RPAS Exploitation Data Centre

Interface Control Document for RPAS Service providers

Issue:	0. 13
Date:	2017,Oct.16

List of tables and figures

List of tables:

No table of figures entries found.

List of figures:

Applicable documents

N/A

Reference documents

RD 1 STANAG 4609 JAIS (Ed. 3) – NATO DIGITAL MOTION IMAGERIE STANDARD
NSA/1117(2009)-JA/S/4609

RD 2 UAS Datalink Local Set
MISB ST 0601.11

RD 3 Motion Imagery Sensor Minimum Metadata Set
MISB ST 0902.6

RD 4 Video Moving Target Indicator and Track Metadata
MISB ST 0903.4

RD 5 MPEG-2 Transport Stream for Class 1/Class 2 Motion Imagery, Audio and Metadata
MISB ST 1402.2

RD 6 Real-Time Protocol for Motion Imagery and Metadata
MISB ST 0804.4

RD 7 Constructing a MISP Compliant File/Stream
MISB TRM 0909.4

RD 8 Motion Imagery Identification System (MIIS) Core Identifier
MISB ST 1204.2

RD 9 SMPTE STANDARD – Data Encoding Protocol Using Key-Length-Value
SMTPE 336M-2007

RD 10 RPAS-DC Web User Interface User Manual
CLS-RPASDC-17-006

List of Contents

1. Object of the document

EMSA is currently implementing innovative maritime surveillance capabilities relying on RPAS fleets, on top of existing coastal and space based capabilities, in order to support EU Member States and EU agencies missions.

EMSA RPAS Data Center will be the unique place where all data from the RPAS missions will be stored, processed and made accessible to EMSA end users.

As described in Module 6 of the EMSA/OP/12/2016 and EMSA/OP/06/2016 framework contracts (FWCs), SPs shall implement system-to-system (S-2-S) interfaces to connect to the DC in order to be able to send the RPAS data collected during the missions.

This document describes the S-2-S interfaces to be put in place between the RPAS Data Centre (**DC**) and the RPAS Service providers (**SP**).

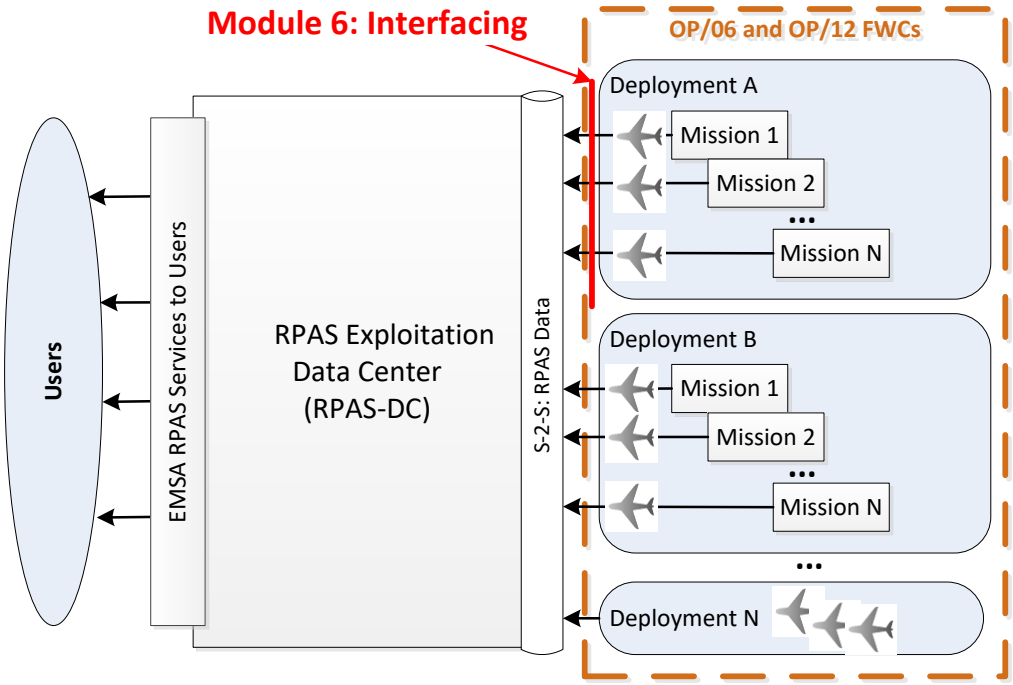


Figure 1 - Module 6 : Interfacing

2. Definitions

FWC	Framework Contracts
DC	RPAS Data Centre
GOP	Group Of Pictures
KLV	Key-Length-Value
LDS	Local Data Set
MISB	Motion Imagery Standards Board
OWL	Ontology Web Language
SP	RPAS Service Providers
URI	Unique Resource Identifier
WS	Web Service

3. RPAS Data Centre interfaces

3.1. General

The interfaces between the Service Providers and the RPAS-DC are depicted below.

In particular the "STANAG 4609 stream receiver" and the "RPAS DC Real Time HTTPS API" interfaces are real-time interfaces which need to be implemented by the SPs to send RPAS sensor data during live missions. They are specified in detail in this document in section 4 and section 5, respectively.

The remaining 3 interfaces, which do not require development, shall be used by the SPs to prepare the missions or to interact with the users during missions. The procedures on how to use/configure these interfaces are provided in annex in section 7.

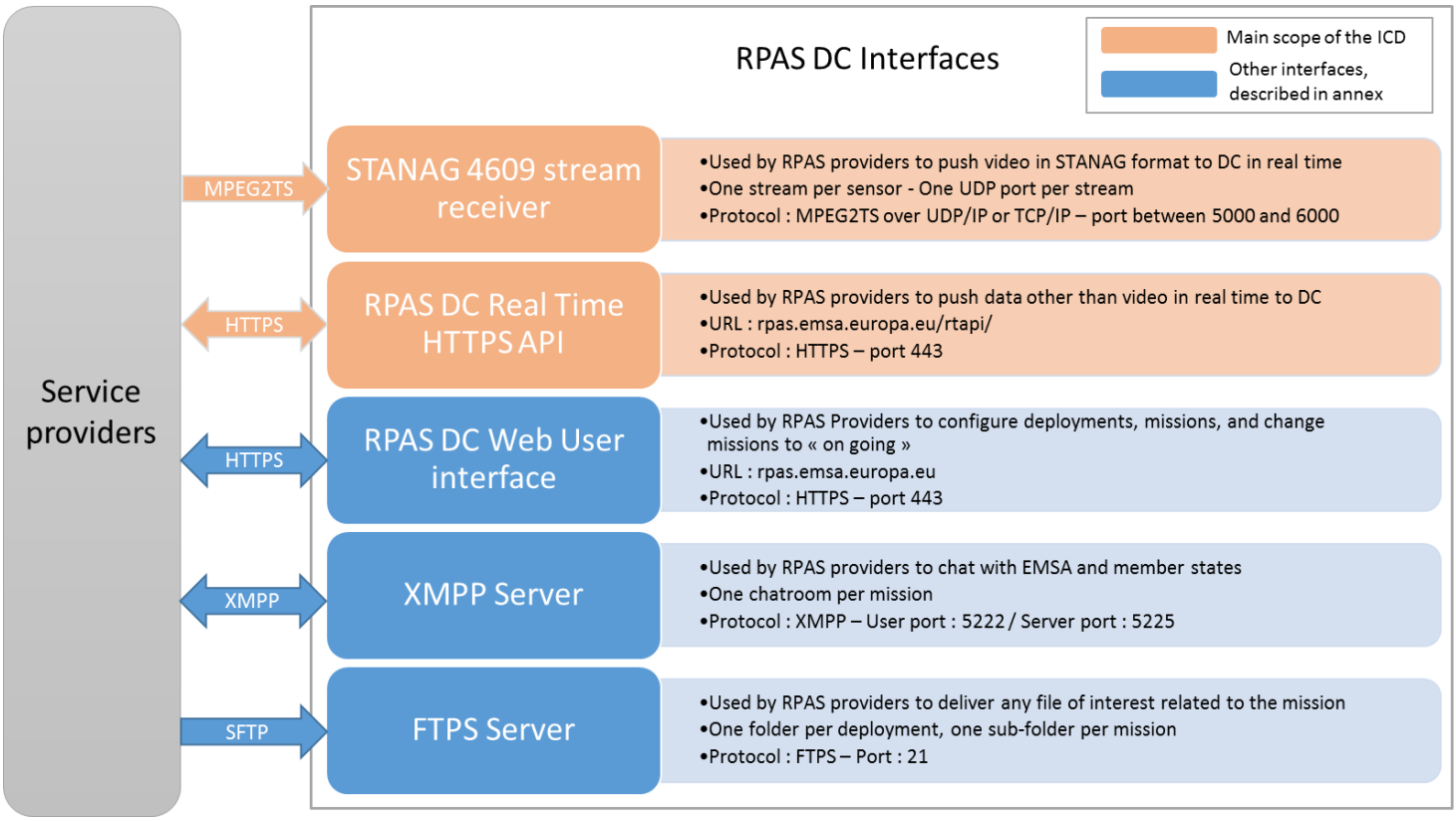


Figure 2 - RPAS DC interfaces

3.2. Time Reference Synchronization

On all interfaces, Universal coordinated time (UTC, also known as "Zulu"), clock signals shall be used as the universal time reference for NATO SMPTE 12M [26] time code systems, allowing systems using time code to accurately depict the actual Zulu time of day of motion imagery acquisition / collection / operations.

Furthermore, Global Positioning System time, corrected to UTC, is the standard for the source of time data. This time reference has to be used to timestamp all data sent to the RPAS Data Center (video, AIS, reports, etc…).

3.3. Security

3.3.1. Web Services Security

In order to guarantee the integrity of data exchange from RPAS-SC to external partners the SSL/TLS protocol is used. This SSL connection is set up in one way SSL: only the server is authenticated by sending to the client its own certificate (valid and signed by a Certificate Authority). The communication is encrypted in both directions.

Moreover, IP filtering (thanks to firewalling equipment) could be set up at DC level (based on IP source and port destination).

3.3.2. UDP connection Security

Video transmissions rely on UDP/IP for low latency connections, or TCP/IP as failback (see section 4.1 Transmission protocol for details). Due to the nature of this protocol, the foreseen method to secure this connexion is a mix of IPSEC or VPN. Both options are proposed to adapt the needs of every mission.

4. STANAG 4609 Stream receiver

All videos, associated metadata and RPAS specific metadata shall be sent following the STANAG 4609 DIGITAL MOTION IMAGERY STANDARD Ed. 3. This interface includes position and flight parameters, video sensors metadata, video streams (EO/IR cameras, Radar) and Target detection.

4.1. Transmission protocol

The protocol for video/metadata streaming is MPEG2TS over UDP/IP by default for lower latencies.

