AFF Consumer Guide - JSON

Introduction

This guide covers integrating with the Spidertracks AFF API which is built in accordance with the AFF JSON Specification, linked here. By using the Spidertracks AFF API, you are classed as a client and must abide by the specification.

1.0 Setup Consumer Account

A consumer is a Spidertracks user account that has been granted API consumer privileges and can query the API endpoint.

1.1 Consumer Connection Account

An account on the Spidertracks website must be created with a valid email address and password. This email address is also used to pass any information about updates, issues, etc. that is associated with the data feed system. It is therefore important to ensure that the email address used can be monitored by staff involved with maintaining the consumer software using the account to retrieve the data feed.

Once this account has been created, the password can be managed by logging into Spidertracks as a normal user (please see Spidertracks support for more information).

1.2 Requesting Service Establishment

Once an account has been created, a request for integration access needs to be lodged with Spidertracks here. We will advise you once this has been processed and integration access has been established for this account.

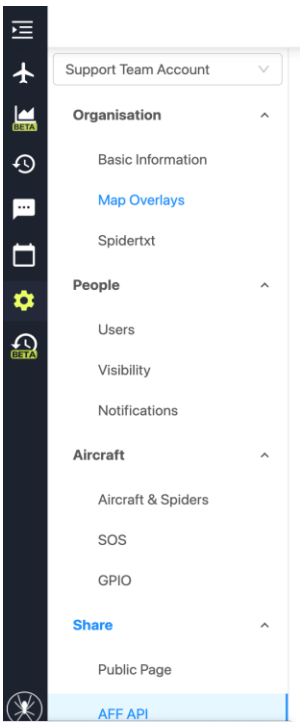
1.3 Publicity

Publicity controls the Spidertracks customers who can allow you to pull their data. Setting an integration feed to private allows only organisations that you are a member of (through the email credentials used for the API) to feed data to that integration account. This is typically used for internal monitoring software within an organisation.

Setting an integration account to public allows all users of the Spidertracks system to make their data available to your data feed as your company name will show up in the list of publicly available feeds. All Spidertracks users can send to the feed, but only the owner of the feed can query the feed for data. A public account does not make the information fed through this integration account visible to the public. The configuration of publicity is performed by Spidertracks support staff.

1.4 Spider Configuration

For an Aircraft to have its location information included in a consumers data feed, it must be subscribed to that data feed. This is managed through the Aircraft - AFF/API menu selection for the Organisation that owns the Spiders.



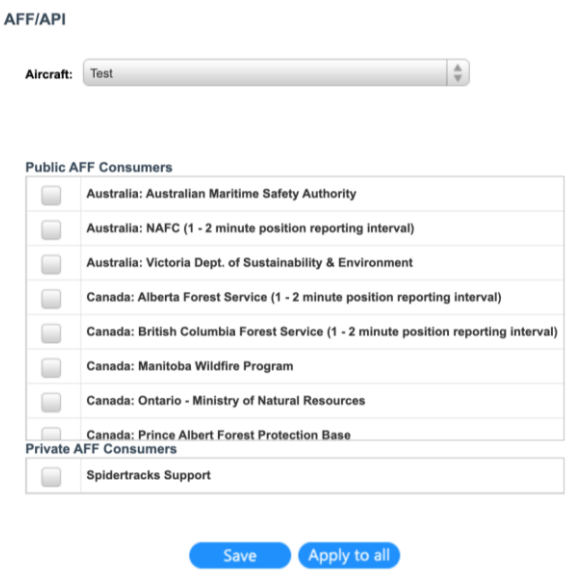


Figure 1: Web interface for the AFF/API page to configure the service.

To configure an aircraft/Spider to send location information, select it in the selection box at the top of the page (Figure 1). Then click the checkbox beside the consumer you wish to subscribe the Aircraft to.

There are the two types of consumers shown, public and private. None of the public consumers are members of this Organisation, however the Aircraft location information can still be sent to the consumer. If the Organisation also has a private consumer established for it, it is shown in the Private AFF Consumers section.

Once an Aircraft has been configured using the interface, its settings can be applied to all aircraft for the Organisation by clicking “Apply to All”. This provides a fast mechanism for configuring the majority of a fleet. Individual aircraft/Spiders can be customised as required after this step.

Note: Only data created after the feed has been enabled will be able to be pulled through the API.

1.5 Validate API Feed Access

To validate your Spidertracks account is set up correctly to process AFF queries, please use your preferred Rest Client to make a query. The next section details how to query the API.

