EXPLORATORY RESEARCH

UTM Platform Architecture including ICD and Integration Test Report

DeliverableID	D5.1
ProjectAcronym	ICARUS
Grant:	894593
Call:	H2020-SESAR-2019-2
Topic:	Topic 31
Consortium coordinator:	E-GEOS SPA
Edition date:	2 November 2021
Edition:	00.01.04
Template Edition:	02.00.00

Authoring & Approval

Authors of the document

Name/Beneficiary	Position/Title	Date
Piotr Dybiec/DroneRadar	Project Manager	1.10.2021
Paweł Korzec/DroneRadar	CEO	29.10.2021
Łukasz Górny-Zając/DroneRadar	Chief Architect	29.10.2021
Adam Staszak/DroneRadar	Senior Developer	29.10.2021
Jan Mróz-Jaworski/DroneRadar	System Architect	29.10.2021
Sławomir Kubiak/DroneRadar	Senior Analyst	29.10.2021

Reviewers internal to the project

Name/Beneficiary	Position/Title	Date
Corrado Orsini/Telespazio	Technical Coordinator	2.11.2021
Peter Hullah/Eurocontrol	Project Team Member	2.11.2021

Approved for submission to the SJU By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date
Cristina Terpessi	Project Coordinator	3.11.2021
Corrado Orsini/Telespazio	Technical Coordinator	3.11.2021
Alberto Mennella/TopView	Technical Coordinator	3.11.2021
Manuel Onate	Communication and Dissemination Manager	5.11.2021
Peter Hullah/Eurocontrol	Project Team Member	5.11.2021

Rejected By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date

Document History

Edition	Date	Status	Author	Justification
00.00.01	30.09.2021	Draft	Piotr Dybiec	Initial ToC, content proposal and document structure

00.00.02	22.10.2021	Draft	Piotr Dybiec, Adam Staszak	Description of DRONE RADAR USSP system architecture
00.00.03	29.10.2021	Draft	Piotr Dybiec, Adam Staszak, Paweł Korzec, Jan Mróz-Jaworski, Łukasz Górny Zając	Description of GISP, WSP, test cases
00.00.04	02.11.2021	Draft		Final version for internal approval
00.01.04	02.11.2021			Final Version

Copyright Statement

ICARUS

INTEGRATED **C**OMMON **A**LTITUDE **R**EFERENCE SYSTEM FOR **U**-**S**PACE

This document is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No. 894593 of the European Union's Horizon 2020 research and innovation programme.

Abstract

This document describes the functional architectures of the USSP (Droneradar), its integration interfaces with the ICARUS CARS system and specify all the interfaces regarding the Weather Service Provider (WSP) and GI Service Provider (GISP) platforms.

Moreover, the document describes the data flows regarding the telemetry service.

Finally, it contains the tests to be performed to test and verify the communication between the platforms mentioned and the ICARUS CARS system.

Table of Contents

List of Figures

1 Introduction

1.1 Purpose of the Document

This document describes how the USSP (Droneradar USSP), GISP and WSP platforms are – as external systems - integrated with the ICARUS CARS. It describes the functionalities/processes of the platforms mentioned, which require the use of the ICARUS CARS system. Platforms in scope are shown and marked blue in the Figure 1. Also, the communication interfaces used to exchange data between these systems are shown and described in this document.

