Document status

Published status		Confidentiality status	
☐	Draft	☒	Internal
☒	Released	☒	NDA
☐	Canceled	☐	Public

Version control

Ver.	Date	Summary of changes
0.11	2021-03-02	Draft initial version
0.12	2021-03-29	Revise source, add Reception
0.13	2021-10-11	Remove dBm from flarm source, add remote id type.

Scope and summary

This document describes the FLARM JSON protocol interface This is an alternative to the FTD-012 Dataport ICD, aiming to be more user-friendly and flexible, while providing richer data for surveillance and drone applications.

The protocol currently implements streaming of status information and traffic information.

Table of contents

1 Introduction

This document describes the FLARM JSON interface protocol. The JSON ICD shares some functionality with the Data Port protocol described in FTD-012[1]. It is advised to familiarize yourself with this document first. FTD-012 is optimized for in-cabin communication, e.g. for visualization on a display. In contrast, this interface aims to a wider range of application through a more user-friendly way of dealing with messages, more flexibility and being more modern than the protocol defined in FTD-012. Specifically, the advantages are:

- Very easy to parse in most programming environments.
- Human-readable (with appropriate formatting).
- Uses sane units (SI) and names.
- Deals with missing data explicitly.
- Versioned and fully extensible.

The protocol is thus particularly suited for new drone and surveillance applications with their complex architectures and quick innovation cycles.

The FLARM JSON protocol is agnostic to the physical method of transmission. It is most commonly used on RS-232 or UART interfaces. Other implementations include UDP broadcasts or MQTT. Messages are suffixed with Windows-style newline "\r\n" sequence. Messages are otherwise minified, that is all whitespace characters (including newline characters) are removed. For clarity, example messages given in this document are formatted nicely, however.

The message definitions can be fully expressed in Protocol Buffers[2] definitions files, see Appendix A. Protocol Buffers also provides a toolchain for creating code to generate, parse and verify FLARM JSON messages.

To allow for future changes, versioning is applied globally to all JSON messages. An old version will always be a subset of the new version, i.e. new versions will only add fields or messages. Consumers should use best effort when dealing with newer than expected version.

In an implementation, consumers of FLARM JSON should make minimal assumptions when parsing messages, e.g. unknown messages should be ignored, it should not be relied on the order of the keys in the JSON message, optional and unknown keys should not be relied on, etc.

All values are given in SI units (meters, meters per second, seconds, etc.). An exception to this rule is the use of degrees instead of radians for some values,

[11] https://flarm.com/support/manuals-documents/
[2] https://developers.google.com/protocol-buffers

such as the latitude, longitude, turn rate, etc. Time is expressed in seconds elapsed since UNIX epoch (1970-01-01 00:00:00 UTC).

2 Protocol Description

Most messages are sent in an unsolicited way, i.e. triggered by an extrinsic event such as a newly received radio packet. Messages can be sent at a fixed interval, every time an update about a traffic is received, every time a navigation solution is available, etc.

All messages adhere to the following structure:

```
{"<messageType>":{<messageContent>}}\r\n
```

Line returns or other whitespace characters are not permitted within JSON strings. The newline sequence is an integral part of every message and can be used to distinguish individual messages.

<messageType> is a string discriminator describing what payload is given. <messageContent> is a JSON object (i.e. "{ … }") containing the actual payload. The payload can be nested.

The definition of the individual messages is given below. Note that certain types are reused throughout for consistency.

2.1 Heartbeat

Message sent every second to inform about the health of the system and the protocol version in use, when the system is powered up. A consumer can use this for monitoring health. If not received for 2 seconds, one can assume the system has failed or is powered down.

Field	Subfield	SubSubfield	Type	Description
protocol				Represents information about the protocol
	version		uint32	Protocol version in use
system				Represents the status of the system sending unsolicited messages
	id		string	A vendor specific identifier, along with a device specific identifier. Used to uniquely identify the source of the data stream

	error			A list of errors. If no error is present, the subfield is not added
		id	uint32	Identifier of the error, exhaustive list can be found in FTD-012.
		sev	uint32	Severity of the error 0: No error, shall not be used. 1: Error is for information only. 2: Error warns of a degraded mode of operation. 3: Error warns of a fatal failure.
		descr	string	Human readable description of an error

Example


```
{
  "heartbeat": {
    "protocol": {
      "version": 1
    },
    "system": {
      "error": [
        {
          "descr": "Low power supply",
          "id": 341,
          "sev": 2
        },
        {
          "descr": "Firmware about to expire",
          "id": 43,
          "sev": 1
        }
      ],
      "id": "FLAPFC11E-0005431"
    }
  }
}
```


Note that in this example, the list of errors contains two items.

