Casia MQTT Communications Interface Control Document

Contents

1. Description

Casia relays data via multiple interfaces (such as Serial and Ethernet) and over different protocols (such as MAVLink and MQTT) in a configurable manner. This document covers the MQTT communications implementation within Casia's FlightCore (FC) software.

The implementation and design of the MQTT protocol described herein was designed for use with Casia G devices which will be connected to the user via the internet over the Casia Network or optionally can be configured to operate on a local network without internet.

Currently the interface is output only and does not support bi-directional communication for command of the Casia device.

2. Document History

Rev.	Date	Changes	Authors	FlightCore Version
1.0	2022/06/17	Document creation.	Andrea G, Brian M, James H	2.4.1
1.1	2022/12/14	Corrections to message details, topics, config.	Brian M	3.1.0
1.3	2023/03/01	Included rough message sizes. Updates to message details that correspond to the latest release.	Brian M	3.2.0
1.4	2023/05/17	Updates to message details that correspond to the new release.	Sam W	3.3.0
1.5	2024/03/05	Corrections to message details.	Sam W	3.3.0
1.6	2025/05/12	Updates to message details that correspond to the new release, including details about night mode. Reformatted document with uAvionix branding.	Raquel D, James H	4.0.0

3. Casia Settings

The following list contains all the available configuration parameters, descriptions, types, and values that are available for the communications module of FlightCore. Modification of these settings can be done via the FlightDeck web interface.

This is a subset of the total FlightCore parameters list as described in the Casia Userguide and other documentation.

Configuration Parameter	Description	Type	Options
comms__enabled	Enables or disables output from the communications module.	bool	-
comms__protocol	Selects the interface protocol to be used.	string	mqtt, mqtts
comms_features_hz	Sets the frequency of the features message in Hz. 0 will send features only once at boot.	float	-
comms__heartbeat_hz	Sets the frequency of the heartbeat message in Hz.	float	-

`comms__status_hz`	Sets the frequency of the status in Hz. 0 will send status only once at boot.	`float`	-
`comms_telemetry_max_hz`	Sets the maximum frequency of the telemetry message in Hz. If the input telemetry to Casia is sent at a rate less than configured here, that will be the rate used. 0 will send telemetry only once at boot.	`float`	-
`comms_server_address`	The IPv4 address or URL of the MQTT broker.	`string`	-
`comms_server_port`	The port number to connect to on the MQTT broker.	`int`	-
`comms_username`	The username to use for authenticating with the broker. Can be left empty if not used.	`string`	-
`comms_password`	The password to use for authenticating with the broker. Can be left empty if not used.	`string`	-
`comms_validate_certificate`	Whether to validate the server certificate from centrally signed authorities or not.	`bool`	-
`comms_certificate _authentication_enabled`	Whether or not to authenticate to the broker using SSL/TLS certificates.	`bool`	-
`comms_client_certiciate`	Client SSL/TLS certiciate ot use to authenticate to the broker.	`string`	-
`comms_client_key`	Client private key for SSL/TLS communication.	`string`	-
`comms_client_key_encrypted`	Whether or not client private key for SSL/TLS communication is encrypted.	`bool`	-
`comms_client_key_password`	Password to decrypt client private key for SSL/TLS communication if encrypted.	`string`	-

4. Messaging

4.1 Revision History

The following table lists the versions of the MQTT message protocol further detailed in this document. This version number is sent within the header message from the device which indicates to the receiver the expected contents of the message.

The version number is specified as major dot minor (e.g. one dot zero is major version one, minor version zero). When a change to the protocol is made that will break past integrations the major version will be incremented, when additions or small changes are made the minor version is incremented.

Version	Changes	FlightCore Version
1.0	Initial version of the MQTT message protocol.	2.4.1
1.1	Modifications made to support Night Mode.	3.5.0

4.2 Header

Each message sent from the communications module has a header as described here. This header gives some basic and useful identification, version, and timing information to the receiver.

Field	Description	Type	Version
`timestamp`	The time when the message was sent in milliseconds since the Unix epoch.	`int`	v1.0
`serial_number`	The serial number of the device that sent the message.	`string`	v1.0
`type`	The message type (see below).	`string`	v1.0

`payload_major_version`	Major version number of the message. If this number changes, it indicates a breaking change, where previous ground stations may not work properly with this message version.	`int`	v1.0
`payload_minor_version`	Minor version number of the message. This indicates the contents of the message have changed, but the change itself is not a breaking change, meaning previous versions of ground stations should still be compatible.	`int`	v1.0
`payload`	The contents of the message. This changes for each type of message sent according to the sections below.	`JSON object`	v1.0

4.3 Types

Different message types are defined to communicate specific information from the Casia device to the user. These types define the payload of the message sent with the header. The following message types are defined.

