Chapter 6 — APIs Your USS Must Implement

This chapter describes all the HTTP endpoints that your USS must expose publicly. These are the endpoints that DECEA and other USSs will call against your server. They are divided into:

1. **Production endpoints** — required for all live operations.
2. **Testing/Homologation endpoints** — required for the validation process.

All endpoints must be accessible over HTTPS from the public internet. The base domain of your server becomes the `uss_base_url` registered in the DSS.

6.1 Production Endpoints (USS-to-USS)

These are defined in `utm.yaml` and `remoteid.yaml` and must be live during production operations.

6.1.1 `GET /uss/v1/operational_intents/{entityid}`

Purpose: Return the full details of one of your USS's Operational Intents to another USS or DECEA that is querying it.

Required scope (caller must have): `utm.strategic_coordination`

Response body:


```
{
  "operational_intent": {
    "reference": {
      "id": "2f8343be-6482-4d1b-a474-16847e01af1e",
      "manager": "your-uss-sub",
      "uss_availability": "Normal",
```

```
      "version": 3,
      "state": "Activated",
      "ovn": "9d158f59-80b7-4c11-9c0c-8a2b4d936b2d",
      "time_start": { "value": "2026-07-01T10:00:00Z", "format": "RFC3339" },
      "time_end": { "value": "2026-07-01T11:00:00Z", "format": "RFC3339" },
      "uss_base_url": "https://uss.yourcompany.com/utm",
      "subscription_id": "78ea3fe8-..."
    },
    "details": {
      "volumes": [ ... ],
      "off_nominal_volumes": [],
      "priority": 0,
      "flight_type": "BVLOS"
    }
  }
}
```

Key behaviors:

- The `ovn` in the reference is what other USSs collect to include in their `key` array.
- When the OIR is in `Nonconforming` or `Contingent` state, the `off_nominal_volumes` field must be populated.
- In `Contingent` state, `volumes` may be empty (only `off_nominal_volumes` applies).

6.1.2 `POST /uss/v1/operational_intents`

Purpose: Receive notifications from other USSs about Operational Intents in your subscription area. This is how you learn about new, updated, or deleted OIRs near your operations.

Required scope (caller must have): `utm.strategic_coordination`

Request body (OIR created or updated):

```
{
  "operational_intent_id": "2f8343be-6482-4d1b-a474-16847e01af1e",
  "operational_intent": {
    "reference": { ... },
    "details": {
      "volumes": [ ... ],
      "off_nominal_volumes": [],
```

```
    "priority": 0
   }
  },
  "subscriptions": [
   {
    "subscription_id": "78ea3fe8-...",
    "notification_index": 3
   }
  ]
}
```

Request body (OIR deleted):

```
{
  "operational_intent_id": "2f8343be-6482-4d1b-a474-16847e01af1e",
  "subscriptions": [
   {
    "subscription_id": "78ea3fe8-...",
    "notification_index": 5
   }
  ]
}
```

When `operational_intent` is absent, the OIR has been deleted.

Expected response: `204 No Content`

Key behaviors:

- Update your internal airspace state upon receiving this notification.
- Check whether the new/updated OIR or the deletion changes your own conflict situation.
- If a new high-priority OIR or Constraint conflicts with your active operation, begin deconfliction.

6.1.3 `GET /uss/v1/constraints/{entityid}`

Purpose: Return the full details of a Constraint managed by your USS. In Phase 1, only DECEA creates Constraints, so this endpoint is primarily implemented by **DECEA's USS**. However, it must also be implemented by all USSs for completeness and future phases.

Required scope (caller must have): `utm.constraint_processing`

Response body:

```
{
  "constraint": {
    "reference": {
      "id": "c036326c-...",
      "manager": "decea-uss",
      "version": 1,
      "ovn": "a1b2c3d4-...",
      "time_start": { "value": "2026-07-01T08:00:00Z", "format": "RFC3339" },
      "time_end": { "value": "2026-07-01T20:00:00Z", "format": "RFC3339" },
      "uss_base_url": "https://uss.decea.mil.br/utm"
    },
    "details": {
      "volumes": [ ... ],
      "type": "com.decea.restricted_area"
    }
  }
}
```

6.1.4 POST /uss/v1/constraints

Purpose: Receive notifications about new, updated, or deleted Constraints in your subscription area. These come from DECEA's USS (and in future phases, potentially other authorized entities).