2.2 Navigation

Represents the navigation state of our own aircraft.

Sent every second, if a GNSS solution is known. No navigation message means no GNSS position/time is available.

Field	Subfield	SubSubfield	Type	Description
pos				Represents a position, using WGS-84 datum, using the ellipsoid as reference for altitude. You can expect latitude/longitude/altitude to be all populated if one is populated. The barometric altitude is optional.
	lat		double	Latitude [deg]
	lon		double	Longitude [deg]
	alt		sint32	Altitude above the ellipsoid [m]
	baro		sint32	Altitude computed from barometric pressure, using the standard atmosphere model [m]
mov				Represents a movement. Note that a target might have a speed of 0 but can be flying (e.g. drones or helicopters)!
	speed		float	Ground speed [m/s]
	gnd		bool	Whether the target is on the ground or not.
	climb		float	Climb rate [m/s]
	turn		float	Turn rate [deg/s]
	track		float	Track [deg]
acc				Represents an accuracy of the GNSS solution.
	horizontal		float	50% confidence interval the 2D position is within the given radius [m]
	vertical		float	50% confidence interval the vertical position is within the given range [m]
	speed		float	50% confidence interval the speed is within the given range [m/s]
time			double	Time [s] the navigation was acquired

Example

```
{
  "navigation": {
    "acc": {
      "horizontal": 14.5,
      "speed": 2.1,
      "vertical": 9.2
    },
    "mov": {
      "climb": 5.2,
      "speed": 4.2,
      "track": 90.2,
      "turn": -2.4
    },
    "pos": {
      "alt": 491,
      "baro": 481,
      "lat": 47.214272,
      "lon": 8.4666112
    },
    "time": 854065654.2
  }
}
```

2.3 Traffic

Represents a target with known position and movement. For example, target whose source is ADS-B, FLARM, ADS-R, TIS-B. The traffic message is output at most once between heartbeat messages only for a given identifier by giving priority to the most accurate source. The same mechanism applies to undirected, i.e. for a given identifier, either a traffic report or an undirected report is sent.

Field	Subfield	SubSubfield	Type	Description
id				Represents a unique identifier for a target. Both the type and its value must be used to create a unique identifier. i.e. two targets, one having a random identifier and the other one having a FLARM identifier, could have the same identifier's value.
	random		uint32	Semi-stable, random identifier not directly attributable to traffic source

	flarm		uint32	FLARM identifier, fixed per sender
	icao		uint32	ICAO aircraft (transponder) address used as identifier
	gen		uint32	Identifier was generated to identify a transponder Mode A/C only equipped (or not equipped with any Electronic Conspicuity device) target
	ext		string	The identifier is an extended identifier.
src				Represents a source informing through what type of system a target was acquired. Only one of the subfield applies.
	flarm			Represents a target received through the FLARM system
		stealth	bool	Stealth mode, a parameter mostly used in glider competitions. See FTD-014, parameter “priv” for more details.
		noTrack	bool	NoTrack mode. a parameter expressing intent to stay private. See FTD-014, parameter “priv” for more details.
	ads_b			Represents a target received through the ADS-B system.
	ads_b_nt			Represents a target received through the ADS-B system, non-transponder based.
	ads_r			Represents a target received through a ground-based system on UAT and rebroadcasted (ADS-R).
	tis_b			Represents a target received through a ground-based system on SSR and rebroadcasted (TIS-B).
	remote_id			Represents a target received through Wifi NAN frames (Remote ID).
	uat			Represents a target received through the UAT system.
type			Target Type	See table below.
pos				Represents a position, using WGS-84 datum, using the ellipsoid as reference for altitude. You can expect latitude/longitude/altitude to be all populated if one is populated. The barometric altitude is optional.