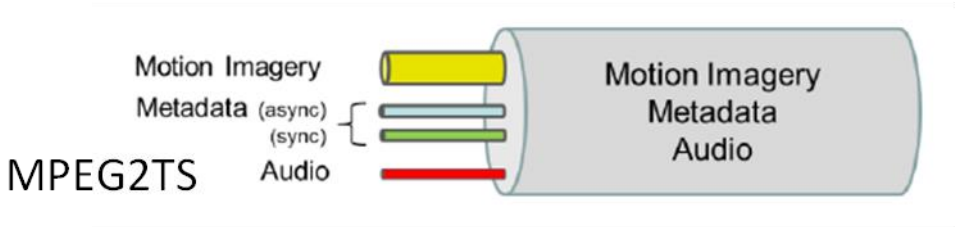


Figure 3 - MPEG2TS Protocol

It is mandatory to have one MPEG2TS stream per sensor. For example, if the RPAS is equipped with EO, IR, SAR and radar sensors, there will be one stream for each, so four MPEG2TS streams, each with a video channel (H.264 compression) and a KLVA channel. Depending on the RPAS configuration the RPAS-DC will provide a dedicated IP port for each MPEG2TS stream. CF annex in section 7.1 for mission configuration details.

Please also refer to the reconnection policy defined in section 8 which provides guidance on the management of connection errors.

The protocol for video/metadata streaming is MPEG2TS over UDP/IP by default for low latency or over TCP/IP if the internet connection between the SP and the DC is not good enough (high packet loss). RTP is also being investigated and this protocol might also be supported in the future.

In addition, it is recommended to send sync metadata. However the DC is also compatible with async metadata.

4.2. General Description of STANAG 4609 standard

The real-time protocol used in the DC as defined in STANAG 4609 standard (RD 1) includes guidance on uncompressed, compressed, and related motion imagery sampling structures, motion imagery time standards, motion imagery metadata standards, interconnections, and common language descriptions of motion imagery system parameters.

This ICD defines how the STANAG 4609 standard shall be used under the scope of the RPAS-DC, namely in the following aspects:

- Compression Systems (SP have to use H.264 as described in RD 1/ Annex C 2.2 with specific profile described in the following section)
- Metadata encoding that is based on the MISB standards

The next sections specify the compression system and the metadata defined in the MISB standard which shall be encoded by the RPAS Service Provider in the video stream.

4.3. H.264 compression system

To get the minimal latency as possible the video stream has to be encoded with the baseline profile to only use I and P frames. As explained in MISB RP0802.2 section 8.7 the B frame optimizes the bitrates of the stream but with high latency cost. We suggest also the SP to use all available options to reduce latency (preset=fast, tune=zerolatency in libav/ffmpeg for example). The final bitrates will be a bit bigger but between the LGCS and the RPAS DC that's not a problem, because the internet connection shall be of better quality.

In video coding, a **group of pictures (GOP) structure**, specifies the order in which intra- and inter-frames are arranged. The GOP is a collection of successive pictures within a coded video stream. To optimize the quality of streaming on the DC side the maximum duration of a group of pictures must be of 2 seconds.

Note: in software like ffmpeg this GOP parameter has to be converted in a number of frames.

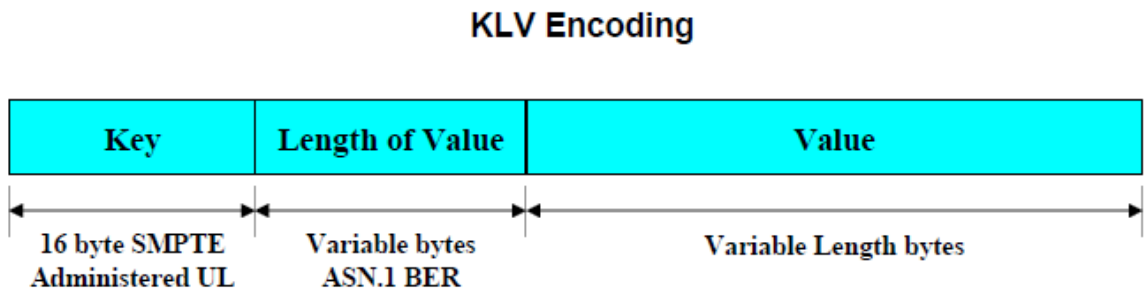
- To simulate a stream that should work with the DC you can download a STANAG compliant demo file provided by ESRI and use the following command line with ffmpeg tools.
- To download the Cheyenne ESRI sample file, go to: https://community.esri.com/docs/DOC-8527
- To download ffmpeg (v3 min), go to: https://ffmpeg.org/download.html

Example of FFMPEG command to use with the Cheyenne ESRI example video (the -g 20 sets the GOP parameter equal to the framerate, so each GOP will last 1 second):

fmpeg -re -i CheyenneVAhospital.mpeg4 -map 0:0 -map 0:2 -pix_fmt yuv420p -c:v libx264 -r:v 20 -g 20 -vf scale=800:-1 -profile:v baseline -level 3.0 -preset ultrafast -tune zerolatency -c:d copy -f mpegts {*udp or tcp link to server with port number}*

4.4. General Description of MISB standards

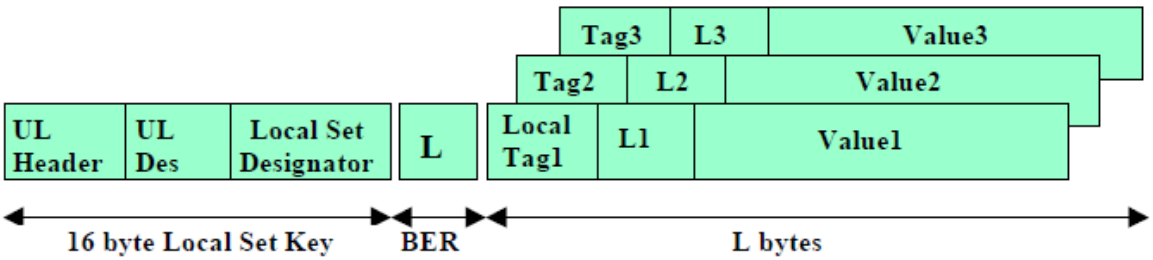
In STANAG 4609 the metadata embedded in the video stream have to be encoded with the MISB standard. This standard encodes data in KLV format (SMTPE 336) and defines a set of Universal Key and Local Set.



A Local Set is defined as a number of data items that are grouped to reduce the length of the Keys for each item within the Set. Data items may be in any order within the Local Set and may be present or absent.

The use of Keys for Local Set coding shall be defined by an accompanying Standard or RP including a Structure Designator and an accompanying Local Set Register including a version number.

The Key of a Local Set shall be 16 bytes in length. MISB define a set of keys and local-set to encode almost all type of RPAS data in space efficient format.



To encode metadata for the DC, the following MISB standards shall be used:

- ST 0601.11 : UAS Datalink Local Set (with all tags specified in ST 0902.6 Motion Imagery Sensor Minimum Metadata Set and others) to encode:
    - Aircraft health status
    - position and flight parameters
    - video sensors metadata
- ST 0903.4 : Video Moving Target Indicator and Track Metadata to encode automatic or manual detected target inside video frames

4.5. Metadata Set description

The following paragraph describes the different Metadata Sets of the MISB standard that shall be used per type of data transmitted.

The following types of data have been identified:

1. RPAS information and sensor metadata:
   - General information on RPAS: aircraft position and flight parameters
   - Video Sensor metadata information (type, footprint, status, current focal, etc.)

2. Sensors data / detection:
   - Targets detected on video streams (EO, IR, maritime and SAR radar)

4.5.1. RPAS information and sensor metadata

The metadata sets are fully described in the Standard 0601.11 MISB UAS Datalink Local Set and the mandatory subset is defined in the Standard 0902.6 MISB Minimum Metadata Set. The ST 0601.11 standard contains more than one way to encode specific parameters, a historical one and an extended one (with more precise storage). In the tables below, for parameters in this situation with two matching tags, the historical version is in orange and the new version is in green. The RPAS DC will support the two different tags for each parameter BUT the new ones are preferred if they are available on the SP LGCS.

The following tags shall be encoded in the video stream by the SP:

1. General information on RPAS

Tag	Tag Name	Max. Size (Bytes)	Units	Update freq.	Comments & Range
1	Checksum	2	Integer		UAS Datalink LDS > In ST 0902.6
2	UNIX Time Stamp	8	Integer	fast	UAS Datalink LDS > In ST 0902.6
3	Mission ID	127	String	10s	UAS Datalink LDS > In ST 0902.6
5	Platform Heading Angle	2	0-360 D°	fast	UAS Datalink LDS > In ST 0902.6
6	Platform Pitch Angle	2	+/- 20 D°	fast	UAS Datalink LDS > In ST 0902.6
90	Platform Pitch Angle (full)	4	+/- 90 D°	fast	UAS Datalink LDS > In ST 0902.6
7	Platform Roll Angle	2	+/- 50 D°	fast	UAS Datalink LDS > In ST 0902.6
91	Platform Roll Angle (full)	4	+/- 90 D°	fast	UAS Datalink LDS > In ST 0902.6
10	Platform Designation	127	String		UAS Datalink LDS > In ST 0902.6
35	Wind Direction	2	D°	5s	UAS Datalink LDS
36	Wind Speed	1	m/s	5s	UAS Datalink LDS
37	Static Pressure	2	mBAR	5s	UAS Datalink LDS
56	Platform Ground Speed	1	uint m/s	fast	UAS Datalink LDS

2. Video sensor metadata information (specific for each sensor)

Tag	Tag Name	Max. Size (Bytes)	Units	Update freq.	Comments & Range
11	Image Source Sensor	127	String		UAS Datalink LDS > In ST 0902.6
12	Image Coordinate System	127	String		UAS Datalink LDS > In ST 0902.6
13	Sensor Latitude	4	+/- 90 D°	fast	UAS Datalink LDS > In ST 0902.6
14	Sensor Longitude	4	+/- 180 D°	fast	UAS Datalink LDS > In ST 0902.6
15	Sensor True Altitude	2	-900 to 19000 m	fast	UAS Datalink LDS > In ST 0902.6
75	Sensor Ellipsoid Height	2	-900 to 19000 m	fast	UAS Datalink LDS > In ST 0902.6
16	Sensor Horizontal FoV	2	0 to 180 D°	fast	UAS Datalink LDS > In ST 0902.6
17	Sensor Vertical FoV	2	0 to 180 D°	fast	UAS Datalink LDS > In ST 0902.6
18	Sensor Rel. Az. Angle	4	0 to 360 D°	fast	UAS Datalink LDS > In ST 0902.6
19	Sensor Rel. El. Angle	4	+/- 180 D°	fast	UAS Datalink LDS > In ST 0902.6
20	Sensor Rel. Roll Angle	4	0 to 360 D°	fast	UAS Datalink LDS > In ST 0902.6
21	Slant Range	4	0 to 5000000 m	fast	UAS Datalink LDS > In ST 0902.6
22	Target Width	2	0 to 10000 m		UAS Datalink LDS > In ST 0902.6
96	Target Width Extended	3	0 to 1500000		UAS Datalink LDS > In ST 0902.6
23	Frame Center Lat.	4	+/- 90 D°2	fast	UAS Datalink LDS > In ST 0902.6
24	Frame Center Lon.	4	+/- 180 D°2	fast	UAS Datalink LDS > In ST 0902.6
25	Frame Center El.	2	-900 to 19000 m	fast	UAS Datalink LDS > In ST 0902.6
78	Frame Center Height Above Ellipsoid	2	-900 to 19000 m	fast	UAS Datalink LDS > In ST 0902.6
26	Offset Corner Lat Point 1	2	D°	fast	UAS Datalink LDS
27	Offset Corner Lon Point 1	2	D°	fast	UAS Datalink LDS
28	Offset Corner Lat Point 2	2	D°	fast	UAS Datalink LDS
29	Offset Corner Lon Point 2	2	D°	fast	UAS Datalink LDS
30	Offset Corner Lat Point 3	2	D°	fast	UAS Datalink LDS
31	Offset Corner Lon Point 3	2	D°	fast	UAS Datalink LDS
32	Offset Corner Lat Point 4	2	D°	fast	UAS Datalink LDS
33	Offset Corner Lon Point 4	2	D°	fast	UAS Datalink LDS
82	Corner Lat Point 1	4	D°	fast	UAS Datalink LDS
83	Corner Lon Point 1	4	D°	fast	UAS Datalink LDS

