Rules-Based Verification of Geospatial Feature Comprehension and Flight Authorization

Product definition

Summary

A number of jurisdictions need to validate service providers' comprehension of geospatial features (restrictions, areas where advisories are needed, etc) with respect to flight planning implications. InterUSS's automated test tool uss_qualifier seems well-suited to provide a common capability that spans many jurisdictions' needs in this area. This document describes a product (in the form of two uss_qualifier test scenarios) that provides this capability.

Background

InterUSS currently maintains a tool to conduct automated tests named uss_qualifier. This tool verifies compliance with requirements by interacting with one or more service providers like a virtual user and determining whether carefully-structured procedures result in expected outcomes. Service providers that want to be tested by uss_qualifier provide the means by which uss_qualifier can act as a virtual user for the service provider by implementing InterUSS-defined automated testing interfaces. The carefully-structured procedures are defined in test scenarios, broken down into test cases, test steps, and test checks associated with requirements, and collected into groups using test suites. Environment configuration and other test data is provided to test scenarios (via test suites, when applicable) via "resources".

Opportunity

Many jurisdictions need to verify compliance to similar requirements regarding service providers' comprehension of geospatial features. Some examples include:

- CASA (Australia) has operating rules where drone safety apps are obligated to "block" planning of flights under certain rule sets when certain types of geospatial features are present, and to "advise" operators when planning flights when other types of geospatial features are present.

- U-space (Europe) defines a mandatory geo-awareness service that provides “geozone” information to operators. Automated testing of USSPs’ implementation of this service could consist of verifying acquisition and comprehension of those geozones.
- FAA (United States) requires USSs providing LAANC service to block LAANC authorizations under certain geospatially-defined conditions and provide advisories to operators in other conditions.

To further harmonization and reduce burden on regulators and service providers, it is desirable to maximize reuse for Geospatial Feature Comprehension globally.

Assumptions

Even with global harmonizations, tests and associated geospatial data will be regional in nature and will need to be configured per region. It is intended that this configuration should not require changes to the core test functionality, and that configuration files could be managed outside of InterUSS repositories.

User Roles

This section provides an overview of the different user roles associated with the user journeys.

Jurisdiction Test Director

A person or group representing the authority (e.g. FAA, CASA, EASA, FOCA…) for a jurisdiction attempting to verify compliance with the jurisdiction’s geospatial feature comprehension and flight authorization requirements.

Test Configuration Manager

A person or group directed by the authority to create an automated test to verify compliance with the jurisdiction’s geospatial feature comprehension and flight authorization requirements. There is nothing preventing a Jurisdiction Test Manager from also performing the duties of a Test Configuration Manager.

Service Provider

Also referred to as ‘Drone Safety Apps’, ‘U-Space Service Providers (USSPs)’, or UAS Service Providers (USSs). These are the companies or organizations wishing to provide services in compliance with the jurisdiction’s geospatial feature comprehension and flight authorization requirements.

User journeys

UJ 1: Test configuration manager configures geospatial feature comprehension checks to verify the requirements of their jurisdiction

The Jurisdiction Test Director will direct the Test Configuration Manager to create an automated test to verify compliance with the jurisdiction's geospatial feature comprehension requirements. To verify each capability defined for the jurisdiction, this Test Configuration Manager will create a "geospatial feature check table" consisting of a list of "feature check" rows, each of which will evaluate expected outcomes from geospatial feature comprehension in a particular situation. This feature check table can be version-controlled by the Test Configuration Manager apart from InterUSS. The uss_qualifier geospatial feature comprehension test scenario will accept this feature check table as a resource input to that scenario.

The Test Configuration Manager will generally build a test baseline including the feature check table they have defined by also creating a test suite which defines one or more capabilities characterized by testing one or more groups of requirements defined in their feature check table (see UJ 3). The Test Configuration Manager may design such a suite to execute multiple test scenarios, each with a different feature check table, or they may design the test suite with a single test scenario using a single feature check table.