	lat		double	Latitude [deg]
	lon		double	Longitude [deg]
	alt		sint32	Altitude above the ellipsoid [m]
	baro		sint32	Altitude computed from barometric pressure, using the standard atmosphere model [m]
mov				Represents a movement. Note that a target might have a speed of 0 but can be flying (e.g. drones or helicopters)!
	speed		float	Ground speed [m/s]
	gnd		bool	Whether the target is on the ground or not.
	climb		float	Climb rate [m/s]
	turn		float	Turn rate [deg/s]
	track		float	Track [deg]
time			double	Time [s] the target was received on the RF channel and a state update could be computed.
rec				List of meta-information about the reception of the target
	rad			Represents meta information about the reception of traffic or undirected report on a radio. Applies if the radio receiver supports signal level measurement.
		dBm	float	Signal level [dBm]
	gnd			Represents meta information about the reception of traffic or undirected report by a ground station (e.g. Skylens).
		dBm	float	Signal level [dBm]
		id	string	Identifier of the receiving station
		dev	float	Frequency deviation [Hz]

TargetType

Value	Description
0	Reserved
1	Glider, motor glider (turbo, self-launch, jet), TMG
2	Tow plane, tug plane
3	Helicopter, gyrocopter, rotorcraft
4	Skydiver, parachute
5	Drop plane for skydivers
6	Hang glider (hard)

7	Paraglider (soft)
8	Aircraft with reciprocating engine(s)
9	Aircraft with jet, turboprop engine(s)
10	Unknown
11	Balloon (hot, gas, weather, static)
12	Airship, blimp, zeppelin
13	Unmanned aerial vehicle (UAV, RPAS, drone)
14	Reserved
15	Static obstacle

Example

```
{
  "traffic": {
    "id": {
      "random": 123
    },
    "mov": {
      "climb": 5.2,
      "speed": 4.2,
      "track": 90.2,
      "turn": -2.4
    },
    "pos": {
      "alt": 491,
      "baro": 481,
      "lat": 47.214272,
      "lon": 8.4666112
    },
    "rec": [
      {
        "rad": {
          "dBm": -64.9
        }
      }
    ],
    "src": {
      "flarm": {}
    },
    "time": 854065654.2,
    "type": 8
  }
}
```

2.4 Undirected

Represents a target without a known position and movement but with a known approximate distance and known altitude. For example, a target whose source is Mode A/C, Mode-S. The undirected message is output once per time slot only for a given identifier by giving priority to the most accurate source. The same mechanism applies to traffic, i.e. for a given identifier, either a traffic report or an undirected report is sent.

Field	Subfield	SubSubfield	Type	Description
id				Represents a unique identifier for a target. Both the type and its value must be used to create a unique identifier. i.e. two targets, one having a random identifier and the other one having a FLARM identifier, could have the same identifier's value
	random		uint32	Semi-stable, random identifier not directly attributable to traffic source
	flarm		uint32	FLARM identifier, fixed per sender
	icao		uint32	ICAO aircraft (transponder) address used as identifier
	gen		uint32	Identifier was generated to identify a transponder Mode A/C only equipped (or not equipped with any Electronic Conspicuity device) target
	ext		string	The identifier is an extended identifier.
src				Represents a source informing through what type of system a target was acquired. Only one of the subfield applies.
	mode_s			Represents a target received through a Mode-S transponder.
	mode_c			Represents a target received through a Mode A/C transponder.
baro			int32	The altitude computed from barometric pressure, using the standard atmosphere model [m]
dist			uint32	Approximate distance, inferred from heuristics depending on signal level [m].
time			double	Time [s] the target was received on the RF channel.
rec				List of meta-information about the reception of the target

	rad			Represents meta information about the reception of traffic or undirected report on a radio. Applies if the radio receiver supports signal level measurement.
		dBm	float	Signal level [dBm]

Example


```
{
  "undirected": {
    "baro": 541,
    "id": {
      "icao": 854045
    },
    "rec": [
      {
        "rad": {
          "dBm": -64.9
        }
      }
    ],
    "src": {
      "modeS": {}
    },
    "time": 854065654.2,
    "dist": 456
  }
}
```


2.5 Info

Represent additional information about a target. Message is sent typically once a minute if additional information is known (depending on the type of target).