84	Corner Lat Point 2	4	D°	fast	UAS Datalink LDS
85	Corner Lon Point 2	4	D°	fast	UAS Datalink LDS
86	Corner Lat Point 3	4	D°	fast	UAS Datalink LDS
87	Corner Lon Point 3	4	D°	fast	UAS Datalink LDS
88	Corner Lat Point 4	4	D°	fast	UAS Datalink LDS
89	Corner Lon Point 4	4	D°	fast	UAS Datalink LDS
48/1	Security Classification	1	1	LUT	UAS Datalink LDS > In ST 0902.6 Default value: **RESTRICTED//** As per MISB ST0102.12
48/2	Classifying Country and Releasing Instructions Country Coding Method	1	1	LUT	UAS Datalink LDS > In ST 0902.6 Required value: **ISO-3166 Two Letter (0x01)** As per MISB ST0102.12
48/3	Classifying Country	6	6	String	UAS Datalink LDS > In ST 0902.6 Value: **Country where the deployment is based**
48/4	Security-SCI/SHI Information	40	20	String	UAS Datalink LDS > In ST 0902.6 **Not needed for this context**
48/5	Caveats	32	20	String	UAS Datalink LDS > In ST 0902.6 **Not needed for this context**
48/6	Releasing Instructions3	40	20	String	UAS Datalink LDS > In ST 0902.6 **Not needed for this context**
48/12	Object Country Coding Method	1	1	LUT	UAS Datalink LDS > In ST 0902.6 Required value: **ISO-3166 Two Letter (0x01)** As per MISB ST0102.12
48/13	Object Country Codes	40	20	String	UAS Datalink LDS > In ST 0902.6 Value: **Country where the deployment is based**
48/22	Security Metadata Version	2	2	Integer	UAS Datalink LDS > In ST 0902.6 Value: **12**
65	UAS Local Set Version	1			UAS Datalink LDS > In ST 0902.6 Value: **11**
94	Motion Imagery Core Identifier	50	None		UAS Datalink LDS > In ST 0902.6 *Not managed by the RPAS-DC for now.* Value: **Set unique value**

4.5.2. Sensors data / detection

The VMTI LS Elements part of the MISB standard permits to describe any information on targets (moving or not). This standard describes a complex structure of imbricated Local Data Set.

VMTI Local Set (LS) Structure		
KLV/TLV elements		
VTargetSeries		
VTarget $Pack_1$	• • •	VTarget $Pack_N$
VMask LS		VMask LS
VObject LS		VObject LS
VFeature LS		VFeature LS
VTracker LS		VTracker LS
VChip LS		VChip LS
VChipSeries[4]		VChipSeries

In our case, we will use the following tags of this standard:

Tag #	Tag Name	Max. Size (Bytes)	Units	Comments & Range
1	Checksum	2	Integer	ST 0903.4 VMTI LS Elements (Table 1)
2	UNIX Time Stamp	8	Integer	ST 0903.4 VMTI LS Elements (Table 1)
5	Total number of Targets Detected in the frame	4	Uint24	ST 0903.4 VMTI LS Elements (Table 1)
6	Number of Reported Targets	4	Uint24	ST 0903.4 VMTI LS Elements (Table 1)
8	Frame Width	3	Uint24	ST 0903.4 VMTI LS Elements (Table 1)
9	Frame Height	3	Uint24	ST 0903.4 VMTI LS Elements (Table 1)
10	VMTI Source sensor	127	UTF-8	ST 0903.4 VMTI LS Elements (Table 1)
101	VTargetSeries	X	Array	ST 0903.4 VMTI LS Elements (Table 1). It is a Variable Length Packs data type as define in section 5.4 of TRM 1006. It contains an array of VTargetPack elements

All the targets detected automatically by sensors emitting a video stream (e.g. video (EO/IR), maritime and SAR radar) will be described by using the VTargetPack for mobiles.

The description of this VTargetPack is done below.

Tag #	Tag Name	Max. Size Bytes	Units	Comments	Specification reference
NA	Target ID	3	Uint24 (BER-OID)	Target ID	ST 0903.4 VMTI LS Elements (Table 2)
1	Target Centroid Pixel Number	6	Uint48	Defines the position of the target within the Motion Imagery frame in pixels. Numbering commences from 1 denoting the Top Left pixel	ST 0903.4 VMTI LS Elements (Table 2)
2	Bounding box Top Left Pixel	6	Uint48	Defines the position of the top left corner of the target bounding box within the Motion Imagery frame in	ST 0903.4 VMTI LS Elements (Table 2)

	Number			pixels. Numbering commences from 1 denoting the Top Left pixel	
3	Bounding box Bottom right Pixel Number	6	Uint48	Defines the position of the bottom right corner of the target bounding box within the Motion Imagery frame in pixels. Numbering commences from 1 denoting the Top Left pixel	ST 0903.4 VMTI LS Elements (Table 2)
6	New detection / target history	2	Uint16	Number of previous times the same target has been detected	ST 0903.4 VMTI LS Elements (Table 2)
9	Target Intensity	3	Uint24	Dominant intensity of the target (mandatory for Maritime Radar / SAR sensor)	ST 0903.4 VMTI LS Elements (Table 2)
17	Target Location	22	Truncation Pack	Full location in lon/lat/h (include only Geospatial Coordinate triplet)	ST 0903.4 VMTI Location Structure (9.12.1)
102	Vobject LS	X	LocalSet	Use Vobject to define source of targets	ST 0903.4 VMTI LS Elements (Table 2)

The Vobject Local Set will be used as described below:

Tag #	Tag Name	Max. Size Bytes	Units	Comments	Specification reference
1	Ontology	127	UTF-8	URI to OWL (CF Annex 9 for details)	ST 0903.4 VMTI LS Elements (Table 4)
2	Ontology_Class	127	UTF-8	Type of target (string defined in OWL).	ST 0903.4 VMTI LS Elements (Table 4)

The source of the target will be contained in the main part of the VMTI, in the tag 10: “VMTI Source sensor”.

5. RPAS DC Real Time HTTPS API

Non video related data from the RPAS payload shall be sent through HTTPS using POST request. This interface includes several types of data, listed in section 5.2.

5.1. Description of the interface

The data is sent via HTTPS POST requests to the DC real-time ingestion API. CF https://en.wikipedia.org/wiki/POST_(HTTP) for more details (See RFC7231 for official specification: https://tools.ietf.org/html/rfc7231#section-4.3.3).

This API is described using the OpenAPI / Swagger specification (https://www.openapis.org / http://swagger.io/specification/). The API configuration file is provided in Annex (section 92). This specification can be visualized by opening index.html using Firefox. The index.html file is available in the rpas_api folder which is included in this ICD package.

Please also refer to the reconnection policy defined in section 8 which provides guidance on the management of connection errors.

5.2. Metadata send through HTTPS interfaces

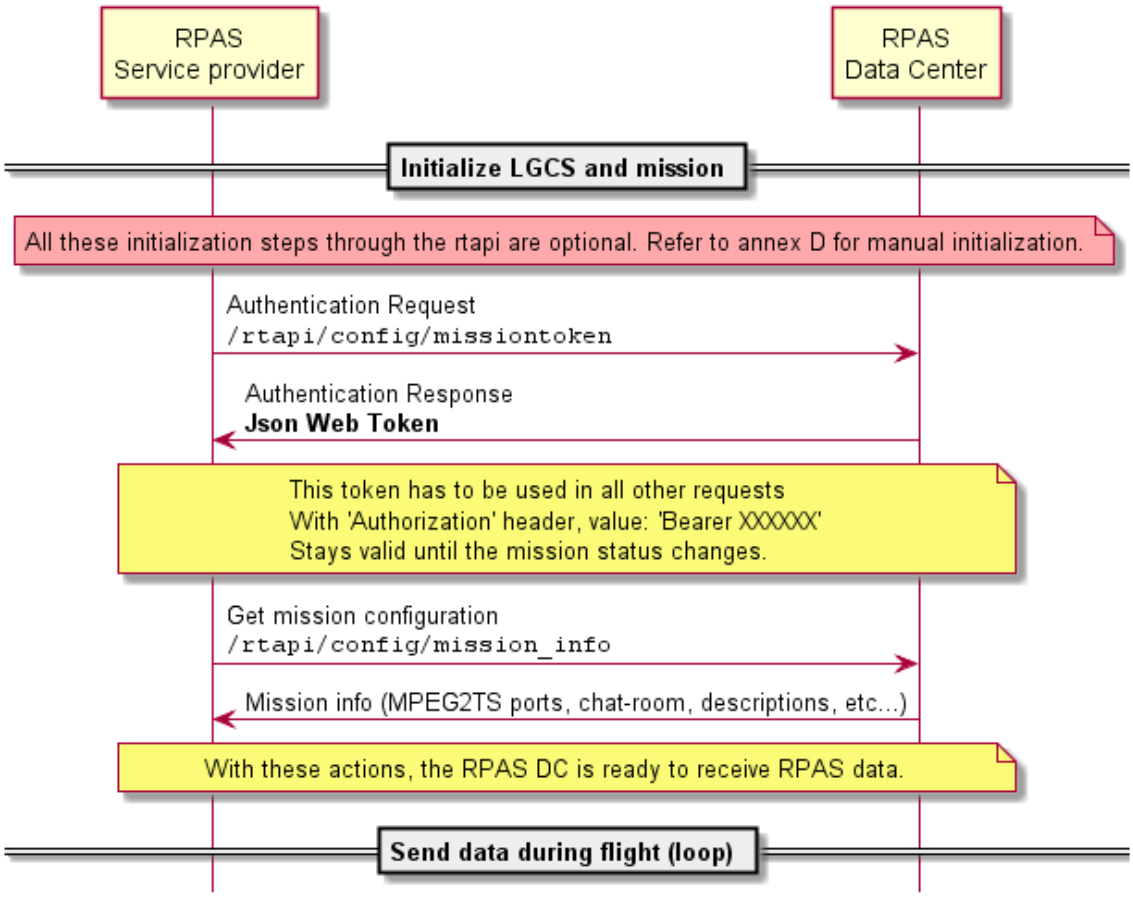
The following paragraph describes the different requests available in the RPAS DC real-time API allowing SPs to send RPAS mission data not managed by the STANAG 4609 interface (CF section 4).