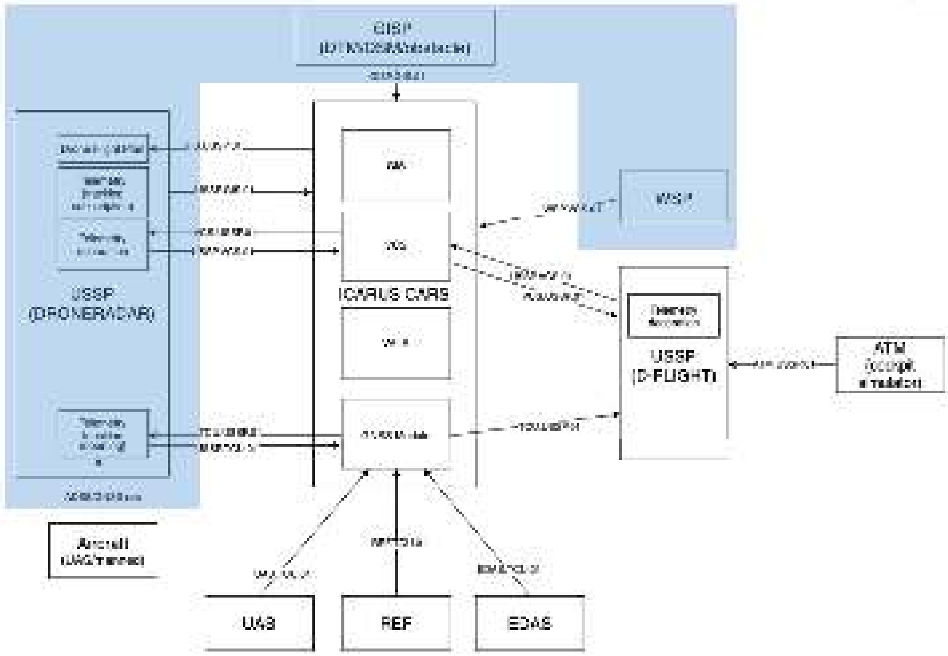

Figure 1 Integration diagram of Droneradar USSP, WSP and GISP with ICARUS CARS

1.2 Acronyms

Acronym	Meaning
API	Application Programming Interface
AGL	Above Ground Level

ASL (proposition)	Above Surface Level
ARAIM	Advanced RAIM
ATC	Air Traffic Control
ATM	Air Traffic Management
ATZ	Aerodrome Traffic Zone
BKG	Bundesamt für Kartographie und Geodäsie
BNC	BKG NTRIP Client
BVLOS	Beyond Visual Line of Sight
CARS	Common Altitude Reference System
CIS	Common Information Service
CORBA	Common Object Request Broker Architecture
CTR	Control zone
DAA	Detect And Avoid
DEM	Digital Elevation Model
DSM	Digital Surface Model
DTM	Digital Terrain Model
DOP	Dilution Of Precision
DSM	Digital Surface Model
DTM	Digital Terrain Model
EASA	European Union Aviation Safety Agency
EDAS	EGNOS Data Access Service
EGNOS	European Geostationary Navigation Overlay Service
EGNSS	European Global Navigation Satellite System
FLTA	Forward Looking Terrain Avoidance
GA	General Aviation
GAMZ	Geometric Altitude Mandatory Zone

GI	Geo Information
GNSS	Global Navigation Satellite System
GO	Ground Obstacle
GPS	Global Positioning System
HALB	Horizontal Alert Buffer
HPL	Horizontal Protection Level
ICAO	International Civil Aviation Organisation
ISA	International Standard Atmosphere
ISM	Integrity Support Message
ISO	International Organisation for Standardisation
MCMF	Multi-Constellation Multi-Frequency
NTRIP	Networked Transport of RTCM via Internet Protocol
PL	Protection Level
QFE	Query Field Elevation
QNH	Query: Nautical Height
RAIM	Receiver Autonomous Integrity Monitoring
RIMS	Ranging Integrity Monitoring Stations
RGIS	Real Time GIS
RNP	Required Navigation Performance
RTCM	Radio Technical Commission for Maritime Services
SBAS	Satellite-Based Augmentation System
SORA	Specific Operations Risk Assessment
TCU	Telespazio's Computing Unit
TSE	Total System Error
UA	Unmanned Aircraft
UAS	Unmanned Aircraft System

USSP	U-Space Service Providers (alias UTM service provider)
UTM	Unmanned aircraft system Traffic Management (alias U-space)
VALB	Vertical Alert Buffer
VCS	Vertical Conversion System
VALS	Vertical Alert Service
VLL	Very-Low-Level
VLOS	Visual Line Of Sight
VPL	Vertical Protection Level
VPN	Virtual Private Network
WALB	Width Alert Buffer

2 USSP Module

There are two functions of USSP platforms which may interact with the ICARUS CARS platform and either use its services or provide input data for the ICARUS platform itself.