Name	Description	Rate	Format	Message Size
`heartbeat`	A periodic notification to the receiver that the device is still online.	1s interval	`JSON object`	~0.1 kB
`status`	The status of the device, storage, connection, and other parameters.	60s interval	`JSON object`	~0.5 kB
`features`	The current features of the device. Software version, Camera features, ADS-B, etc.	60s interval	`JSON object`	~1-2 kB
`telemetry`	The current location and other telemetry details of the device. The actual rate of the message depends on Casia telemetry inputs.	5s interval	`JSON object`	~0.5 kB
`intruder`	Details on a singular detected intruder.	Asynchronous	`JSON object`	min ~1 kB max ~40 kB ~1 kB for ADS-B ~20 kB for Vision

4.3.1 Heartbeat

The heartbeat message does not contain a payload, it is a header-only message.

4.3.2 Status

The status message contains all the status information of the device. Typically, it is data that changes more frequently than the features message, so it can be required at a higher frequency.

Field	Description	Type	Version
`is_online`	If casia is online, LWT will set this to false.	`bool`	1.0

detector_mode	Specifies day or night mode for Casia detector.	enum	1.1
timestamp_boot	Timestamp when the device booted in ms since the Unix epoch.	intv	1.0

4.3.2.1 Detector Mode Enum

Value	String Representation	Description
0	disabled	The detector is disabled.
1	day	The detector is in day detection mode.
2	night	The detector is in night detection mode.

4.3.3 Features

The features message contains the current features of the device. Usually it is configured at a lower rate than the status message, since it does not contain data that changes frequently.

The features message contains the current features of the device. Usually it is configured at a lower rate than the status message, since it does not contain data that changes frequently.

Name	Description	Type	Version
camera_count	Number of configured cameras.	int	1.0
cameras	Details on each camera. See Cameras section below.	JSON list	1.0
adsb_count	Number of configured ADS-B receivers.	int	1.0

4.3.3.1 Cameras Object

Cameras is a JSON list that contains the details for each detected camera.

```
[ camera_0, camera_1, etc... ]
```

Each camera is a JSON object containing the following fields:

Name	Description	Type	Version
index	Camera index, starting from 0	int	1.0
serial_number	The Serial Number of the camera	string	1.0
occlusion_mask	A list of Occlusion regions. Pixels covered by occlusion regions are completely omitted by the image processing. The most common use case is to block out a propeller from the image. See Masks section below.	JSON list	1.0
rejection_mask	A list of Rejection regions. Pixels covered by rejection regions are not omitted by the image processing, but	JSON list	1.0

	cannot generate detections. The most common use case is to block out false positive coming from the ground (e.g. moving cars). See Masks section below.		
`extrinsic_matrix`	A 3x4 matrix in row-major order. It encodes the position and orientation of the camera relative to the vehicle. Pitch, Roll, Yaw, X, Y and Z can be obtained from here.	`JSON list`	1.0
`fov_horizontal`	Horizontal field of view, in degrees	`float`	1.0
`fov_vertical`	Vertical field of view, in degrees	`float`	1.0
`image_width`	Width of the image, in pixels	`int`	1.0
`image_height`	Height of the image, in pixels	`int`	1.0
`resolution`	Resolution of the camera, in pixels	`int`	1.0

4.3.3.2 Masks Object

Each mask is a list of regions. A region is a polygon represented by an ordered list of vertexes. A polygon must be represented by at least 3 vertices; thus, the minimum size of the region is 6. An example polygon is shown below.

```
{x0, y0, x1, y1, x2, y2, ...}
```

4.3.4 Telemetry

The telemetry message describes location and other telemetry details of the device.

Name	Description	Type	Version
`timestamp`	Current timestamp as reported by Casia's clock, in milliseconds since the Unix epoch.	`int`	1.0
`timestamp_gps`	Current timestamp as reported by the GPS device, in milliseconds since the Unix epoch.	`int`	1.0
`latitude`	Position latitude in decimal degrees.	`float`	1.0
`longitude`	Position longitude in decimal degrees	`float`	1.0
`altitude`	Altitude in meters in AMSL reference frame.	`float`	1.0
`heading`	System heading in radians from true north.	`float`	1.0
`velocity_x`	System velocity in the X axis.	`float`	1.0
`velocity_y`	System velocity in the Y axis.	`float`	1.0

velocity_z	System velocity in the Z axis.	float	1.0
roll	System angle in radians.	float	1.0
yaw	System yaw angle in radians.	float	1.0

4.3.5 Intruder

The intruder message can represent a Vision intruder (either day or night), an ADS-B intruder, or a correlated intruder where ADS-B and Vision information for the same aircraft are combined.