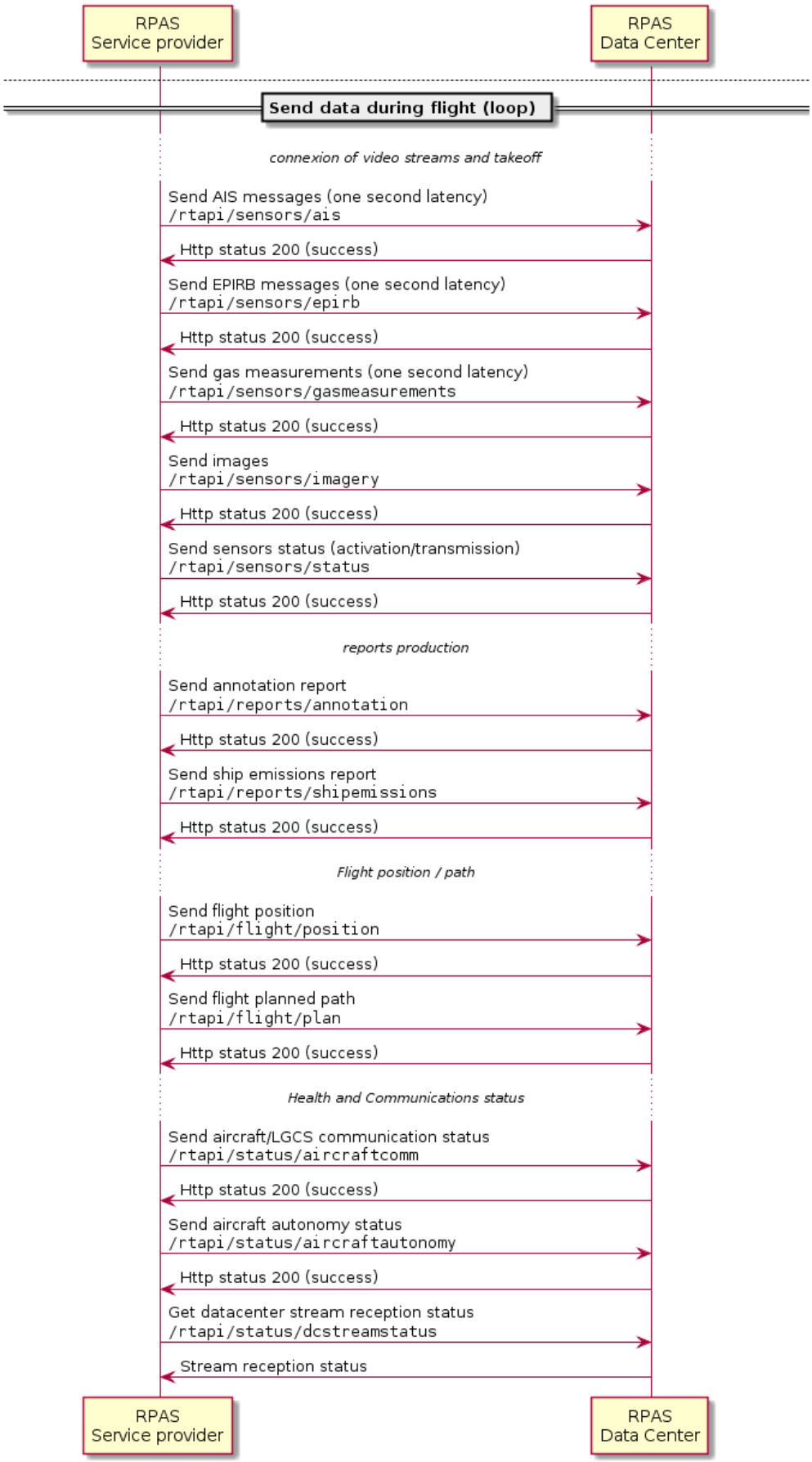
The types of data to be sent through HTTPS interface have been identified in the table below. Detailed information is provided in the API configuration file.

Type	Data	Web Service	Mandatory
Sensor	AIS messages (NMEAv4)	/sensors/ais	Yes
	EPIRB messages	/sensors/epirb	Yes
	Live Gas Measurements (CO2, SO2, NO, NO2, temperature, humidity, pressure, sensor position, timestamp) Only for emissions monitoring.	/sensors/gasmeasurements	Yes
	Images from specific sensors (EO/IR HR, SAR images)	/sensors/imagery	Yes
	Get status of RPAS sensors (LGCS to RPAS-DC)	/sensors/status	Yes
Reports	THETIS Ship Emissions Report (Consolidated Sulphur percentage and NOx measurements) Only for emissions monitoring.	/reports/shipemissions	Yes
	Manually detected AOI or POI (including Oil Spill Detection reports, area of interest, point of interest, detected target, mobile frequency for example)	/reports/annotation	Yes

Flight status	Real-time RPAS position (to overcome the fact that the video stream which already includes the position could be turned off during transit)	/flight/position	Yes
	Real-time updated flight plan (Array of waypoints which can be updated live by the SPs)	/flight/plan	Yes
RPAS/LGCS status	LGCS-RPAS active communication channel (RADIO/SATCOM) and signal quality For RPAS with SATCOM: Type could be SATCOM or RADIO, For RPAS without SATCOM: Type: RADIO. In all cases the quality shall be reported.	/status/aircraftcomm	Yes
	RPAS endurance information	/status/aircraftautonomy	Yes
	RPAS-DC data stream reception status	/status/dcstreamstatus	No
Configuration	Get valid token to push data during mission	/config/missiontoken	No
	Get mission information	/config/missionInfo	No

The following sequence diagrams describe the process to use the different web services.




RPAS
Service provider
RPAS
Data Center
Send data during flight (loop)
connexion of video streams and takeoff
Send AIS messages (one second latency)
/rtapi/sensors/ais
Http status 200 (success)
Send EPIRB messages (one second latency)
/rtapi/sensors/epirb
Http status 200 (success)
Send gas measurements (one second latency)
/rtapi/sensors/gasmeasurements
Http status 200 (success)
Send images
/rtapi/sensors/imagery
Http status 200 (success)
Send sensors status (activation/transmission)
/rtapi/sensors/status
Http status 200 (success)
reports production
Send annotation report
/rtapi/reports/annotation
Http status 200 (success)
Send ship emissions report
/rtapi/reports/shipemissions
Http status 200 (success)
Flight position / path
Send flight position
/rtapi/flight/position
Http status 200 (success)
Send flight planned path
/rtapi/flight/plan
Http status 200 (success)
Health and Communications status
Send aircraft/LGCS communication status
/rtapi/status/aircraftcomm
Http status 200 (success)
Send aircraft autonomy status
/rtapi/status/aircraftautonomy
Http status 200 (success)
Get datacenter stream reception status
/rtapi/status/dcstreamstatus
Stream reception status
RPAS
Service provider
RPAS
Data Center

6. Annex A - Operational Environments (System Context)

This section describes the production and pre-production end-points of RPAS-DC (hostname, ports, and services).

Interface	Hostname (production)	Hostname (pre-production)	Port	protocol
STANAG 4609 stream receiver	rpas.emsa.europa.eu	rpas-qo.cls.fr	5000 to 6000 dynamic assignment per mission (one stream, one ip port)	MPEG2TS through UPD/IP or TCP/IP
RPAS-DC Real Time HTTPS API	rpas.emsa.europa.eu	rpas-qo.cls.fr	443	HTTP over TLS
RPAS-DC Web User interface	rpas.emsa.europa.eu	rpas-qo.cls.fr	443	HTTP over TLS
XMPP Server	rpas.emsa.europa.eu	rpas-qo.cls.fr	5222	XMPP client
	rpas.emsa.europa.eu	rpas-qo.cls.fr	5269	XMPP server to server
FTPS Server	rpas.emsa.europa.eu	rpas-qo.cls.fr	21 (3000-3030 for dynamic data port)	FTP over TLS

7. Annex B – Introduction to other interfaces

As explained in section 3, the 3 interfaces below do not require development and shall be used by the SPs to prepare the missions or to interact with the users during missions.

7.1. RPAS-DC web user interface

The RPAS Data Centre Web user interface will be used by the SPs in cooperation with EMSA to setup deployments and missions in order to connect the SPs systems to the RPAS-DC during operations.

The RPAS-DC web user interface will allow end-users to follow the mission in real time: all data received from the service providers (via STANAG and RTAPI interfaces) will be displayed through this web interface, on a map and video windows.

More information on how to use this RPAS-DC web user interface is available in the RPAS-DC user manual (**RD 10**).

7.2. XMPP Server

The XMPP server allows to configure a chat connection between SP and EMSA end users.

Any XMPP client or server could be used to connect to our XMPP server. We recommend using Gajim, which has been fully tested during RPAS-DC acceptance tests.

To connect to the XMPP server, you need:

- A login/password on RPAS-DC web user interface
- To be authorized on at least one deployment

To configure a new XMPP account and join a chatroom:

Field	Value
Jabber ID	*[RPASDC login]*
Chat server	rpas.emsa.europa.eu
Conference server	conference.rpas.emsa.europa.eu
Room ID	*[EMSA_ID of the mission]*

7.3. FTPS Server

The FTPS server allows to exchange data (mission reports, HR images or video of object of interest, etc.) between SP and EMSA end users during or at the end of a mission.