A) Telemetry:

- For reporting positions of all aircraft – the USSP can collect telemetry data from various sources, among them from the GNSS module of the ICARUS CARS platform
- For publishing reliable information about aircraft positions – the ICARUS CARS system will be fed with reliable telemetry data provided by the USSP system, previously decorated through the VCS service

B) Operational Flight Plans

The ICARUS CARS GIS service will be used to validate the declared mission height against a reliable DSM model to calculate altitude (absolute height).

These two main functionalities are described in more detail from the process and system design perspective in the following sections.

2.1 Telemetry functionality

The implementation of the telemetry functionality in the Droneradar USSP system, used to validate the ICARUS CARS environment, is aligned with design and functional specifications released by the GOF 2.0 consortium [1].

As defined in these specifications and shown in Figure 2, the Traffic/Telemetry service is fed with surveillance data from a number of data sources (Position Reporters). As an be seen in the diagram, the telemetry position reporting module will implement the PositionReportSubmissionInterface. It will also allow telemetry data to be gathered from different sources. In example given in Figure 1, the Droneradar USSP can be supplied with telemetry data based on barometric height measurements coming from a network of sensors (e.g., ADSB), directly from GNSS receivers on-board the UAVs (GNSS module provides raw data from UAS), or from the ICARUS CARS GNSS module (via the TCU.USSP.01 interface). In the last case, the raw GNSS data will be pre-processed by the CARS system to validate the measured geolocation and assure integrity of the measurements.

Whenever any system or application receives a telemetry feed, it needs to subscribe to the TrafficTelemetrySubscriptionInterface of the Tracker module. After successful subscription, the consumer will be supplied with traffic data streams through its exposed TrafficTelemetryNotificationInterface. As soon as the subscription expires e.g., due to the unsubscribe request, the telemetry data transmission is stopped for the defined endpoint. One such telemetry data consumer is a GIS module of the ICARUS CARS platform. The USSP.GIS.01 interface was defined to integrate the GIS module and telemetry.