Requirements

1. uss_qualifier geospatial feature comprehension test scenario must accept a resource containing
    a. A table ("geospatial feature check table") containing a list of feature checks with the information specified in Annex 1 (required; may not be empty)
2. When uss_qualifier executes a geospatial feature comprehension test scenario, each feature check must be executed in the provided sequence.
3. For each feature check in the geospatial feature check table, for each service provider being tested as a geo-awareness map provider (see UJ 5):
    a. If the feature check's expected_result is "Block" or "Advise":
        i. Query service provider under test (see Annex 2) for any geospatial features which satisfy all the filter criteria in the feature check with the query's resulting_operational_impact as the feature check's expected_result
        ii. Record passing test check if service provider under test reports that there are one or more matching geospatial features (otherwise record failure)
    b. If the feature check's expected_result is "None": (*Note: "None" expected result is achieved by a negative response to the USS when asked for "BlockOrAdvise"*)

i. Query service provider under test (see Annex 2) for any geospatial features which satisfy all the filter criteria in the feature check with the query's resulting_operational_impact as "BlockOrAdvise"
ii. Record passed test check if service provider under test reports that there are no matching geospatial features (otherwise record failure)

UI Mock

The below code block is an example of how a single geospatial feature check could be defined.

Unset

```
- geospatial_check_id: 1
  requirement_id:
    - HBY0010
  description: Testing flights in conflict with [ASD0005] Restricted Airspace
results in BLOCK.
  operation_rule_set: HBY
  volumes:
    - outline_circle:
        lng: 150.731327
        lat: -33.804542
        radius:
          value: "1"
          units: Meters
      start_time:
        offset_from:
          base_time:
            next_day_of_week:
              base_time:
                start_of_test:
              day_of_week: ["M", "T", "W", "Th", "F", "Sa", "Su"]
          offset_time: T12h
      duration: T1M
      max_altitude:
        value: 120
        units: Meters
        reference: GroundLevel
  restriction_source: CASA
  expected_result: Block
```

UJ 2: Test configuration manager configures a flight authorization test for their jurisdiction

The Jurisdiction Test Director will direct the Test Configuration Manager to create an automated test to verify compliance with the jurisdiction's flight authorization requirements. To verify each capability defined for the jurisdiction, this test configuration manager will create a "flight check table" consisting of a list of "flight check" rows, each of which evaluates expected outcomes from attempting to obtain a flight authorization in a particular situation. This flight check table can be version-controlled by the test configuration manager apart from InterUSS. The uss_qualifier flight authorization test scenario will accept this flight check table as a resource input to that scenario.

The Test Configuration Manager will generally build a test baseline including the flight check table they have defined by also creating a test suite which defines one or more capabilities characterized by testing one or more groups of requirements defined in their flight check table. The Test Configuration Manager may design such a suite to execute multiple test scenarios, each with a different flight check table, or they may design the test suite with a single test scenario using a single flight check table.

Requirements

1. uss_qualifier flight authorization test scenario must accept a resource containing:
    a. A table ("flight check table") containing a list of flight checks with the information specified in Annex 3 (required; may not be empty)
2. When uss_qualifier executes a flight authorization test scenario, each flight check must be executed in the provided sequence.
3. When uss_qualifier executes a flight authorization test scenario which contains a flight check table, for every service provider being tested as a flight planner:
    a. uss_qualifier must attempt to create a flight in the service provider under test according to the `FlightRequest` specified in Annex 4
    b. uss_qualifier must record a passed test check if the response from the service provider under test matches the expected behavior, or record a failed test check otherwise