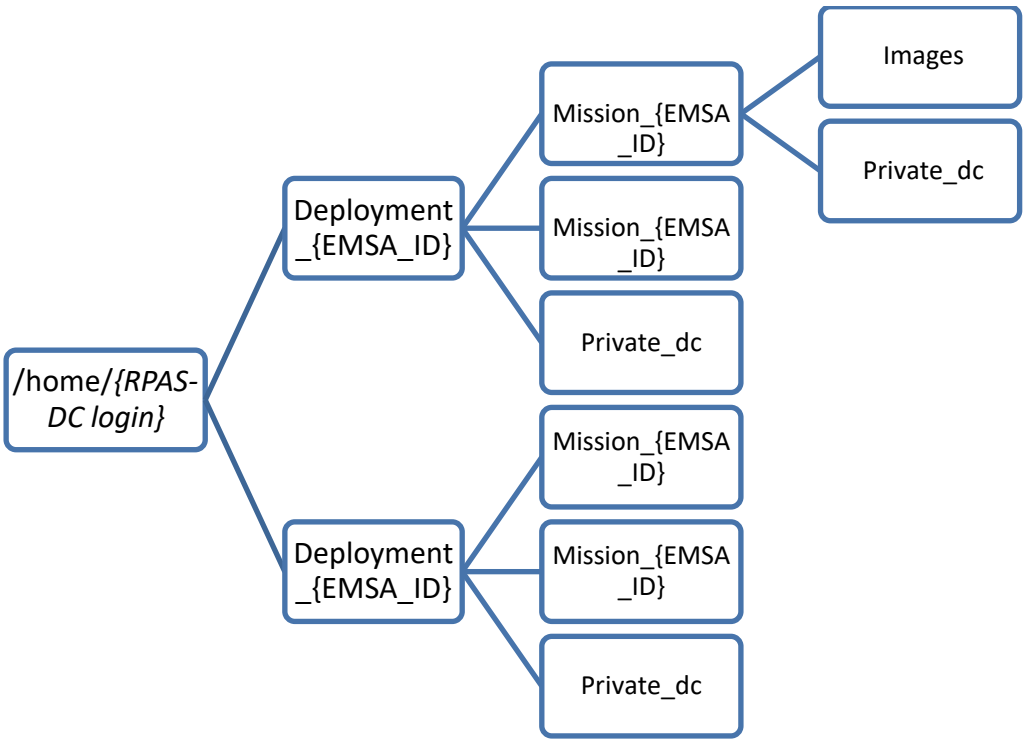
To connect to the FTPS server, we recommend using Filezilla.

To access to the FTPS:

Field	Value
Login	*[RPASDC login]*
FTP server	rpas.emsa.europa.eu
Port	21

To visualize the list of deployments you are authorized on, please change the repository to: /home/*[RPAS-DC login]*.

The structure of the FTP is the following:



- To publish any document related to a deployment, please put it in the "Deployment_{EMSA_ID} repository.
- To publish any document related to a mission, please put it in the "Mission_{EMSA_ID} repository.

All documents published on the FTP can be seen / edited / deleted by any user authorized on the deployment, except the "Private_dc" repository.

8. ANNEX C – Reconnection Policy

The following table describes the policy on the number of reconnection attempts for STANAG-4609 streams and for the different web-services.

Type	Data	Service	Reconnection attempts	Delay between attempts
Video stream	STANAG 4609 video stream	MPEG2TS over UDP/IP or TCP/IP	N/A if using UDP protocol If using TCP protocol, these parameters should be used: • Connexion timeout: 15s • Reconnect interval 1s to 15s (multiplier 1.5) SP should use the Web API /rtapi/dcstreamstatus to know if the DC is receiving the video stream.	N/A (udp)
Sensors	AIS messages	/sensors/ais	0 (wait for next message)	N/A
	EPIRB messages	/sensors/epirb	unlimited	20s
	Live Gas Measurements	/sensors/gasmeasurements	0 (wait for next message)	N/A
	Pictures	/sensors/imagery	15	20s
	Get status of all sensors	/sensors/status	At SPs discretion. Performance: not more than 1 request per 10 seconds.	N/A
Reports	THETIS Ship Emissions Report (Consolidated Sulphur percentage and NOx measurements)	/reports/shipemissions	40	20s
	Manual detected AOI or POI (including Oil spills)	/reports/annotation	15	20s
Flight status	RPAS position	/flight/position	0 (wait for next message)	N/A
	Mission flight plan	/flight/plan	15	20s
RPAS/LGCS status	LGCS-RPAS active communication channel (RF/SATCOM) and signal	/status/aircraftcomm	0 (wait for next message)	N/A

	quality			
	RPAS endurance information	/status/aircrafautonomy	0 (wait for next message)	N/A
	Get RPAS-DC data stream reception status	/status/dcstreamstatus	At SPs discretion. Performance: not more than 1 request per 10 seconds.	N/A
Configuration	Get valid token to push data during mission	/config/missiontoken	Idem as above. If needed (Mission manager should contact RPAS-DC support).	N/A
	Get mission information	/config/missionInfo	Idem as above.	N/A

9. ANNEX D – Ontology Web Language to be use with Video Motion Target Indicator standard

The **Ontology Web Language** (**OWL**) is a family of knowledge representation languages for authoring ontologies. Ontologies are a formal way to describe taxonomies and classification networks, essentially defining the structure of knowledge.

To define ontology to be used in the Video Motion Target Indicator standard, the RPAS DC will provide a super-set of classification over the SWEET from the JPL: https://sweet.jpl.nasa.gov/

SWEET ontologies are written in the OWL ontology language and are publicly available. SWEET 2.3 is highly modular with 6000 concepts in 200 separate ontologies.

This section will also contain a table with the list of classification to be used in VMTI VObject entry.

To be defined.