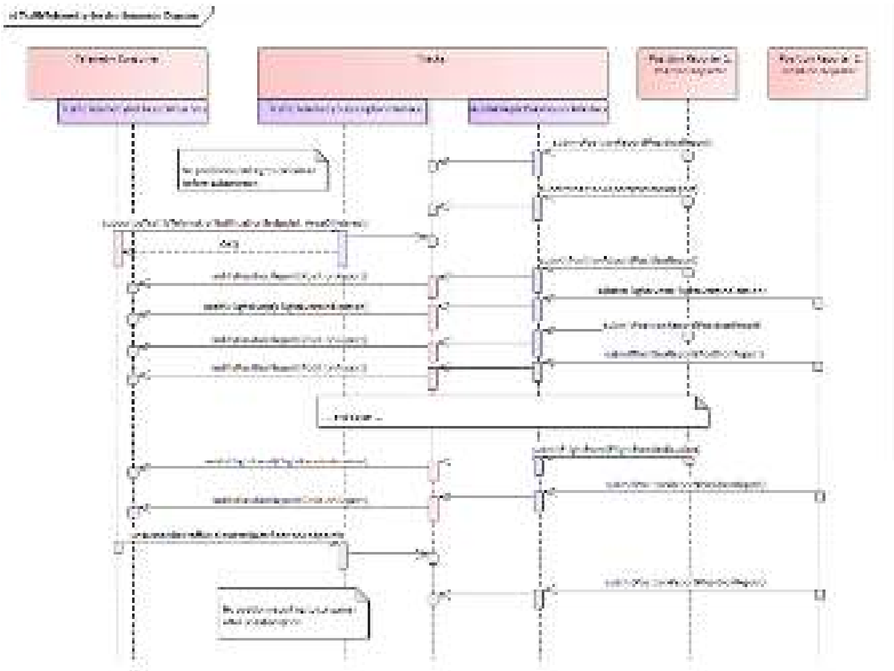

Figure 2 Telemetry Service flow as defined by GOF 2.0

The ICARUS CARS allows this telemetry processing flow to be modified and enriched. In the modified flow we introduce an additional step. The telemetry decoration step is added between collecting the telemetry data and publishing it. For this, we use the VCS (Vertical Conversion Service) of the ICARUS CARS. This is accessed via the VCS.USSP.01/USSP.VCS.01 interface. The VCS service of the ICARUS CARS allows telemetry data to be decorated with additional calculated height information. VCS is a module that performs mathematical calculations, described in [2]. Since the VCS conversion is called (requested) for every location update, it must be efficient. For instance, to serve five aircraft sending telemetry at a 5Hz interval, the performance of altitude conversion must be at least capable of handling 5 x 5 = 25 requests per second. The efficiency of the VCS is understood as the ability to scale with increasing traffic. It should be remembered that the VCS conversion is a safety critical conversion, hence its operational use requires high scalability and availability.

Figure 3 shows an example of such telemetry data decorated by VCS. This telemetry decoration step allows the measured values to be enriched with the unified, translated heights, calculated based on reliable reference information and all available calibration data. Finally, this would ensure that published telemetry data is appropriately adjusted, normalised, and validated.

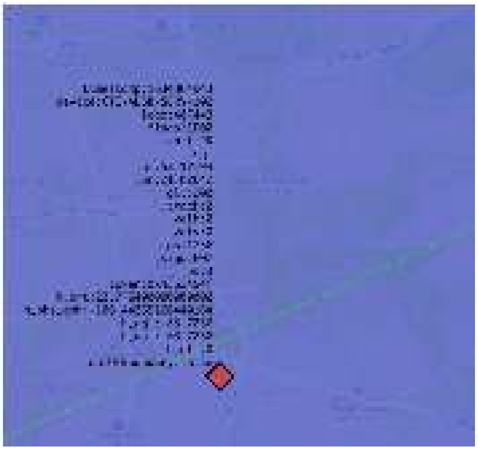

Figure 3 Example of decorated telemetry data on Droneradar USSP

For the prototype demonstration, the overall telemetry data processing related flow is implemented as shown in Figure 4. It differs from the target, describe above the flow – it has been modified and simplified due to a lack of real implementations of the GOF 2.0 recommendations at the time of preparation of this document.

The solution supports the possibility of acquiring traffic data from different sources, but the data is not yet reported through the real implementation of the PositionReportSubmissionInterface - there are no exposed interfaces available at this point. The individual data from different data sources is currently loaded directly into the system's database (RedisDB). The Droneradar USSP system natively relies on telemetry based on ADS-B. First few steps on the diagram show how the telemetry data is transmitted from the on-board transmitter to an ground receiver and then to the location database.

There is a dedicated "icarus-api" service in the Droneradar USSP system to communicate with the ICARUS CARS services. It is invoked each time the ICARUS CARS service is called. To collect all the necessary data required by VCS service, such as QNH data and weather station data, the icarus-api interrogates the Weather Service Provider module (WSP) via the defined WSP.VCS.01 interface (the same interface is used to call the WSP by the VCS itself). As soon as the data package is ready, the actual request is forwarded to the VCS module. The VCS module computes all height transformations and sends decorated telemetry information back. This new information is stored in the locations database.

After storing the decorated telemetry data in the database, it becomes available to all telemetry consumers by accessing the database directly (pulling information). There is currently no subscription-based mechanism available to publish this information. When it is defined and deployed, the USSP.GIS.01 interface should implement this interface.