UI Mock

The below code block is an example of how a single flight check could be defined.

```
Unset
- flight_check_id: 1
  requirement_id:
    - AA0005
    - AA0055
```

```
  description: Submission of an authorisation request in an area wholly within
the GCD.
  operation_rule_set: ReOC
  expect_to_be_accepted: true
  conditions_expectation: Irrelevant
  volumes:
    - outline_circle:
        lng: 149.147236
        lat: -35.323009
        radius:
          value: 100
          units: Meters
      start_time:
        offset_from:
          base_time:
            next_day_of_week:
              base_time:
                start_of_test: {}
              day_of_week: ["M", "T", "W", "Th", "F", "Sa", "Su"]
          offset_time: T12h
      duration: T3H
      max_altitude:
        value: 40
        units: Feet
        reference: GroundLevel
  operation_details:
    uas_serial_number: "7834JHG9999123"
    uas_registration_number: ""
    operator_registration_number: "9981234"
    au_information:
      flight_profile: MANUAL
      pilot_phone_number: "0412345678"
      aircraft_type: ROTORCRAFT
      reoc_operator_number: "9798"
```

UJ 3: Test configuration manager configures a test that verifies participant capabilities

The Jurisdiction Test Director will direct the Test Configuration Manager to create an automated test to verify compliance with the jurisdiction's geospatial feature comprehension and flight authorization requirements. Each jurisdiction will have a unique set of capabilities (i.e. "Airspace Awareness Map" and "Airspace Authorisation" in AU). For each capability there are a set of criteria, including successful checks related to requirements, that each service provider must meet to verify compliant provision of that capability. This mapping from requirements/criteria to capabilities is referred to as a "capability definition." Some requirements might support multiple capabilities, but each capability will have a unique set of requirements/criteria that differentiate it from other capabilities. The capability definition can be version-controlled by the test configuration manager apart from InterUSS. The capability definition will be incorporated into the InterUSS test configuration, via the test suite definition.

Requirements

1. The uss_qualifier test suite must accept capability definitions as part of the configuration.
    a. The configuration will support the definition of multiple capabilities.
    b. The list of requirements which a particular participant must pass are defined for each capability individually.
2. At the conclusion of a test, uss_qualifer must list the capabilities that were successfully passed/checked for each participant as part of the test report.
    a. For each requirement listed in the individual capability's definition, if every listed requirement is checked and every check of every listed requirement passes, then the capability is considered verified.
    b. If any check involving any listed requirement fails, then the capability is not verified.
    c. Note that if a requirement is listed in a capability definition but omitted from a test scenario, then it will not be possible to verify that capability.
3. uss_qualifier must include the software version of each service provider tested as part of the test report

UI Mock

The below code block is an example of how two participant verifiable capabilities could be defined.

```
Unset
   participant_verifiable_capabilities:
  - id: "1"
    name: Airspace Awareness Map
    dscription: ""
```

```
    verification_condition:
      requirements_checked:
        checked:
          requirements:
            - HBY010
  - id: "2"
    name: Airspace Authorization
    description: ""
    verification_condition:
      requirements_checked:
        checked:
          requirements:
            - AA0005
            - AA0055
```

UJ 4: Test configuration manager configures individual service provider environment details

Each service provider will have their own testing injection endpoints for their respective applications. Also, a single service provider will have different injection endpoints for geospatial testing and flight authorization testing. Each service provider's endpoints need to be configured into the automated tests definition for the uss_qualifier to know where to send the test stimuli. The test configuration manager will receive the service provider injection endpoint information from each service provider and configure the automated test accordingly.

Requirements

1. The uss_qualifier test suite must accept service provider environment details (injection endpoints) as part of the configuration.
2. Changes in environment information (such as injection endpoints) must not change the test baseline identifier.

UI Mock

The below code block is an example of how a service provider environment configuration could be defined.

```
Unset
utm_auth:
  resource_type: resources.communications.AuthAdapterResource
```

```
  specification:
    environment_variable_containing_auth_spec: AUTH_SPEC
flight_planners:
  resource_type: resources.flight_planning.FlightPlannersResource
  dependencies:
    auth_adapter: utm_auth
  specification:
    flight_planners:
      - participant_id: service_provider_under_test
        injection_base_url: http://sp.example.com/flightPlanner
geospatial_map_provider:
  resource_type: resources.geospatial_map_provider.GeospatialMapProviders
  dependencies:
    auth_adapter: utm_auth
  specification:
    geospatial_map_provider:
      - participant_id: service_provider_under_test
        injection_base_url: http://sp.example.com/geospatial
```

UJ 5: Service provider is tested by a geospatial feature comprehension test as a geo-awareness map provider

A service provider wishing to be tested by a geospatial feature comprehension test scenario as a geo-awareness map provider must implement an InterUSS-defined API allowing uss_qualifier to act as a virtual user examining areas on a map when determining how to plan a flight with particular characteristics. The service provider will provide the test configuration manager with the specific environment details such as the injection endpoint. The InterUSS automated testing API for geo-awareness map providers must support this use case by satisfying the following requirements.