10. ANNEX E – HTTPS Real-time API OpenAPI configuration file

```
swagger: '2.0'
info:
  description: |
    Introduction
    ============

    This specification describes the different webservices of the RPAS DC
    real-time API which are made available to the SPs. This interface shall be
    used to send additional RPAS mission data complementing the STANAG 4609
    interface.

    ![sequence diagram about use of real-time api](./mission_sequence.png
    "Sequence diagram about use of real-time api")

  version: 0.13
  title: RPAS DataCenter HTTPS RealTime API
  contact:
    email: rdejoux@cls.fr
  license:
    name: Closed licence
host: rpas.emsa.europa.eu
basePath: /rtapi/v0
tags:
```

```
  - name: sensors
    description: |
      The "sensors/*" webservices shall be used to send to the RPAS DC real time
      data from non-video sensors installed on the RPAS : AIS, EPIRB, Live Gas
      measurements and images.
  - name: reports
    description: |
      The "reports/*" webservices shall be used to send annotations on
      information of interest identified by the operator, such as: THETIS Ship
      Emissions report, objects of interest of any kind (oil spill report, area
      of interest, point of interest, mobile frequency detection, target, etc.).
  - name: flight
    description: |
     The "flight/*" webservices shall be used to send data concerning the
     trajectory of the rpas: flight plan or last position received (position).
  - name: status
    description: |
      The "Status/*" webservices shall be used to exchange information
      concerning the status of the communication between the RPAS and the LGCS
      (aircraftcomm), and between the LGCS and the RPAS-DC (dcstreamstatus). In
      addition, it can be used to provide information on the remaining autonomy
      of the RPAS for the given mission (aircrafautonomy).
  - name: config
    description: |
      The "config/*" webservices may be used by the Service Provider to access all
information
      needed for the configuration of connection between the LGCS and the RPAS-DC.
schemes:
  - https
paths:
  /sensors/status:
    post:
      tags:
        - sensors
      summary: Current status of RPAS sensors (EO, IR, Radar, AIS, EPIRB, etc...)
      description: |
        The objective of this web service is to send information to the RPAS-DC
        on the On/Off and transmission status of each sensor installed on the
        RPAS. For instance, if a given sensor is Off it is expected that no data
        shall be available on the RPAS-DC and no action is needed by the user.
        In case the sensor is ON and no data is available, the user should
        report it.

        This message shall be sent at 1 minute rate.

        The list of sensors loaded on the RPAS for a given mission is configured
        in the RPAS-DC at the beginning of the mission and is accessible using
        /config/missioninfo and/or through the web interface.
      operationId: pushStatus
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
          name: sensors_status
          description: |
            Status of all sensors loaded in RPAS (activated (T/F),
            transmission ('Off', 'LGCS', 'RPAS_DC')).
          required: true
          schema:
            $ref: '#/definitions/SensorsStatus'
      responses:
        '200':
          $ref: '#/responses/success'
        '400':
```

```
          $ref: '#/responses/error'
        '401':
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /sensors/ais:
    post:
      tags:
        - sensors
      summary: Push newly received AIS messages
      description: |
        The objective of this web service is to send in real time the AIS
        messages received on board the RPAS during the mission. These messages
        have to be sent with a maximum latency of one second, and in the NMEAv4
        format with date in tag prefix (tag "c", value: unix timestamp of the
        reception of message). A set of messages can be sent in a single
        request.

        The NMEA AIS Message encoding is described in these standards:
          - [ITU-R M.1371-5](https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-
M.1371-5-201402-I!!PDF-E.pdf)
            (annex 8)
          - NMEA 0183 (section 7 for tags and annex C.2.1 for payload encoding
            in NMEA format)

        Another description of the AIVDM and NMEAv4 tags is available here (free
        access):
          - http://catb.org/gpsd/AIVDM.html
      operationId: pushAIS
      consumes:
        - application/x-text-nmea-aivdm
      produces:
        - application/json
      parameters:
        - in: body
          name: ais_messages
          description: |
            AIS messages received by RPAS, one message per line, each message
            MUST be timestamped with the 'c' NMEAv4 tag.
          required: true
          schema:
            type: string
            description: Plain text with one message per line
            example: |
              \c:1493210051*51\!AIVDM,1,1,,A,39NSFuU000wcQtJKcljqf5vF2000,0*22
              \c:1493210051*51\!AIVDM,1,1,,B,13IPeI0P00wcO8HKclVHKwvH2875,0*09
              \c:1493210051*51\!AIVDM,1,1,,A,13HOS7PP00wbvr0KWr@hSgvH0875,0*20
      responses:
        '200':
          $ref: '#/responses/successAIS'
        '400':
          $ref: '#/responses/error'
        '401':
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /sensors/epirb:
    post:
      tags:
        - sensors
      summary: Push newly received EPIRB messages
      description: |
```

```
        The objective of this web service is to send in real time any EPIRB
        alert received on the RPAS to the DC. These messages have to be sent
        with a maximum latency of one second, in decoded form and if possible
        with raw payload in base64 encoding. Raw payloads are described in the
        [Cospas-Sarsat C/S T.001 - Issue
        4](http://cospas-sarsat.int/images/stories/SystemDocs/Current/CS-T001-MAY-
2017.pdf)
        Annex A and B. The types of messages to be transmitted are: -
        User-Location protocol (section A3.3.4) - Standard-Location Protocols
        (section A3.3.5) - National Location Protocols (section A3.3.6)

        **The 121.5Mhz EPIRB beacon decoding will be described later.**
      operationId: pushEPIRB
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
          name: epirb_messages
          description: |
            Provide EPIRB message decoded in a json structure
          required: true
          schema:
            $ref: '#/definitions/EPIRBMessage'
      responses:
        '200':
          $ref: '#/responses/success'
        '400':
          $ref: '#/responses/error'
        '401':
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /sensors/gasmeasurements:
    post:
      tags:
        - sensors
      summary: Push a new set of gas measurements
      description: |
         For OP06/Lot2, a gas measurement sensor is installed onboard the RPAS.
         This webservice shall be used to continuously push CO2, SO2, NO2, NO
         measurements in realtime throughout the mission. Not to be confused
         with the THETIS Ship Emissions report provided through
         /reports/shipemissions which is sent once per ship measured.
      operationId: pushGasMeasurements
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
          name: gas_measurement
          description: |
            Set of gas measurements with precision time and sensor location
          required: true
          schema:
            $ref: '#/definitions/Gasmeasurement'
      responses:
        '200':
          $ref: '#/responses/success'
        '400':
          $ref: '#/responses/error'
        '401':
```

```
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /sensors/imagery:
    post:
      tags:
        - sensors
      summary: Push a new georeferenced image (GeoJPEG2000)
      description: |
         Images of interest (suspect vessel, oil spill, etc) taken during a
         mission and which the RPAS Service Provider deems relevant to be shared
         with a user, shall be sent through this web service.

         In order to be able to display pictures on the map at the RPAS DC
         level, the images must be sent in GEOJPEG2000 format.

         It is recommended to send the images with the highest available
         resolution/quality.
      consumes:
        - multipart/form-data
      produces:
        - application/json
      parameters:
        - in: formData
          description: |
            The image from sensor to be pushed (GeoJPEG2000).
            Image has to be pushed as
            [formData](https://www.w3.org/TR/html401/interact/forms.html#h-17.13.4)
          name: imagery
          type: file
          required: true
        - in: formData
          type: object
          name: imagery_metadata
          description: |
            Imagery metadata (timestamp, frame center, comment). Detail below in
            **Models → ImageryMetadata**
          required: true
          schema:
            $ref: '#/definitions/ImageryMetadata'
      responses:
        '200':
          $ref: '#/responses/success'
        '400':
          $ref: '#/responses/error'
        '401':
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /reports/shipemissions:
    post:
      tags:
        - reports
      summary: Push a new "THETIS Ship Emissions Report" (sulphur and NOx report)
      description: |
        For OP06/Lot2, in addition to sending continuous gas measurements (refer
        to /sensor/gasmeasurements), the provider shall send "Ship Emissions
        Reports" once per ship analyzed. This ship emissions report contains the
        sulphur percentage value in the fuel (%) and NOx emission (g/kWh). These
        consolidated values are calculated from the continuous measurements
        taken over the vessel plume. The reports shall be sent as soon as they
        are available during the mission.
```

```
    operationId: pushEmissionsReport
    consumes:
      - multipart/form-data
    produces:
      - application/json
    parameters:
      - in: formData
        type: object
        name: shipemissions_report
        description: |
          The emissions report in json format
        required: true
        schema:
          $ref: '#/definitions/ShipEmissionsReport'
      - in: formData
        type: file
        name: pdf_report
        description: |
          PDF version of the Ship Emissions Reports. Detail below in
          **Models → ShipEmissionsReport**
        required: false

    responses:
      '200':
        $ref: '#/responses/success'
      '400':
        $ref: '#/responses/error'
      '401':
        $ref: '#/responses/auth_error'
      '403':
        $ref: '#/responses/access_error'
    security:
      - jwt_auth: []
/reports/annotation:
  post:
    tags:
      - reports
    summary: Push a new annotation report
    description: |
      The objective of this web service is to provide means to the Service
      Provider to share in a generic way annotations (point, polygons etc..)
      with the end users. The annotations sent through this webservice are
      displayed at the level of the RPAS-DC web interface. The Service
      Provider shall use this web service to report on any object (point) or
      geographic area (polygon) of interest, such as OilSpills, Mobile
      Frequency Detection, areas of interest, target of interest, location,
      etc. The annotations, in addition to provide geometry, are timestamped
      and may include comments and/or additional data which can be appended
      depending on the type of annotation reported.

      Each annotation has an "id". The service providers can use this field to
      update previous annotations which have changed overtime by sending an
      annotation message with the same id. This will replace the previous
      annotation at the level of the RPAS-DC.
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
      - in: body
        name: annotation_report
        description: |
          The annotation report in json format
        required: true
        schema:
```

```
            $ref: '#/definitions/AnnotationReport'
      responses:
        '200':
          $ref: '#/responses/success'
        '400':
          $ref: '#/responses/error'
        '401':
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /flight/position:
    post:
      tags:
        - flight
      summary: Push a new rpas position report
      description: |
        In order to have the track of the RPAS continuously displayed at the
        level of the RPAS-DC, the SP shall send via this web service the
        position of the RPAS at the rate of once per second.

        The RPAS position is also contained in the KLV of the STANAG4609 video,
        but in case the video stream is down/off (which could be the case for
        several hours during transit) the position of the RPAS would not be
        sent. This webservice was created to overcome this issue.

        This message shall include the following information on the RPAS
        position:
          - longitude, latitude and altitude (WGS84 / EPSG 4326)
          - Ground speed of the airborne platform in m/s (as tag 56 in MISB
ST0601.11)
          - Aircraft heading angle, relative between longitudinal axis
            and True North measured in the horizontal plane in degrees
            (as tag 5 in MISB ST0601.11)
        The rpas position has to be provided through this interface throughout
        the whole flight.
      operationId: pushAircraftPosition
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
          name: aircraft_position
          description: |
            The aircraft position in json format
          required: true
          schema:
            $ref: '#/definitions/AircraftPosition'
      responses:
        '200':
          $ref: '#/responses/success'
        '400':
          $ref: '#/responses/error'
        '401':
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /flight/plan:
    post:
      tags:
        - flight
      summary: Push a new rpas flight plan
```

```
      description: |
        The flight plan shall be composed of an array of waypoints which are
        ordered sequentially.

        Each waypoint is defined with the following parameters:
          1. ID (example: 01, 02, 03, etc...); numbered sequentially.
          2. Description (just a string, example: Loiter, Return Home, ...)
          3. Coordinates (lat, lon, alt)

        In addition an activated property allows to define the current active
        waypoint, if any. The current active waypoint can be a waypoint part of
        the flight plan, an additional discrete waypoint not part of the flight
        plan or none, in case there is no active waypoint.

        Possible values for the property:
          1. an ID of one of waypoints part of the flight plan
          2. a discrete navigation point (lon/lat/alt) in a list
          3. `null` if no waypoint is active. That means the provider is not
             navigating using waypoints method. Other possible means are "hold
             flight altitude, speed and heading" or "manual" which is not
             managed by the RPAS-DC.
      operationId: pushAircraftPlannedPath
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
          name: aircraft_planned_path
          description: |
            The aircraft planned path in json format
          required: true
          schema:
            $ref: '#/definitions/AircraftPlan'
      responses:
        '200':
          $ref: '#/responses/success'
        '400':
          $ref: '#/responses/error'
        '401':
          $ref: '#/responses/auth_error'
        '403':
          $ref: '#/responses/access_error'
      security:
        - jwt_auth: []
  /status/aircraftcomm:
    post:
      tags:
        - status
      summary: Push aircraft communication status
      description: |
        The interest of this WS is to know from end user side what is the
        communication channel used between RPAS and LGCS (RADIO or SATCOM) and
        what is the signal quality of this communication channel. It will enable
        end users to better understand why the video stream is degraded for
        example.