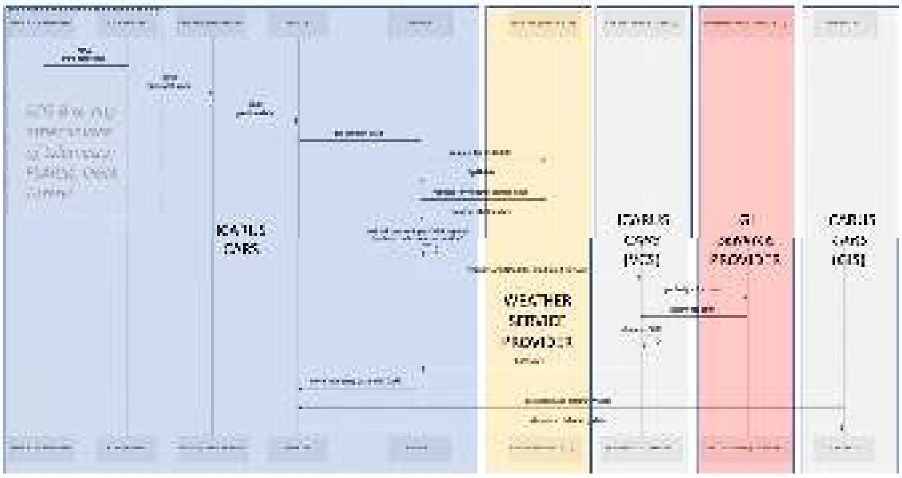

Figure 4 The end-to-end flow of telemetry information (Baro -> Ellipsoid)

2.1.1 Telemetry-related integration interfaces

The following, telemetry integration-related interfaces have been identified between the USSP and the ICARUS CARS systems:

- The TCU.USSP.01 interface
- The VCS.USSP.01/USSP.VCS.01 interface (Barometric -> Ellipsoid and Ellipsoid -> Barometric implementations)
- The USSP.GIS.01 interface

Of these, only the VCS.USSP.01 has appropriate functional support on the Droneradar USSP side and thus it is the only one currently implemented and described below. The two remaining interfaces are scoped in [2] but not yet defined and implemented.

2.1.1.1 VCS.USSP.01/USSP.VCS.01 (Barometric -> Ellipsoid version)

In the VCS.USSP.01 scenario, a barometric altitude is converted to ellipsoid altitude. A single request to the VCS service should contain the following information:

```
{
          "VehicleType":  1 - aircraft (height from barometric altimeter)
          "Lat": Latitude
          "Lon": Longitude
          "HObsQne": barometric height in [m]
          "Pw": Pressure of the nearest weather station
          "Hw": Height of the nearest weather station
          "PQnhAirport": QNH Preassure
}
```

The VehicleType should be set according to the type of altimeter used (0-GNSS, 1-barometric).

Lat, Lon and HObsQne (for barometric altimeters) values are measured and should be taken from the telemetry data.

The Pw, Hw and PQnhAirport values can be fetched from the WSP based on the measured Lat and Lon.

An example request and response:


```
"input": {
                "vehicle_type": 1,
                "lat": 52.17801,
                "lon": 21.02615,
                "h_obs_qne": -22.86,
                "p_w": 1019,
                "h_w": 104.22,
                "p_qnh_airport": 1027
        }
"altitude": {
                "h_ort": 129.02938226761734,
                "h_obs_qnh": 90.74118291365912,
                "h_agl": 23.59938226761733,
                "h_asl": 23.59938226761733,
                "h_ell": 166.21298226761735
        }
```


For demonstration purposes, a simplification has been adopted in which the horizontal distance of the measuring sensor from the aircraft is negligible. In subsequent tests, an attempt will be made to average the pressure values between the measurement points and calculate the maximum error value resulting from the lack of knowledge of the exact pressure at the place where an air system is.

2.1.1.2 VCS.USSP.01/USSP.VCS.01 (Ellipsoid -> Baro version)

In this case, an ellipsoid altitude is converted to barometric altitude. A single request to the VCS service should contain the following information:

```
{
          "VehicleType":  0 - drone (height from GNSS)
          "Lat": Latitude
          "Lon": Longitude
          "H_ell": GNSS height in [m]
          "Pw": Pressure of the nearest weather station
          "Hw": Height of the nearest weather station
          "PQnhAirport": QNH Preassure
}
```

2.2 Drone Flight Plan functionality

During the flight plan analysis (strategic level), the USSP system will query the GI Service via the GIS.USSP.01 interface to obtain the maximum mission flight elevation.

GI service input:

1) A polygon representing the Operational Flight Plan
2) The declared height of the flight

The GI Service, using the validated DSM model, calculates the actual maximum absolute height of the planned flight, resulting in the maximum operational flight plan altitude. The value of the maximum operational flight plan altitude is used to find the relative distance of the planned mission from the aeronautical airspaces, whose vertical limits are most often defined by altitudes (absolute heights)..