Field	Subfield	SubSubfield	Type	Description
id				Represents a unique identifier for a target. Both the type and its value must be used to create a unique identifier. i.e. two targets, one having a random identifier and the other one having a FLARM identifier,

				could have the same identifier's value
	random		uint32	Semi-stable, random identifier not directly attributable to traffic source
	flarm		uint32	FLARM identifier, fixed per sender
	icao		uint32	ICAO aircraft (transponder) address used as identifier
	gen		uint32	Identifier was generated to identify a transponder Mode A/C only equipped (or not equipped with any Electronic Conspicuity device) target
	ext		string	The identifier is an extended identifier.
partNumber			string	Part number (FLARM)
swVersion			string	Software version (FLARM)
flightId			string	Flight identifier (ADS-B)

Example

```
{
  "info": {
    "flightId": "LX9832",
    "id": {
      "flarm": 7439845
    },
    "partNumber": "C11E",
    "swVersion": "7.04"
  }
}
```

Appendix A – Protocol Buffer definition

In the listing below, the protocol definition is given in Protocol Buffers[3] syntax. Language specific (python, c++, Java, c) files can be generated with "protoc", the Protocol Buffers compiler.

This is one possible way to implement a JSON parser on the customer side since a standard mapping between a protocol buffer message and its JSON representation exists[4].

```
// FLARM_JSON.proto
syntax = "proto3";

package flarm;

message Identifier {
  oneof type {
    uint32 random = 1;
    uint32 flarm = 2;
    uint32 icao = 3;
    uint32 gen = 4;
    string ext = 5;
  }
}

message SourceFlarm {
  bool stealth = 2;
  bool noTrack = 3;
}

message SourceTransponder {
}

message SourceGroundStation {
}

message SourceWifi {
}

message Source {
  oneof type {
    SourceFlarm flarm = 1;
```

[3] https://developers.google.com/protocol-buffers
[4] https://developers.google.com/protocol-buffers/docs/proto3#json

```
    SourceTransponder mode_c = 2;
    SourceTransponder mode_s = 3;
    SourceTransponder ads_b = 4;
    SourceTransponder ads_b_nt = 5;
    SourceGroundStation ads_r = 6;
    SourceGroundStation tis_b = 7;
    SourceWifi remote_id = 8;
    SourceTransponder uat = 9;
  }
}

message ReceptionRadio {
  float dBm = 1;
}

message ReceptionGroundStation {
  float dBm = 1;
  string id = 2;
  float dev = 3;
}

message Reception {
  oneof recv {
    ReceptionRadio rad = 1;
    ReceptionGroundStation gnd = 2;
  }
}

message Position {
  double lat = 1;
  double lon = 2;
  sint32 alt = 3;
  sint32 baro = 4;
}

message Movement {
  float speed = 1;
  bool gnd = 2;
  float climb = 3;
  float turn = 4;
  float track = 5;
}

enum TargetType {
  RESERVED = 0;
```

```
  GLIDER = 1;
  TOWPLANE = 2;
  HELICOPTER = 3;
  PARACHUTE = 4;
  DROPPLANE = 5;
  FIXED_HG = 6;
  SOFT_HG = 7;
  ENGINE = 8;
  JET = 9;
  UNKNOWN = 10;
  BALLOON = 11;
  AIRSHIP = 12;
  UAV = 13;
  RESERVED_2 = 14;
  STATIC = 15;
}

message Traffic {
  Identifier id = 1;
  Source src = 2;
  TargetType type = 3;
  Position pos = 4;
  Movement mov = 6;
  double time = 7;
  repeated Reception rec = 8;
}

message Undirected {
  Identifier id = 1;
  Source src = 2;
  TargetType type = 3;
  int32 baro = 5;
  uint32 dist = 6;
  double time = 7;
  repeated Reception rec = 8;
}

message Accuracy {
  float horizontal = 1;
  float vertical = 2;
  float speed = 3;
}

message Navigation {
  Position pos = 1;
```

```
  Movement mov = 3;
  Accuracy acc = 4;
  double time = 5;
}

message Info {
  Identifier id = 1;
  string partNumber = 2;
  string swVersion = 3;
  string flightId = 5;
}

message Protocol {
  uint32 version = 1;
}

enum ErrorSeverity {
  NONE = 0;
  INFO = 1;
  WARN = 2;
  FATAL = 3;
}

message Error {
  uint32 id = 1;
  ErrorSeverity sev = 2;
  string descr = 3;
}

message System {
  repeated Error error = 1;
  string id = 2;
}

message Heartbeat {
  Protocol protocol = 1;
  System system = 2;
}

message Payload {
  oneof unsolicited {
    Traffic traffic = 1;
    Undirected undirected = 2;
    Navigation navigation = 3;
    Info info = 4;
```

```
    Heartbeat heartbeat = 5;
  }
}
```