2.0 Query AFF API - JSON

To query the Spidertracks AFF feed, you need to make a POST request to the following Spidertracks API endpoint:

https://apigw.spidertracks.io/go/aff/feed

When making the request, there are several things to do:

- The request needs HTTP Basic Authentication using your Spidertracks website login. Make sure the request has the Authorisation HTTP header set.
- The Content-Type header must be included for the request and be set to application/json.
- The body element should be a valid JSON Object.

2.1 Query Body Sample

The attributes and values that you need to set when making the request are highlighted below. For parameter key definitions, refer to the AFF JSON specification linked here.

```
{
   "type": "dataRequest",
   "dataInfo": [
      {
         "affVer": "1.1",
         "provider": "spidertracks",
         "reqTime": "2020-06-04T01:26:00Z",
         "sysId": "spidertracks"
      }
    ],
   "msgRequest": [
      {
         "to": "spidertracks",
         "from": "Name of Client",
         "msgType": "dataRequest",
         "dataCtrTime": "2020-06-09T00:12:01Z"
      }
   ]
}
```

Green highlights: Should be changed with each query. The “reqTime” parameter should be the time at which the query is made and “dataCtrTime” will adjust the data that is returned.
Yellow highlights: Replace with the name of your organisation.

To ensure you do not miss points when querying the AFF feed, use the following process.

1. Make the initial request to the feed.
2. Process returned points and save the highest dataCtrTime for the next request.
3. Request positions using the highest dataCtrTime read in the previous request as the body date.
4. Repeat the process from step 2.

Please also note:

- The feed should not be queried more frequently than once every 30 seconds.
- The AFF specification requires data to be kept for a minimum of two weeks. Spidertracks does not guarantee that data beyond this period will be available.
- It is helpful if you set the User-Agent header on the request to something that identifies your organisation. This will assist Spidertracks with solving any issues you encounter.
- The Spidertracks feed endpoints are https URLs so you will need to ensure the go.spidertracks.com SSL certificate is trusted by your system.
- Responses are limited to 1000 points at a time. If there are 1500 points in the time frame you are requesting, the first request will return the oldest 1000 points in chronological order. In order to get the next 500 points, make another request using the most recent point time you received as the query body in the second request.

2.2 AFF API Response

The following is an example AFF API response. All the data complies with GeoJSON standard which is a geospatial data interchange format based on JavaScript Object Notation (JSON).

The response includes Additional Telemetry Unit (ATU) events and these are defined in Appendix A.


```
{
   "type": "FeatureCollection",
   "dataInfo": [{
      "affVer": "1.1",
      "provider": "AFF",
      "rptTime": "2017-06-18T21:50:16.719Z",
      "sysId": "spidertracks"
   }],
   "features": [{
      "type": "Feature",
      "properties": {
         "atu": {
            "events": {
               "tank": {
                  "drop": "end",
```

```
                "buttonMode": 97
              }
            }
          },
          "cog": 140,
          "ctrId": "spidertracks",
          "esn": "300034012609560",
          "fix": "3D",
          "hdop": 8,
          "posTime": "2017-08-28T23:44:30Z",
          "dataCtrTime": "2017-08-28T23:44:37Z",
          "spd": 0,
          "src": "GPS",
          "unitId": "HBEAT",
          "trackId": 2137
        },
        "geometry": {
          "type": "Point",
          "coordinates": [174.760535, -36.85815, 100.0]
        }
    }]
}
```

3.0 Troubleshoot and Monitor API

3.1 Error Responses

If there is a problem with the request you will receive a non 200 HTTP status code. The body of the response from Spidertracks will typically contain a description of the error encountered.

3.1.1 Bad Request Response

HTTP Status Code: 400


```
{
  "status": "BAD_REQUEST",
  "messages": [
     "Unrecognized token 'undefined': was expecting ('true', 'false' or 'null')\n at [Source: (PushbackInputStream); line: 5, column: 31]\n at [Source: (PushbackInputStream); line: 5, column: 9] (through reference chain: com.spidertracks.aviator.api.model.aff.AffJsonDataRequest[\"dataInfo\"]->java.util.ArrayList[0])"
  ],
  "errorCode": "missing-required-field"
}
```


3.1.2 Missing AFF Version Response

HTTP Status Code 500


```
{
  "error": "Version null is not valid. Expecting version 1.1",
  "dataInfo": [
     {
        "affVer": null,
        "provider": null,
        "rptTime": null,
        "reqTime": "2021-02-25T00:00:00Z",
        "sysId": null
     }
  ],
  "debug": [
     {
        "reqMeth": "POST",
        "httpHost": "10.135.128.39",
        "httpAccept": null,
        "contType": "application/json",
```


```
        "contLen": 262,
        "script": "/aff/feed",
        "protocol": "HTTP/1.1",
        "referer": "http://go.spidertracks.com/api/aff/feed"
      }
  ]
}
```


3.1.3 No Authentication

HTTP Status Code: 401
<HTML> output

3.2 Heartbeat Data

To ensure the Spidertracks service is running, a heartbeat Spider runs 24/7, configured to send a point every two minutes. There is an option when setting up an AFF feed to send the heartbeat Spider point data to your feed. This provides testing data without having to run your own Spider. You can request the heartbeat data be turned off once you have your system up and running. The heartbeat Spidertracks trackid will approximately change every 24 hours at midnight UTC.