It is assumed that all USSPs and the CIS should rely on validated and reliable DSM service providers. There should be a national authority responsible for granting authorisations to DSM service providers.

2.2.1.1 GIS.USSP.01

This interface is used to pass the Operational Flight Plan polygon from the USSP and receive in turn the calculated maximum elevation of the area.

3 GI Service Provider Module (GISP)

For the prototype implementation of the ICARUS CARS, the GI Service Provider (GISP) provides the elevation information for a given point or area based on the built-in DTM and DSM models. Currently the DTM/DSM model itself is not provided through the available API. In future it will be available via GeoTIFF files.

3.1.1 DTM/DSM related integration interfaces

3.1.1.1 GISP.GIS.01

A request for information about terrain elevation can refer to a single point or to a polygon defined by a series of points. In case of a single point the elevation returned applies to this point. In case of a polygon, the values returned represent the minimum and maximum values of the elevation within the defined polygon.

This interface should be highly scalable to support 1000s of requests per second (for point type of request).

An example request and response for a point:


```
{
  "geofeature": {
           "type": "Feature",
           "geometry": {
                "type": "Point",
                "coordinates": [
                  16.914957457864695,
                  52.39664053545001
                ]
           },
           "properties": {}
   }
}
{
  "data": {
    "status": "success",
    "result": {
      "error": 0.1,
      "max_val": null,
      "min_val": null,
      "height": 84.06
    }
  }
}
```


All units are in metres.

An example request and response for a polygon:

```
{
  "etc": {
          "name": "test minmax payload"
  },
  "geofeature": {
           "type": "Feature",
           "geometry": {
                "type": "Polygon",
                "coordinates": [
                        [
                                [16.1234953, 53.0917227],
                                [16.1230704, 53.0916719],
                                [16.1227101, 53.0915274],
                                [16.1224694, 53.091311],
                                [16.1223849, 53.0910558],
                                [16.1224694, 53.0908006],
                                [16.1227101, 53.0905843],
                                [16.1230704, 53.0904397],
                                [16.1234953, 53.090389],
                                [16.1239203, 53.0904397],
                                [16.1242805, 53.0905843],
                                [16.1245212, 53.0908006],
                                [16.1246058, 53.0910558],
                                [16.1245213, 53.091311],
                                [16.1242805, 53.0915274],
                                [16.1239203, 53.0916719],
                                [16.1234953, 53.0917227]]
                        ]
                },
                "properties": { }
                }
}
{
  "data": {
    "status": "success",
    "result": {
      "error": 0.15,
      "max_val": 79.01,
      "min_val": 64.64,
      "height": null
    }
  }
}
```

4 Weather Service Provider Module (WSP)

The Weather Service Provider (WSP) provides two types of information:

- Reliable information about current QNH pressure values for a given location
- Readings from weather sensors: measured pressure and temperature

The QNH information is generated according to aviation rules:

- Firstly, if available, the CTR QNH for given area below a CTR and/or TMA

- Secondly, if the above is not available, the regional QNH is provided

- Lastly, if none of above, the contingency QNHP is provided

The WSP also provides pressure and temperature measurements from local weather sensors – based on the given location, it will return the current readings from the nearest sensor.

4.1.1 WSP-related integration interfaces

4.1.1.1 WSP.VCS.01

The WSP.VCS.01 interface provides several services:

- List of available QNH regions
- Listing information about a QNH region using its name
- Fetching QNH data based on the location

Depending on the activity in the QNH region, different responses will be sent.