Valid intruder fields can differ between Day Vision, Night Vision, ADS-B, and Day or Night Correlated intruder types. Please note that fields may be present and include a value even when they are invalid for that intruder type. The following table reports which fields should be used and which should be discarded for each intruder type.

Field	Description	Type	Intruder Type	Version
sensor_timestamp	The time reported by the sensor of when the intruder was detected (e.g. for a camera, this is the timestamp of the frame capture).	int	All	1.0
sensor_type	Sensor type that reported the detection. See Sensor Type.	enum	All	1.0
id	Unique ID for the intruder. *Note: Intruder IDs are unique only for a single device, they are not globally unique to all Casia systems.*	int	All	1.0
icao_address	Intruder's ICAO address from the ADS-B message.	int	ADS-B, Correlated	1.0
callsign	Intruder callsign as reported by ADS-B or the string "VISION" if a visual detection.	string	All	1.0
type	Vision system intruder type classification as integer. See type_name field for a string representation.	enum	Vision, Correlated	1.0
type_name	Vision system intruder type classification as string.	string	Vision, Correlated	1.0
adsb_emitter_type	Intruder type classification following ADSB standards. See Emitter Type in MAVLink documentation.	int	Day Vision, ADS-B, Correlated	1.0
adsb_emitter_type_name	ADS-B emitter type classification as string.	string	Day Vision, ADS-B, Correlated	1.0
sensor_id	ID of the sensor that reported the detection.	int	All	1.0
frame_id	ID of the frame that generated the detection.	int	All Vision	1.0
latitude	Latitude in decimal degrees.	float	Day Vision, ADS-B, Correlated	1.0
longitude	Longitude in decimal degrees.	float	Day Vision, ADS-B, Correlated	1.0

altitude	Altitude in meters above the altitude reference. See altitude_type to determine the altitude reference.	float	Day Vision, ADS-B, Correlated	1.0
altitude_type	Altitude reference. See Altitude Type in MAVLink documentation.	enum	All	1.0
heading	Heading in radians from true north.	float	Day Vision, ADS-B, Correlated	1.0
velocity_vertical	Vertical velocity in meters/second.	float	Day Vision, ADS-B, Correlated	1.0
velocity_horizontal	Horizontal velocity in meters/second	float	Day Vision, ADS-B, Correlated	1.0
squawk	Aircraft squawk codes. *Note: 1200 is the placeholder for vision detections.*	float	All	1.0
position_x	X position in meters, relative to the device.	float	Day Vision	1.0
position_y	Y position in meters, relative to the device.	float	Day Vision	1.0
position_z	Z position in meters, relative to the device.	float	Day Vision	1.0
velocity_x	X velocity in meters per second, coordinates relative to the device.	float	Day Vision	1.0
velocity_y	Y velocity in meters per second, coordinates relative to the device.	float	Day Vision	1.0
velocity_z	Z velocity in meters per second, coordinates relative to the device.	float	Day Vision	1.0
azimuth	The horizontal angle to the intruder relative to the device, in radians.	float	All Vision	1.0
elevation	The vertical angle to the intruder relative to the device, in radians.	float	All Vision	1.0
range	Slant range to the intruder, in meters.	float	Day Vision	1.0
azimuth_rate	Rate of change of azimuth angle to the intruder, in radians per second.	float	All Vision	1.0
elevation_rate	Rate of change of elevation angle to the intruder, in radians per second.	float	All Vision	1.0
range_rate	Rate of change of the slant range to the intruder, in meters per second.	float	Day Vision	1.0
dynamic_object_confidence	0 to 100 confidence value that the object is a moving object.	int	Day Vision	1.0
classifier_confidence	0 to 100 confidence value that the object's class type is correct.	int	Day Vision	1.0
detector_confidence	0 to 100 confidence value that the object is an aircraft, if detected by the deep learning detection system.	int	Day Vision	1.0
correlation_type	Correlation type for the detected intruder.	enum	All Vision	1.0

`correlated_ids`	Id values of previous intruders that have been correlated with this one.	`JSON List`	`All Vision`	1.0
`images`	Image of the detected intruder, if available.	`JSON List`	`All Vision`	1.0

4.3.5.1 Sensor Type Enum

Value	String Representation	Description
0	Vision	Object was detected visually.
1	ADS-B Direct	Object was detected via ADS-B, directly connected to Casia.
2	ADS-B Indirect	Object was detected via ADS-B, indirectly connected via the autopilot.