Requirements

1. InterUSS must define an API for service providers to implement which contains a “query” endpoint that mimics a service provider user examining a given area on the geo-awareness map
    a. The query endpoint must accept the request information defined in Annex 2
    b. The query endpoint must return a boolean result indicating whether any geospatial features relevant to the filter criteria are present on the geo-awareness map
2. InterUSS must include in its API for service providers an endpoint that allows InterUSS to determine what version of software is currently running

3. The endpoints must be secured by OAuth 2.0 access tokens similar in style to ASTM-standard access tokens, but using a scope unique to InterUSS map querying automated testing

UJ 6: Service provider is tested by a flight authorization test as a flight planner

A service provider wishing to be tested by a flight authorization test scenario as a flight planner must implement an InterUSS-defined API allowing uss_qualifier to act as a virtual user attempting to plan a flight with specified characteristics. The service provider will provide the test configuration manager with the specific environment details such as the injection endpoint. The InterUSS automated testing API for flight planners must support this use case by satisfying the following requirements.

Requirements

1. InterUSS must define an API for service providers to implement which contains an endpoint that mimics an attempt by a service provider user to create a flight with the specified characteristics
    a. This creation endpoint must accept the request information defined in Annex 4
    b. This creation endpoint must return
        i. Result of the attempt to create the flight [Planned, Rejected, Failed, NotSupported]
        ii. If Planned or Rejected:
            1. Whether any advisories/conditions were provided to the user
2. InterUSS must include in its API for service providers an endpoint that mimics an attempt by a service provider user to cancel or close a flight
    a. This deletion endpoint must accept the flight plan ID
3. InterUSS must include in its API for service providers an endpoint that allows InterUSS to determine what version of software is currently running
4. The endpoints must be secured by OAuth 2.0 access tokens similar in style to ASTM-standard access tokens, but using a scope unique to InterUSS flight planning automated testing

Annex 1: Feature check format

The feature check table will consist of an ordered list of rows, each row containing the following information.

Geospatial feature check fields

All fields are required unless otherwise specified.

Field [Type]	Description
geospatial_check_id [string]	Unique (within table) test step/row identifier.
requirement_id [List<string>]	Jurisdictional identifier of the requirement this test step is evaluating.
description [string]	Human-readable test step description to aid in the debugging and traceability.
operation_rule_set [string]	The set of operating rules (or rule set) under which the operation described in the feature check should be performed.
volumes [List<Volume4DTemplate>]	Spatial and temporal definition of the areas the virtual user intends to fly in. A service provider is expected to provide geospatial features relevant to any of the entire area specified and for any of the entire time specified.
restriction_source [string]	Which source for geospatial features describing restrictions should be considered when looking for the expected outcome.
expected_result [enum]	["Block", "Advise", "None"] "Block": When a service provider being tested as a geo-awareness map provider is queried for whether any features are present for the specified volumes that would cause the flight described in this feature check to be blocked, the service provider must respond affirmatively; responding negatively will cause a failed check. "Advise": When a service provider being tested as a geo-awareness map provider is queried for whether any features are present for the specified volumes that would provide an advisory to the viewer viewing a map relevant to the planning of the flight described in this feature check, the service provider must respond affirmatively; responding negatively will cause a failed check. The service provider does not need to include the content or number of advisories in its response. "None": When a service provider being tested as a geo-awareness map provider is queried for whether any features matching the other criteria in this feature check and causing a “block” or “advise” per above are present with the specified criteria, the service provider must respond negatively; responding affirmatively will cause a failed check.