Examples of supported queries:

A) List of all QNH regions defined in the system

```
curl -X GET "https://qnh.ica2.droneradar.xyz/qnh/regions" -H  "accept: application/json"
```

Response

Commented [PH1]: Could be good to abbreviate this list by giving the first two regions and the last, separated by "...", rather than the whole list


```
{
  "data": [
    {a
      "Name": "QNH01",
      "Description": "QNH region 01",
      "IsActive": "true"
    },
    {
      "Name": "QNH05",
      "Description": "QNH region 02",
      "IsActive": "true"
    },
    …
    {
      "Name": "QNHEPKT",
      "Description": "CTR Katowice",
      "IsActive": "true"
```

```
    }
  ],
  "type": "success"
}
```


B) Fetching a QNH region's data using its name – the same as for fetching QNH and weather sensor data

curl -X POST "https://qnh.ica2.droneradar.xyz/qnh/region" -H "accept: application/json" -H "Content-Type: application/json" -d "{ \"Name\": \"QNH04\"}"

Response


```
{
        "data": {
                "Name": "QNH04",
                "Pressure": "999",
                "PressureUnit": "hPa",
                "Temperature": "10.8",
                "TemperatureUnit": "C",
                "Height": "0",
                "HeightUnit": ""
        },
        "type": "success"
}
```


Example response from a weather sensor:


```
{
        "name": "sh04",
        "lat": 52.17801,
        "lon": 21.02615,
        "temperature": 28.15389633178711,
        "pressure": 1014.7744140625,
        "elevation": 104,3
        "elevation_reference": "elipsoide"
}
```


C) Fetching the QNH data based on location – the response will vary depending on the current region's activity

curl --location --request POST '127.0.0.1:5002/qnh/point' --header 'Content-Type:application/json' --data-raw '{"Point": {"type": "Point","coordinates":[52.229675, 21.012230]}}'

Response


```
{
        "data": {
                "Name": "QNHEPWA",
                "Pressure": "1001",
                "PressureUnit": "hPa",
                "Temperature": "9.7",
                "TemperatureUnit": "",
                "Height": "0",
                "HeightUnit": ""
        },
        "type": "success"
}
```

5 Testing and validation approach

In this section, a description of test cases will be provided, with the main goal of validating the interoperability of the Droneradar USSP, WSP and GISP with the ICARUS CARS platform.

The following tests will be detailed:

1) Test connection and communication between the Droneradar USSP and the VCS module of ICARUS CARS
2) Test connection and communication with the WSP
3) Test connection and communication with the GISP

Connection of the VCS with the USSP	
Test case Id	*USSP.01*
Test designed by	*Adam Staszak, Piotr Dybiec*
Test executed by	*Adam Staszak, Jan Mróz-Jaworski*
Testing date	*IX-X 2021*
Test purpose	*The test aims to: 1) check the communication between the USSP and the VCS microservice 2) verify the ability to send the request to the VCS endpoint and receive the response*
Test environment	*The test environment involves the use of the following components: 1) Test instance of the Droneradar USSP system 2) VCS microservice hosted within the Telespazio facilities*
References	*D5.1 Ch. 2.1.1.1*
Validation target/metric	*The expected results are: 1) Establish a connection between the Droneradar USSP and the VCS endpoints 2) Successful submission of the request and reception of the response (at this stage values provided are not validated, if they are correct)*
Tested components	*DRONE RADAR USSP, VCS*
Tested interfaces	*VCS.USSP.01*

Other interfaces	*N.A.*		
Pre-test conditions	*There should be a UAS traffic generator available to generate the telemetry data. The VCS service should have the access to appropriate DSM data. The GISP and WSP services should be connected to the VCS and the USSP.*		
Test tools	*Droneradar USSP*		
Integration test steps Definition and results Log	**# Step**	**Description**	**Result**
	1	*Establish a connection between the Droneradar USSP and the VCS*	*Passed*
	2	*For the given telemetry data set, send a request to the VCS to perform telemetry decoration*	*Partially Passed*
	3	*Receive the response from the VCS with decorated telemetry data*	*Passed*
Test verdict	PASS/ FAIL/ PENDING	• *The connection between the USSP and the VCS was successfully established* • *The request to the VCS was sent, although the tested implementation of the VCS had very limited performance – it should be improved* • *The response back was received correctly but the payload contained numerical errors. The algorithm/application needs to be corrected.*	
Change requests		*Improve interface/service performance beyond 1 req/5 sec, e.g., to 1 req/sec (for testing purposes)*	
Defects		*The recalculating algorithm and/or implementation must be adjusted to provide correct calculations*	