4.3.5.2 Intruder Type Enum

Value	String Representation	Description
0	Unknown	Object is of an unknown type.
1	Small Plane	Object is classified as a small plane.
2	Helicopter	Object is classified as a helicopter.
3	Multirotor	Object is classified as a multirotor drone.
4	Bird	Object is classified as a bird.

4.3.5.3 Altitude Type Enum

Value	String Representation	Description
0	Pressure Altitude	Altitude is reported from a barometer.
1	Geometric Altitude	Altitude is reported from a GNSS device.

4.3.5.4 Correlation Type Enum

Value	String Representation	Description
0	Uncorrelated	The detection is not correlated to any other detections.
1	Vision Correlated	Two vision detections have been correlated and the data has been fused. This can occur between overlapping cameras.
2	Vision and ADS-B Correlated	A vision detection has been correlated with an ADS-B detection and the data has been fused.

4.3.5.5 Images List & Image Object

The images list contains two image objects, the “bounding_box” and “patch”.

The bounding box image is tightly cropped from the full camera frame around the detected object, it contains no padding, and the dimensions of the image vary.

The patch image is cropped from the full camera frame centered on the detected object and has a fixed dimension of 128px square. This is the image we recommend displaying in a user interface.

Below is the structure of the image object, note that "metadata" and "data" are top level keys, with width, height, x, y, and format being nested keys within the metadata object.

Name		Description	Type
`metadata`	`width`	Width of the image in pixels.	`int`
	`height`	Height of the image in pixels.	`int`
	`x`	X position of the top left of the image within the larger camera frame.	`int`
	`y`	Y position of the top left of the image within the larger camera frame.	`int`
	`format`	String representation of the image format (e.g. ".png")	`string`
`data`		Binary image data encoded in base64.	`string`

4.4 Topics

MQTT messages are sent on topics which allow organization and segregation of the send data. Within the Casia Network a hierarchical topic and message structure is used to separate each message from the other and to separate messages from each device. The following is the topic structure template and examples.

```
device/<serial number>/<message type>

device/GACM-0100-000004/heartbeat

device/GACM-0100-000004/status

device/GACM-0100-000004/intruder

etc...
```

To subscribe to all messages from a device, the # symbol can be used as a wildcard as follows.

```
device/GACM-0100-000004/#
```

Note: Subscribing to all messages from all devices is not allowed.

5. Protocols

5.1 MQTT

The MQTT protocol is designed for IoT applications for sensors on remote networks and is therefore suitable for use cases such as the Casia G where devices are deployed in distance locations across disparate networks but need to communicate to a common client/user.

The protocol follows the publisher/subscriber pattern, where each Casia device publishes data on specific topics, and the client can decide to subscribe to one or multiple topics, coming from one or multiple Casia devices.

5.1.1 Quality of Service, Retention, and Will

The MQTT protocol defines several special functions which Casia utilizes which we will discuss here, this includes quality of service (QoS), message retention, and the last will and testament.

Quality of Service is used by Casia to guarantee the delivery the messages between different sessions, in case of client disconnection. Quality of Service is available in three levels listed below. The higher the level of QoS the more bandwidth is needed to transmit the message and the more overhead there is within the protocol.

- Delivery at most once (QOS 0)
- Delivery at least once (QOS 1)
- Delivery exactly once (QOS 2)

Message Retention allows the broker to store the last message and the corresponding QoS for a given topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic. Casia uses this mechanism to retain information relevant to the display of the system in a user interface, allowing the interface to immediately populate with information before Casia sends refreshed data.

Last Will and Testament (LWT) is a standard MQTT message provided by the client device that the broker stores until it detects that the client has disconnected ungracefully. In response to the ungraceful disconnect, the broker sends the last-will message to all subscribed clients of the last-will message topic. Casia uses a last will and testament with the status message to notify clients that it has gone offline by setting the `is_online` field to false.

Message Type	QoS Level	Retained	Will
heartbeat	0	No	No
status	0	Yes	Yes
features	0	Yes	No
telemetry	0	Yes	No
intruder	1	No	No

5.2 MQTTS

Everything that applies to MQTT also applies to MQTTS. The only difference is that MQTTS uses SSL encryption over TLS providing in addition:

- Data encryption between the Casia device and the MQTT broker.
- Server authentication if server certificate validation is enabled and the server has a valid certificate.

6 References & Resources

We recommend researching the MQTT protocol with the following resources.

HiveMQ - MQTT Essentials