Annex 2: Map query format

The map query to a USS under test (a query for whether the USS under test has matching geospatial features) will consist of a request containing the following information.

MapQuery fields

Field [Type]	**Description**
volumes [List<Volume4D>]	Spatial and temporal definition of the areas the virtual user intends to fly in. A service provider must only consider in its response geospatial features relevant to a flight in this area for this time.
restriction_source [string]	Which set of restrictions should be considered; geospatial features not originating from this restriction source should be omitted from results.
operation_rule_set [string]	Only geospatial features which are applicable to flights operating under this jurisdictional rule set should be considered.
resulting_operational_impact [enum]	["Block", "Advise", "BlockOrAdvise"] "**Block**": Limit response to any features that would cause an operation as described to be blocked. "**Advise**": Limit response to any features that would provide an advisory for an operation as described. "**BlockOrAdvise**": For the described operation, limit response to any features that would cause an operation as described to be blocked, or that would provide an advisory for an operation as described.

Annex 3: Flight check format

The flight check table will consist of an ordered list of rows, each row containing the following information.

FlightCheck fields

All fields are required unless otherwise specified.

Field **[Type]**	**Description**
flight_check_id [string]	Unique (within table) identifier for this flight check/row.
requirement_id [List<string>]	Identifier of the jurisdiction's requirement that would be violated if this feature check failed.
description [string]	Human-readable description of the check being performed in this row to aid in debugging and traceability of failures.
operating_rules [string]	Jurisdiction-specific name of the operating rules under which this feature check should be performed. For instance, a UAS in the United States may be operated under FAR Part 107. A UAS in Australia may be operated under a remotely-piloted aircraft operator certificate (ReOC). A UAS in U-space may be operated as a normal-priority flight. Most jurisdictions will have more than one set of operating rules.
expect_to_be_accepted [enum]	**No**: When a flight planner service provider is requested to accept the flight described in this step, the service provider must decline to create the flight. Accepting the flight successfully will cause a failed check. **Yes**: The flight planner service provider must successfully accept the flight requested in this step (conditions or advisories may be indicated). Failing to accept the flight successfully will cause a failed check. **Either:** The service provider may choose to accept the flight or not. Presumably this option would be accompanied by a specific conditions_expectation to ensure that conditions were present (or absent) if the flight were accepted.
conditions_expectation [enum]	**Irrelevant**: Whether conditions accompanying the flight planning attempt are present is irrelevant to this feature check (default). **MustBePresent**: If the flight is accepted, it must be accompanied by some conditions/advisories. If the flight is not accepted, whether conditions are present is irrelevant. **MustBeAbsent**: If the flight is accepted, it must be unconditional (no accompanying conditions). If the flight is not accepted, whether conditions are present is irrelevant.

Field [Type]	Description
volumes [List<Volume4DTemplate>]	Spatial and temporal definition of the areas the virtual user intends to fly in. A service provider is expected to only successfully accept the flight if the user is authorized to fly in the entire area specified for the entire time specified.
operation_details [Operation Details]	Jurisdiction-required operation details (details a user in that jurisdiction might need to provide to obtain a flight authorization).

Annex 4: Flight planning request format

The request for a USS under test to attempt to authorize a flight will consist of the following information.

FlightRequest fields

Field [Type]	Description
volumes [List<Volume4D>]	Spatial and temporal definition of the areas the virtual user intends to fly in. A service provider must only successfully accept the flight if the user is authorized to fly in the entire area specified for the entire time specified.
operation_details [OperationDetails]	Jurisdiction-required operation details (details a user in that jurisdiction might need to provide to obtain a flight authorization).