Connection with the WSP	
Test case Id	*WSP.01*
Test designed by	*Adam Staszak, Piotr Dybiec*

Test executed by	*Adam Staszak, Jan Mróz-Jaworski*		
Testing date	*IX-X 2021*		
Test purpose	*The test aims to:* *1) check the communication with the Weather Service Provider platfo* *2) check the communication with the weather station sensor*		
Test environment	*The test environment involves the use of the following components:* *1) Weather Service Provider platform* *2) Weather station sensor*		
References	*D5.1 Ch. 4.1.1.1*		
Validation target/metric	*The expected results are:* *1) Establish a connection to the WSP endpoint* *2) Successful submission of the request and reception of the response related to QNH and measured pressure by the weather sensor (at this stage values provided are not validated, if they are correct)*		
Tested components	*WSP*		
Tested interfaces	*WSP.VCS.01*		
Other interfaces	*N.A.*		
Pre-test conditions	*The WSP platform must be up and configured with QNH-related data for the given area (country). The weather sensor should be up and running.*		
Test tools	*Command line tool such as curl*		
Integration test steps Definition and results Log	**# Step**	**Description**	**Result**
	1	*Establish a connection to the WSP platform*	*Passed*
	2	*Send a simple query to the WSP related to QNH and receive the response* curl --location --request POST '127.0.0.1:5002/qnh/point' --header 'Content-Type:application/json' --data-raw '{"Point": {"type": "Point","coordinates":[52.229675, 21.012230]}}'	*Passed*

	3	*Send a simple query (by name) to the WSP related to weather sensor and receive the response* curl - X POST "https://qnh.ica2.droneradar.xyz/qnh/region" -H "accept: application/json" -H "Content-Type:	*Passed*
Test verdict	PASS/ FAIL/ PENDING	• *Integration tests with the WSP for both QNH queries and weather sensor queries were passed*	
Change requests		*N.A.*	
Defects		*N.A.*	

Connection with the GISP	
Test case Id	*GISP.01*
Test designed by	*Adam Staszak, Piotr Dybiec*
Test executed by	*Adam Staszak, Jan Mróz-Jaworski*
Testing date	*IX-X 2021*
Test purpose	*The test aims to check the communication with the GI Service Provider platform.*
Test environment	*The test environment involves the use of the following components:* *1) GI Service Provider platform* *2) Applicable DSM data for the test area*
References	*D5.1 Ch. 3.1.1.1*

<table>
<tr><td>Validation target/metric</td><td colspan="3">The expected results are:
1) Establish a connection to the GISP endpoint
2) Successful submission of the request and reception of the response related to an elevation query for a given point</td></tr>
<tr><td>Tested components</td><td colspan="3">WSP</td></tr>
<tr><td>Tested interfaces</td><td colspan="3">GISP.GIS.01</td></tr>
<tr><td>Other interfaces</td><td colspan="3">N.A.</td></tr>
<tr><td>Pre-test conditions</td><td colspan="3">The GISP platform is loaded with DSM data for area under test</td></tr>
<tr><td>Test tools</td><td colspan="3">Command line tool like curl</td></tr>
<tr><td rowspan="3">Integration test steps

Definition and results

Log</td><td># Step</td><td>Description</td><td>Result</td></tr>
<tr><td>1</td><td>Establish a connection to the GISP platform</td><td>Passed</td></tr>
<tr><td>2</td><td>Submit a request for a defined point and receive the response with the elevation data</td><td>Passed</td></tr>
<tr><td>Test verdict</td><td>PASS/
FAIL/
PENDING</td><td colspan="2">• Integration tests were passed</td></tr>
<tr><td>Change requests</td><td></td><td colspan="2">N.A.</td></tr>
<tr><td>Defects</td><td></td><td colspan="2">N.A.</td></tr>
</table>

6 Applicable and reference documents

[1] GOF2.0 D2.2 - Appendix A Traffic/Telemetry Service Specification, GOF Consortium, rel. 00.00.03

[2] ICARUS D4.2, “ICARUS Prototype”