        It should be updated every minute, and as soon as the communication
        channel changes from RADIO to SATCOM for instance.
      operationId: pushAircrafComm
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
```

```
        name: aircraft_comm
        description: |
          The aircraft communication status in json format.
        required: true
        schema:
          $ref: '#/definitions/AircraftComm'
    responses:
      '200':
        $ref: '#/responses/success'
      '400':
        $ref: '#/responses/error'
      '401':
        $ref: '#/responses/auth_error'
      '403':
        $ref: '#/responses/access_error'
    security:
      - jwt_auth: []
/status/aircraftautonomy:
  post:
    tags:
      - status
    summary: Push a aircraft autonomy status
    description: |
      This web service allows the end user to have information on the autonomy
      of the RPAS. Depending on the autonomy displayed, the user will know if
      the scope of the mission can be extended or vice-versa.

      The message includes the estimated remaining flight time and distance.
      It shall be sent at a rate of 1 message per minute.
    operationId: pushAircrafAutonomy
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
      - in: body
        name: aircraft_autonomy
        description: |
          The aircraft autonomy in json format
        required: true
        schema:
          $ref: '#/definitions/AircraftAutonomy'
    responses:
      '200':
        $ref: '#/responses/success'
      '400':
        $ref: '#/responses/error'
      '401':
        $ref: '#/responses/auth_error'
      '403':
        $ref: '#/responses/access_error'
    security:
      - jwt_auth: []
/status/dcstreamstatus:
  get:
    tags:
      - status
    summary: Get datacenter stream reception status
    description: |
      As the STANAG 4609 streams are sent over UDP protocol, there is no
      return status to check if the streams are correctly received. The object
      of this web service is to enable SP to check if these streams are well
      received by the RPAS DC. There is one status per stream sensor
      configured for the current mission.
    operationId: getDCStreamStatus
    produces:
```

```
      - application/json
    responses:
      '200':
        $ref: '#/definitions/DCStreamStatus'
    security:
      - jwt_auth: []
/config/missiontoken:
  post:
    tags:
      - config
    summary: Get JWT token for the current Ongoing mission
    description: |
      In order to secure the connection to the different WS, an authentication
      token is needed for all web services. The token is specific per LGCS and
      per on-going mission. Therefore, the token expires as soon as the
      mission status changes to "finished" or "aborted".

      To get the token, the SP shall know the LGCS username/password.

      If there is no "on-going" mission declared for the LGCS identified by
      username/password, an error will be raised (HTTP Status 400 with error
      message). Note that this token can be retrieved directly from the Web
      User interface of RPAS DC.
    operationId: pushAircraftPlannedPath
    consumes:
      - application/x-www-form-urlencoded
    produces:
      - application/json
    parameters:
      - in: formData
        description: |
          RPAS Service LGCS specific user name
        name: lgcs_username
        type: string
        required: true
      - in: formData
        description: |
          RPAS Service LGCS specific password
        name: password
        type: string
        required: true
    responses:
      '200':
        $ref: '#/responses/token'
      '400':
        $ref: '#/responses/error'
      '401':
        $ref: '#/responses/auth_error'
/config/missionInfo:
  get:
    tags:
      - config
    summary: Get mission information in a json format
    description: |
      The interest of this web service is to get all information concerning
      the current on-going mission directly, without connecting to the RPASDC
      Web user interface. It can be used to get the chatroom name, the link to
      mission ftp folder to release files, or the RPAS configuration status,
      to check which sensors are declared for that mission.
    operationId: getMissionInfo
    produces:
      - application/json
    responses:
      '200':
        $ref: '#/definitions/MissionInfo'
    security:
```

```
        - jwt_auth: []

securityDefinitions:
  jwt_auth:
    type: apiKey
    name: Authorization
    description: |
      Use Authorization header with content like: 'Bearer XXXXXX (valid mission
      Json Web Token)'
    in: header

responses:
  successAIS:
    description: |
      Successful request. Body will contain a JSON with detail of ingest
      DataCenter
    schema:
      $ref: '#/definitions/HttpSuccessAIS'
  success:
    description: |
      Successful request. Body will contain a JSON with detail of ingest
      DataCenter
    schema:
      $ref: '#/definitions/HttpSuccess'
  token:
    description: |
      Successful request. Body will contain a JSON with a valid Json Web Token
    schema:
      $ref: '#/definitions/TokenSuccess'
  error:
    description: Error, detail in response JSON
    schema:
      $ref: '#/definitions/HttpError'
  auth_error:
    description: Authentication error, invalid or missing Token
  access_error:
    description: Forbidden access (but valid Token)

definitions:
  SensorStatus:
    type: object
    description: 'Current status of a sensor'
    properties:
      name:
        type: string
        description: |
          Name of sensor. Name can be retrieved using /config/missioninfo and/or
          through the web interface.
      activated:
        type: boolean
        description: 'Is the sensor activated on board'
      transmission:
        type: number
        description: |
          Transmission status of data from the sensor:
            0: if offline
            1: if data is send to LGCS
            2: if data is send to RPAS_DC
    example:
      sensor: AIS Receiver
      activated: true
      transmission: 2
  SensorsStatus:
    type: object
    required:
      - timestamp
```

```
      - number_of_sensors
      - sensors
    properties:
      timestamp:
        type: integer
        format: int64
        description: |
          Timestamp of sensor status (microsec since 1970-01-01, as in STANAG 4609)
        example: 1508399808000000
      number_of_sensors:
        type: number
        description: Number of sensors installed in RPAS
      sensors:
        type: array
        description: List of sensors with current status
        items:
          $ref: '#/definitions/SensorStatus'

  EPIRBMessage:
    type: object
    required:
      - reception_date
      - country
      - identifier
      - latitude
      - longitude
    properties:
      reception_date:
        type: integer
        format: int64
        description: |
          Timestamp of reception date of the message (microsec since 1970-01-01, as
in STANAG 4609)
        example: 1508399808000000
      country:
        type: string
        description: Decoded country code provided by the message
      identifier:
        type: object
        required:
          - protocol_code
          - type
          - id
        properties:
          protocol_code:
            type: integer
            description: |
              Code content in bits 37 to 39/40 of the Protocol Code (PC).
              See table A2 in Cospas-Sarsat C/S T.001 - Issue 4 documents
          type_id:
            type: string
            description: |
              Type of identifier as described in Cospas-Sarsat. Valid values are:
                - mmsi
                - callsign
                - identifier
                - national identifier
          id:
            type: string
            description: the id provided by the message
      latitude:
        type: number
        format: float
        description: |
          Latitude provided by the message (with 6 digits for precision on WGS84
          ellipsoid, as described in RFC-7946)
```

```
      longitude:
        type: number
        format: float
        description: |
          Longitude provided by the message (with 6 digits for precision on
          WGS84 ellipsoid, as described in RFC-7946)
      raw_payload:
        type: string
        description: |
          Raw binary message base64 encoded

  Gasmeasurement:
    type: object
    required:
      - precision_timestamp
      - sensor_latitude
      - sensor_longitude
      - sensor_altitude
      - CO2
      - SO2
      - NO2
      - NO
    properties:
      precision_timestamp:
        type: integer
        format: int64
        description: |
          Timestamp of the measures (microsec since 1970-01-01, as in STANAG 4609)
        example: 1508399808000000
      sensor_latitude:
        type: number
        format: float
        description: |
          Latitude of the sensor (with 6 digits for precision on WGS84
          ellipsoid, as described in RFC-7946)
        example: 38.5
      sensor_longitude:
        type: number
        format: float
        description: |
          Longitude of the sensor (with 6 digits for precision on WGS84
          ellipsoid, as described in RFC-7946)
        example: -7.5
      sensor_altitude:
        type: number
        format: float
        description: |
          Meters over ground of the sensor (with max 1 digits for precision,
          ground based on WGS84 ellipsoid)
        example: 640
      CO2:
        type: number
        description: |
          CO2 concentration in ppm; two digits behind comma
        example: 1120.21
      SO2:
        type: number
        description: |
          SO2 concentration in ppm; two digits behind comma
        example: 153.91
      NO2:
        type: number
        description: |
          NO2 concentration in ppm; two digits behind comma
        example: 253.31
      NO:
```

```
        type: number
        description: |
          NO concentration in ppm; two digits behind comma
        example: 69.41
      temperature:
        type: number
        description: |
          Atmospheric temperature in degree celcius; two digits behind comma
        example: 14.41
      pressure:
        type: integer
        description: |
          Barometric pressure in Pascals
        example:  101325
      relative_humidity:
        type: number
        description: |
          Atmospheric relative humidity in %; two digits behind comma
        example: 56.22

  ImageryMetadata:
    type: object
    required:
      - timestamp
      - frame_center_latitude
      - frame_center_longitude
    properties:
      timestamp:
        type: integer
        format: int64
        description: |
          Timestamp of when the imagery was taken (microsec since 1970-01-01, as in
STANAG 4609)
        example: 1508399808000000
      frame_center_latitude:
        type: number
        format: float
        description: |
          Latitude of the frame center of the image (with 6 digits for precision on
WGS84
          ellipsoid, as described in RFC-7946)
      frame_center_longitude:
        type: number
        format: float
        description: |
          Longitude of the frame center of the image (with 6 digits for precision
on WGS84
          ellipsoid, as described in RFC-7946)
      comment:
        type: string
        description: |
          Visual observations, or any other comment related to the image content.

  ConcentrationDesc:
    type: object
    properties:
      value:
        type: number
      accuracy:
        type: number
  ShipEmissionsReport:
    type: object
    required:
      - date_time_of_obs
      - vessel_information
      - observation_latitude
```

```
      - observation_longitude
      - position_in_seca
      - position_in_port
      - sulphur_percentage
    properties:
      date_time_of_obs:
        type: string
        format: date-time
        description: |
          Date and time of the observation/measurement. Format must follow
          RFC-3339 section 5.6 with UTC timezone (Z)
        example: |
          2014-03-14T14:15:27Z
      vessel_information:
        type: object
        description: |
          A set of info related to the vessel identification,
          e.g. IMO, MMSI, Ship name and/or Call Sign
        required:
          - imo
          - mmsi
          - ship_name
          - callsign
          - flag
        properties:
          imo:
            type: integer
            description: |
              Vessel IMO
          mmsi:
            type: integer
            description: |
              Vessel MMSI
          name:
            type: string
            description: |
              Vessel name
          callsign:
            type: string
            description: |
              Vessel callsign
          flag:
            type: string
            description: |
              Vessel country, must use the ISO-3166 Two Letter coding method
            example: PT
      observation_latitude:
        type: number
        format: float
        description: |
          Geographical latitude (with 6 digits for precision on WGS84 ellipsoid,
          as described in RFC-7946) of position of ship during the vessel
          observation
      observation_longitude:
        type: number
        format: float
        description: |
          Geographical longitude (with 6 digits for precision on WGS84 ellipsoid,
          as described in RFC-7946) of position of ship during the vessel
          observation
      position_in_seca:
        type: boolean
        description: |
          Set to true if vessel position is in SECA
      position_in_port:
        type: boolean
```

```
        description: |
          Set to true if vessel position is in port area
      number_of_measurements:
        type: integer
        description: |
          Number of measurements taken during one vessel observation
        example: 3
      sulphur_percentage:
        type: number
        format: float
        description: |
          Derived Sulphur percentage in the fuel; Percentage, three digits
          behind comma
        example: 0.120
      sulphur_measurement_uncertainty:
        type: number
        format: float
        description: |
          Sulphur measurement uncertainty expressed as a sulphur percentage
          value +/- at 1 x RSD; Percentage, three digits behind comma
        example: 0.200
      nox_emission:
        type: number
        format: float
        description: |
          Derived NOx emission value; g/kWh, three digits behind comma
        example: 2.340
      nox_measurement_uncertainty:
        type: number
        format: float
        description: |
          The NOx measurement uncertainty expressed as g/kWh +/- at 1 x RSD;
          value, three digits behind comma.
        example: 0.200
      quality_score:
        type: number
        format: float
        description: |
          Quality score of the measurement ranging from 0-10 where 10 expresses
          a measurement that satisfies all sensor quality criteria in terms of
          gas concentration levels and sensor exposure time in plume; Value, two
          digits behind comma
        example: 9.36
      gas_measurements:
        type: object
        description: |
          A set of gas measurements linked with Emissions Report calculations:
          SO2, CO2, NO and NO2.
        properties:
          SO2:
            description: 'SO2 value unit is ppm'
            $ref: '#/definitions/ConcentrationDesc'
          CO2:
            description: 'CO2 value unit is ppm'
            $ref: '#/definitions/ConcentrationDesc'
          NO:
            description: 'NO value unit is ppm'
            $ref: '#/definitions/ConcentrationDesc'
          NO2:
            description: 'NO2 value unit is ppm'
            $ref: '#/definitions/ConcentrationDesc'
        example:
          SO2:
            value: 123.9
            accuracy: 0.12
          CO2:
```

```
            value: 1234
            accuracy:  0.32
          NO2:
            value: 226.9
            accuracy: 0.23
          NO:
            value: 43.4
            accuracy: 0.52
      comment:
        type: string
        description: |