The heartbeat Spider's ESN is 300534060224590 and its unitId is "HBEAT".

Appendix A - Additional Telemetry Unit Events

Additional Telemetry Unit (ATU) events consist of four possible categories with multiple unique key-value pairs inside of them.

The four categories are:

- Aircraft
- Bucket
- Tank
- Hardware

ATU Events Mapping				
ATU Category	**ATU Event Name**	**ATU Event Value**	**Button Mode**	**Explanation**
Aircraft	normal	0 - 10	0 - 10	A normal position point.
Aircraft	now	true	15	A point that was triggered by the Now function in the GO website.
Aircraft	geofence	0	40	Geofence event on geofence 0
Aircraft	geofence	1	41	Geofence event on geofence 1
Aircraft	geofence	2	42	Geofence event on geofence 2
Aircraft	geofence	3	43	Geofence event on geofence 3
Aircraft	geofence	4	44	Geofence event on geofence 4
Aircraft	geofence	5	45	Geofence event on geofence 5
Aircraft	watch	on	50	Watch was turned on
Aircraft	watch	off	51	Watch was turned off
Aircraft	watch	on resend	52	Watch was turned on but the previous Watch on point hasn't been acknowledged as received
Aircraft	watch	off resend	53	Watch was turned off but the previous Watch off point hasn't been acknowledged as received

Aircraft	blocks	on	55	Blocks On Event, the aircraft is deemed to have come to a complete stop after having been on a flight
Aircraft	blocks	off	56	Blocks Off Event, the aircraft has deemed to have moved after power up and has not yet been in flight
Aircraft	automated watch	on	57	Automated Watch On Point (caused by passing through transition speed)
Aircraft	status	speed up	58	Speed Up point (Passing through transition speed - accelerating)
Aircraft	status	slow down	59	Slow Down point (Passing through transition speed less 5 knots - decelerating)
Aircraft	mark 1	value pulled from user's description	61	Mark 1 button press event - Description pulled from user's description
Aircraft	mark 2	value pulled from user's description	62	Mark 2 button press event - Description pulled from user's description
Aircraft	mark 3	value pulled from user's description	63	Mark 3 button press event - Description pulled from user's description
Aircraft	mark 4	value pulled from user's description	64	Mark 3 button press event - Description pulled from user's description
Aircraft	radius	value pulled from user's description	68	Turned on Radius - Description pulled from user's description

Aircraft	heading change	true	69	Heading change event
Aircraft	Spidertxt	true	70	Spidertext Message Event (Text Trigger)
Aircraft	text control message	true	75	Command triggered when the Spider receives a connection/disconnection event
Aircraft	ROC	exceeded	80	Rate of Climb Event (Rate of Climb Trigger)
Aircraft	ROC	normal	81	Rate of Climb Un-trigger Event (Safe)
Aircraft	ROD	exceeded	82	Rate of Descent Event (Rate of Descent Trigger)
Aircraft	ROD	normal	83	Rate of Descent Un-trigger Event (Safe)
Aircraft	power	on	90 - 97*	Engine on event
Aircraft	power	off	90 - 97*	Engine off event
Aircraft	status	takeoff	90 - 97*	Take off event
Aircraft	status	landing	90 - 97*	Landing event
Bucket/Tank	drop	start	90 - 97*	Start of drop event
Bucket/Tank	drop	end	90 - 97*	End of drop event
Bucket/Tank	fill	start	90 - 97*	Fill start event
Bucket/Tank	fill	end	90 - 97*	Fill end event
Any	-	-	100	Custom event
Aircraft	SOS	opened	111	SOS button press event

* Spiders with a General Purpose Input / Output (GPIO) port can connect to switches on the aircraft e.g. the collective switch of a helicopter. The GPIO port has four input pins, each having a button mode corresponding to an active state and a button mode corresponding to an inactive state. Each pin can be configured to any of these events: Engine On, Engine Off, Take Off, Landing, Start of drop, End of drop, Fill Start, Fill End. The customer must set the pin to event configuration within the website. This means that button modes from 90 to 97 can

be associated to any of these events, in any specific order. Hence, AFF API clients should not expect an event for a particular button mode.