `query_type=cs`
- Specify the search query, either airport ID/name or callsign according to the index selection: `query=<airport_or_callsign>`
- Time range selection; return flights from `begin_date` to `end_date` days back (inclusive). The date is expressed in `YYYY-MM-DD` and the selected time zone: `dbeg=<begin_date>&dend=<end_date>` Example: `dbeg=2015-27-09`.
- Select time zone for reported times in UTC+x hours: `tz=<x>`

  Fractional values (needed for some Australian time zones) are possible, e.g. `tz=10.5`

Airport names may contain UTF-8 characters which must be percent-escaped.

Example reply:


```
{
"begin_date": "2015-10-31",
"query_type": "ap",
"n_entries": 2,
"sorties":
[
{"seq": 144430352901, "id": "flarm:DD1234", "cs": "VH-NXY",
"launch": "T", "tow_id": "flarm:DF1234", "tow_cs": "VH-BAB", "tow_seq": 144430550902,
"type": 1, "date": "2015-11-01",
"tkof": {"time": "09:28", "loc": "YBSS", "rwy": "2"},
"ldg": {"time": "11:00", "loc": "YBSS", "rwy": "2"},
"dalt": "540", "dt": "1:32"},

{"seq": 144430550902, "id": "flarm:DF1234", "cs": "VH-BAB",
"launch": "S", "tow_id": "flarm:DD1234", "tow_cs": "VH-NXY", "tow_seq": 144430550901,
```

```
"type": 2, "date": "2015-11-01",
"tkof": {"time": "09:28", "loc": "YBSS", "rwy": "2"},
"ldg": {"time": "09:33", "loc": "YBSS", "rwy": "9"},
"dalt": "400", "dt": "5"}
],
"sum_dt": "37",
"first_tkof": "09:28",
"last_ldg": "11:00,
"max_dalt": 540
```


This reply lists two flights at the queried airport YBSS (`"query_type": "ap"`). Glider (`"type": 1`) VH-NXY with FLARM ID `"id": "flarm:DD1234"` is taking off at 09:28 local time on runway 02, being towed (`"launch": "T"`) by VH-BAB with FLARM ID `"id": "flarm:DF1234"`.

VH-NXY reaches a maximum altitude (`dalt`) of 540m AGL and lands after 1 hour and 32 minutes (`dt`) at 11:00 local time.

The tow plane (`"type": 2`) takes off simultaneously and lands 5 minutes later on runway 09 after reaching a maximum altitude of 400m AGL.

There was a total of `"sum_dt"` one hour and 37 minutes flown on that day. The first takeoff was at 09:28 and the last landing at 11:00. The maximum altitude reached was 540m AGL.

The launch method `"launch"` can be `"T"` for aerotow, `"S"` for a motor-glider self launch, `"W"` for a winch launch or `""` for a powered aircraft or when it couldn't be determined.

Fields like `"time"`, `"loc"`, `"rwy"` etc. may be empty if the respective values could not be determined.

The `"date"` field refers to the date of takeoff.

The `"seq"` field contains a unique sequence number for each record. Use at least 8 byte integers to process.

10 Troubleshooting

If your targets don't appear on the tracking site, check the following:

- Are you sending the data to the correct IP address and socket?
- Are you using one source socket per tracking client?
- Is the syntax correct?
- Is the vehicle type OR'ed with 0x100 (allow tracking)?
- Are the gpstime fields correct? They have to be UTC and in integer milliseconds since 1970. If you are simulating targets, is your computer synchronized to UTC? Use e.g. NTP synchronization.
- Are millisecond values transmitted as integers, i.e. without a decimal point?
- Are the sequence numbers of `ktrax_pos` packets strictly monotonic?