          Visual observations, or any other comment related to the emissions
report.

  AnnotationReport:
    type: object
    required:
    - geometry
    - properties
    properties:
      type:
        enum:
        - Feature
      geometry:
        $ref: '#/definitions/Geometry'
      properties:
        type: object
        required:
          - id
          - timestamp
          - type
          - comment
        properties:
          id:
            type: string
            description: |
              Unique identifier of the pollution, must be reused in case of
              report update.
          timestamp:
            type: integer
            format: int64
            description: |
              Reference timestamp of the annotation (microsec since 1970-01-01, as
in STANAG 4609)
            example: 1508399808000000
          label:
            type: string
            description: |
              Short description of the annotation
          extra_data:
            type: object
            description: |
              Set of extra data about annotation (flat dictionary, string,
              integer, number and boolean values only)
            additionalProperties:
              type: ['string', 'integer', 'number', 'boolean']
          comment:
            type: string
    example:
      geometry:
        type: Polygon
        coordinates:
        - [[-7.16, 36.96],
           [-7.16, 37.03],
           [-6.93, 37.03],
           [-6.93, 36.96]]
```

```
      properties:
        id: ELOpolygon
        timestamp: 1508399808000000
        label: OilSpill
        extra_data:
          estimate_lenght: 12345
          color: 'black'
        comment: oil spill detected by Elodie

  AircraftPosition:
    type: object
    required:
      - timestamp
      - latitude
      - longitude
      - altitude
      - ground_speed
      - heading
    properties:
      timestamp:
        type: integer
        format: int64
        description: |
          Timestamp of the localisation (microsec since 1970-01-01, as in STANAG
4609)
        example: 1508399808000000
      latitude:
        type: number
        format: float
        description: |
          Latitude of the platform (with 6 digits for precision on WGS84
          ellipsoid, as described in RFC-7946)
      longitude:
        type: number
        format: float
        description: |
          Longitude of the platform (with 6 digits for precision on WGS84
          ellipsoid, as described in RFC-7946)
      altitude:
        type: number
        format: float
        description: |
          Meters over ground of the platform (with max 1 digits for precision,
          ground based on WGS84 ellipsoid)
      ground_speed:
        type: number
        format: float
        description: |
          Ground speed of the aircraft in m/s (with 1 or 2 digits for precision)
      heading:
        type: number
        format: float
        description: |
          Aircraft heading angle in degree (with 0 or 1 digits for precision)
    example:
      timestamp: 1508399808000000
      latitude: 37
      longitude: -6.1
      altitude: 650
      ground_speed: 6
      heading: 20

  WayPoint:
    type: object
    required:
      - id
```

```
      - description
      - latitude
      - longitude
      - altitude
    properties:
      id:
        type: integer
        minimum: 0
        description: |
          Id of the WayPoint
        example: 1
      description:
        type: string
        description: |
          Description of the waypoint, example: Loiter, Return home, target XXX
        example: Loiter
      latitude:
        type: number
        format: float
        description: |
          Latitude of the waypoint (with 6 digits for precision on WGS84
          ellipsoid, as described in RFC-7946)
      longitude:
        type: number
        format: float
        description: |
          Longitude of the waypoint (with 6 digits for precision on WGS84
          ellipsoid, as described in RFC-7946)
      altitude:
        type: number
        format: float
        description: |
          Meters over ground of the waypoint (with max 1 digits for precision,
          ground based on WGS84 ellipsoid)
    example:
      id: 1
      description: Loiter
      latitude: 37
      longitude: -6.1
      altitude: 650

  AircraftPlan:
    type: object
    description: |
      A collection of waypoints (lon/lat/alt) with ID and description which are
      ordered sequentially.

      An activated property with the following possible values:
        1. an ID of one of waypoints part of the flight plan
        2. a discrete navigation point (lon/lat/alt) in a list
        3. null if no waypoint is active (hold flight or manual)
    required:
      - timestamp
      - waypoints
      - activated
    properties:
      timestamp:
        type: integer
        format: int64
        description: |
          Timestamp of the definition of the new aircraft plan (microsec since
1970-01-01, as in STANAG 4609)
        example: 1508399808000000
      waypoints:
        type: array
        description: Ordered list of waypoints
```

```
        items:
          $ref: '#/definitions/WayPoint'
      activated:
        type: any
        description: |
           A activated property with the following possible values:
             1. an ID of one of waypoints part of the flight plan
             2. an discrete navigation point (lon/lat/alt) in a list
             3. null if no waypoint is active (hold flight or manual)
        example: [ -7.5, 35, 650]

  AircraftComm:
    type: object
    description: The aircraft communication status
    required:
      - timestamp
      - comm_type
    properties:
      timestamp:
        type: integer
        format: int64
        description: |
          Timestamp of communication status (microsec since 1970-01-01, as in
STANAG 4609)
        example: 1508399808000000
      comm_type:
        type: string
        enum:
          - RADIO
          - SATCOM
      comm_quality:
        type: object
        description: |
          Parameters that are usually taken as indicators to quantify the
          quality of a radio link
        properties:
          c_n0:
            type: number
            description: |
              Signal strength - carrier-to-noise-density ratio (C/N0) is the
              ratio of the carrier C to the noise power density N0, expressed in
              dB-Hz
          BER:
            type: number
            description: |
              BER
          PRR:
            type: number
            description: |
              Packet Reception Rate (PRR), number of packets successfully
              received, taking into account also the network capacity
          latency:
            type: number
            description: |
              Latency in ms

  AircraftAutonomy:
    type: object
    required:
      - timestamp
      - estimate_distance_autonomy
      - estimate_time_autonomy
    properties:
      timestamp:
        type: integer
        format: int64
```

```
        description: |
          Timestamp of autonomy computation (microsec since 1970-01-01, as in
STANAG 4609)
        example: 1508399808000000
      estimate_distance_autonomy:
        type: integer
        description: distance in kilometers
      estimate_time_autonomy:
        type: integer
        description: duration in minutes
    example:
      timestamp: 1508399808000000
      estimate_distance_autonomy: 314
      estimate_time_autonomy: 142

  DCStreamStatus:
    type: object
    required:
      - timestamp
      - sensors
    properties:
      timestamp:
        type: integer
        format: int64
        description: |
          Timestamp of stream status information (microsec since 1970-01-01, as in
STANAG 4609)
      sensors:
        type: array
        minItems: 1
        items:
          type: object
          required:
            - name
            - receive_data
          properties:
            name:
              type: string
              description: |
                Sensor name as referenced in RPAS-DC database
            receive_data:
              type: boolean
              description: |
                True if data is received for that sensor.
            reception_bandwidth:
              type: number
              description: |
                Reception bandwidth of stream on the DataCenter side (in bits
                per seconds). Only provided for stream sensor (EO, IR, etc.).
    example:
      timestamp: 1508399808000000
      sensors:
        - name: EO
          receive_data: true
          reception_bandwidth: 4987472
        - name: SAR
          receive_data: false
          reception_bandwidth: 9790048

  MissionInfo:
    type: object
    properties:
      name:
        type: string
        description: Name of the mission
      abstract:
```

```
        type: string
        description: Description of the mission (purpose, etc.)
      emsa_id:
        type: string
      comment:
        type: string
        description: Comment about the mission
      status:
        type: string
        description: Current status of the mission
        enum:
          - Planned
          - Ongoing
          - Canceled
          - Aborted
          - Finished
          - Archived
      start_datetime:
        type: string
        format: date-time
        description: Start date of the mission
      end_datetime:
        type: string
        format: date-time
        description: Planned end date of the mission
      rpas:
        type: object
        properties:
          name:
            type: string
            description: RPAS name defined by the Service Provider
          callsign:
            type: string
            description: RPAS official CALLSIGN
          emsa_code:
            type: string
            description: RPAS code defined by EMSA
          sensors:
            type: array
            minItems: 1
            items:
              type: object
              properties:
                name:
                  type: string
                  description: Sensor name as referenced in RPAS-DC database
                streaming_address:
                  type: string
                  description: |
                    URL to be used to send sensor stream with MPEG2TS protocol
                    (STANAG 4609 interface) over UDP.

                    *streaming_address* is provided only for sensors that
                    produce video stream (EO, IR, LIR, SAR, maritime radar, ...)
      chat_room_name:
        type: string
      ftp_url:
        type: string
    example:
      name: Beta
      abstract: First Brest Mission
      emsa_id: mission_brest1
      comment: Test
      status: Planned
      start_datetime: '2016-10-01T13:00:00Z'
      end_datetime: '2017-10-01T13:00:00Z'
```

```
      rpas:
        name: AR5
        callsign: '7056'
        emsa_code: AR5_324487
        sensors:
          - name: IR
            streaming_address: "tcp://127.0.0.1:5000"
          - name: Optical
            streaming_address: "tcp://127.0.0.1:5002"
      chat_room_name: 1_beta
      ftp_url: ftp://127.0.01/Deployment_brest/Mission_brest1
  HttpSuccessAIS:
    type: object
    properties:
      success:
        type: boolean
        description: Always at true for success response
      nb_receive_data:
        type: integer
        description: Number of received messages in the request
    example:
      success: true
      nb_receive_data: 5
  HttpSuccess:
    type: object
    properties:
      success:
        type: boolean
        description: Always at true for success response
    example:
      success: true
  TokenSuccess:
    type: object
    properties:
      success:
        type: boolean
        description: Always at true for success response
      jwt:
        type: string
        description: |
          Json Web Token to be provided for secure request (in the Authorization
          header)
    example:
      success: true
      jwt:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOjEyLCJwb3J0Ijo1MDAwLCJtaWQiOjIwfQ.Yql
27-OAbNJJLfMaxZarfDu2kH9inUFXwBfdDKuFbAQ
  HttpError:
    type: object
    properties:
      success:
        type: boolean
        description: Always at false for error response
      message:
        type: string
        description: Message with error details
    example:
      success: false
      message: Invalid token to do this request

################################################
# Definitions needed to define specific geojson #
################################################
  geometryCollection:
    title: GeometryCollection
    description: A collection of geometry objects
```

```
    required:
    - geometries
    properties:
      type:
        type: string
        enum:
          - GeometryCollection
      geometries:
        type: array
        items:
          $ref: '#/definitions/Geometry'
  feature:
    title: Feature
    description: A Geo JSON feature object
    required:
    - geometry
    - properties
    properties:
      type:
        type: string
        enum:
          - Feature
      geometry:
        oneOf:
        - type: 'null'
        - $ref: '#/definitions/Geometry'
      properties:
        type:
          - object
          - 'null'
  featureCollection:
    title: FeatureCollection
    description: A Geo JSON feature collection
    required:
    - features
    properties:
      type:
        type: string
        enum:
        - FeatureCollection
      features:
        type: array
        items:
          "$ref": "#/definitions/feature"
  Geometry:
    description: One geometry as defined by GeoJSON
    type: object
    required:
      - type
      - coordinates
    oneOf:
      - title: Point
        properties:
          type:
            type: string
            enum:
              - Point
          coordinates:
            "$ref": "#/definitions/position"
      - title: MultiPoint
        properties:
          type:
            type: string
            enum:
              - MultiPoint
          coordinates:
```

```
            "$ref": "#/definitions/positionArray"
      - title: LineString
        properties:
          type:
            type: string
            enum:
              - LineString
          coordinates:
            "$ref": "#/definitions/lineString"
      - title: MultiLineString
        properties:
          type:
            type: string
            enum:
              - MultiLineString
          coordinates:
            type: array
            items:
              "$ref": "#/definitions/lineString"
  Point:
    description: A Point as defined by GeoJSON
    type: object
    required:
      - type
      - coordinates
    properties:
      type:
        type: string
        enum:
          - Point
      coordinates:
        "$ref": "#/definitions/position"
  Polygon:
    description: A Polygon as defined by GeoJSON
    type: object
    required:
      - type
      - coordinates
    properties:
      type:
        type: string
        enum:
          - Polygon
      coordinates:
        $ref: "#/definitions/polygon"
  MultiPolygon:
    description: A MultiPolygon as defined by GeoJSON
    type: object
    required:
      - type
      - coordinates
    properties:
      type:
        type: string
        enum:
          - MultiPolygon
      coordinates:
        type: array
        items:
          $ref: "#/definitions/polygon"
  position:
    description: |
      A single position (lon/lat with 6 digits for precision on WGS84 ellipsoid,
      as described in RFC-7946)
    type: array
    minItems: 2
```

```
    items:
      type: number
    additionalItems: false
  positionArray:
    description: |
      An array of positions (lon/lat with 6 digits for precision on WGS84
      ellipsoid, as described in RFC-7946)
    type: array
    items:
      $ref: "#/definitions/position"
  lineString:
    description: An array of two or more positions
    allOf:
      - $ref: "#/definitions/positionArray"
    minItems: 2
  linearRing:
    description: An array of four positions where the first equals the last
    allOf:
      - $ref: "#/definitions/positionArray"
    minItems: 4
  polygon:
    description: An array of linear rings
    type: array
    items:
      $ref: "#/definitions/linearRing"
```