Annex 5: Common elements

Volume4D fields

Field [Type]	Description
outline_polygon [List<lat, lng>]	Polygonal 2D outline/footprint of the specified area (may not be defined if outline_circle is defined)
outline_circle [lat, lng, radius]	Circular 2D outline/footprint of the specified area (may not be defined if outline_polygon is defined)
start_time	The time at which the virtual user may start using the

[DateTime]	specified geospatial area for their flight.
end_time [DateTime]	The time at which the virtual user will be finished using the specified geospatial area for their flight.
min_altitude [Altitude]	The minimum altitude at which the virtual user will fly while using this volume for their flight.
max_altitude [Altitude]	The maximum altitude at which the virtual user will fly while using this volume for their flight.

Operation Details fields

Field [Type]	**Description**
uas_serial_number [string]	Serial number of the UAS, if specified by the virtual user.
uas_registration_number [string]	Registration number assigned to the UAS by the jurisdictional authority, if specified by the virtual user.
operator_registration_number [string]	Registration number assigned to the operator of the flight by the jurisdictional authority (including, e.g., pilot license number), if specified by the virtual user.

<jurisdiction code>_information [<as defined by jurisdiction>]	Any additional information required from users for particular jurisdictions will be provided in fields named according to the jurisdiction and containing fields defined by that jurisdiction. The property names and expected property value types must be defined by the jurisdiction in which this test is being conducted. Example:
	`"au_information": {`
	`"flight_profile": "AUTOMATED_GRID",`
	`"pilot_phone_number": "+61 491 570 313",`
	`"aircraft_type": "AEROPLANE"`
	`}`

Altitude fields

Field [Type]	**Description**
value [float]	Vertical distance above the reference datum in the specified units
units [enum]	Units of measure for the vertical distance (“Meters”, “Feet”)

reference [enum]	Reference datum relative to which vertical distance is measured (“GroundLevel”, “WGS84Ellipsoid”)

TestTime fields

Exactly one of the fields below must be specified

Field [Type]	**Description**
absolute_time [DateTime]	Use a precise timestamp which does not change with test conditions. *The value of absolute_time is limited given that the specific time a test will be started is unknown, and the jurisdictions usually impose a limit on how far in the future an operation can be planned.*
start_of_test [<None>]	If specified, use the timestamp at which the current test run started.
next_day_of_week [TestTime, List<Day of week enum>]	Use a timestamp equal to midnight beginning the next occurrence of any of the specified days of the week following the specified reference timestamp.
next_sun_position [TestTime, angle above horizon]	Use a timestamp equal to the next time after the specified reference timestamp at which the sun will be at the specified angle above the horizon. Sun angle calculations will be based on a location relevant to the query (e.g., within flight volumes), but is not guaranteed to precisely match any particular location.
offset_from [TestTime, TimeDelta]	Use a timestamp that is offset by the specified amount from the specified time.

Example TestTime usages (logical; not describing actual syntax, which would be JSON/YAML/object-based):

- sunrise = next_sun_position(next_day_of_week(start_of_test, [M, T, W, Th, F, Sa, Su]), 0)

- sunset = next_sun_position(offset_from(next_day_of_week(start_of_test, [M, T, W, Th, F, Sa, Su]), 12 hours), 0)

Volume4DTemplate fields

Field [Type]	**Description**

outline_polygon [List<lat, lng>]	Polygonal 2D outline/footprint of the specified area (may not be defined if outline_circle is defined). As defined by link.
outline_circle [lat, lng, radius]	Circular 2D outline/footprint of the specified area (may not be defined if outline_polygon is defined). Radius in Meters.
start_time [TestTime]	The time at which the virtual user may start using the specified geospatial area for their flight. (may not be defined if duration and end_time are defined)
end_time [TestTime]	The time at which the virtual user will be finished using the specified geospatial area for their flight. (may not be defined if duration and start_time are defined)
duration [iso8601 duration]	If only one of start_time and end_time is specified, then the other time should be separated from the specified time by this amount (may not be defined in both start_time and end_time are defined)
min_altitude [Altitude]	The minimum altitude at which the virtual user will fly while using this volume for their flight. (Optional and if not provided will be assumed to be ground level)
max_altitude [Altitude]	The maximum altitude at which the virtual user will fly while using this volume for their flight.

Appendix 1: Test creation flow