Required scope (caller must have): `utm.constraint_management`

Request body (Constraint notification):

```
{
  "constraint_id": "c036326c-...",
  "constraint": {
    "reference": { ... },
    "details": { "volumes": [ ... ], "type": "..." }
  },
  "subscriptions": [
    { "subscription_id": "...", "notification_index": 1 }
  ]
}
```

When `constraint` is absent, the Constraint has been deleted.

Expected response: `204 No Content`

6.1.5 GET /uss/v1/operational_intents/{entityid}/telemetry

Purpose: Serve live telemetry for an OIR in `Nonconforming` or `Contingent` state. DECEA's monitoring systems and other USSs call this to track a drone that has deviated from its plan.

Required scope (caller must have): `utm.conformance_monitoring_sa`

Response body:

```
{
  "telemetry": {
    "time_measured": { "value": "2026-07-01T10:32:15Z", "format": "RFC3339" },
    "position": {
      "longitude": -45.876,
      "latitude": -23.210,
      "altitude": { "value": 115.2, "reference": "W84", "units": "M" },
      "accuracy_h": "HA10m",
      "accuracy_v": "VA10m"
    },
    "velocity": {
      "speed": 8.5,
      "units_speed": "MetersPerSecond",
      "track": 270.0
    }
  },
  "next_telemetry_opportunity": { "value": "2026-07-01T10:32:25Z", "format": "RFC3339" }
}
```

This endpoint must only be available when the OIR is in `Nonconforming` or `Contingent` state. You may return `404` in other states.

6.1.6 GET /uss/flights

Purpose: Return basic flight information for all active drones in a given geographic view area. This is the primary Remote ID polling endpoint.

Required scope (caller must have): `rid.display_provider`

Query parameters:

- `view` — Bounding box as `lat1,lng1,lat2,lng2` (southwest and northeast corners).
- `recent_positions_duration` — How many seconds of recent position history to include (optional).

Response body:


```
{
 "timestamp": { "value": "2026-07-01T10:32:15Z", "format": "RFC3339" },
 "flights": [
  {
   "id": "flight-uuid-here",
   "aircraft_type": "Helicopter",
   "current_state": {
    "timestamp": { "value": "2026-07-01T10:32:15Z", "format": "RFC3339" },
    "timestamp_accuracy": 0.1,
    "position": {
     "lat": -23.2071,
     "lng": -45.8750,
     "alt": 115.2,
     "accuracy_h": "HA10m",
     "accuracy_v": "VA10m",
     "extrapolated": false
    },
    "track": 270.0,
    "speed": 8.5,
    "speed_accuracy": "SA3mps",
    "vertical_speed": 0.0,
    "operational_status": "Airborne"
   },
   "recent_positions": [ ... ]
  }
 ],
 "no_isas_present": false
}
```

Update frequency: The data returned must reflect the drone's actual position, updated at most every **10 seconds**.

6.1.7 `GET /uss/flights/{id}/details`

Purpose: Return detailed information for a specific flight, including UAS identification and operator data.

Required scope (caller must have): `rid.display_provider`

Response body:

```
{
  "details": {
    "id": "flight-uuid-here",
    "uas_id": {
      "registration_id": "PR-XXXX"
    },
    "operator_id": "operator-registration-id",
    "operator_location": {
      "position": { "lat": -23.208, "lng": -45.878 },
      "altitude": { "value": 0, "reference": "W84", "units": "M" }
    },
    "operation_description": "Package delivery - Sector A",
    "auth_data": {
      "format": 0,
      "data": ""
    }
  }
}
```

6.2 Additional Required Endpoints

These endpoints are required by the qualification rules (`uss-qualification-rules.md`) and are called during homologation testing and normal operations.

6.2.1 GET /version

Purpose: Return the current deployed version of your USS software.

No authentication required (or use your standard validation).

Response body:

```
{
  "version": "1.4.2"
}
```

6.2.2 GET /diagnostics/time

Purpose: Return the current system time and NTP synchronization status. Used by DECEA to verify your system clock is properly synchronized.

Response body:

```
{
  "system_time": "2026-07-01T10:32:15.482Z",
  "ntp_sync": {
    "synchronized": true,
    "source": "ntp.decea.gov.br",
    "stratum": 2,
    "offset_ms": 38,
    "last_sync": "2026-07-01T10:32:05.000Z"
  },
  "timestamp_format": "ISO 8601"
}
```

6.2.3 POST /telemetry

Purpose: Accept drone telemetry data injection during DECEA's homologation testing. DECEA uses this to simulate drone position updates and test your conformance monitoring logic.

Request body:

```
{
  "flight_id": "a8c4af2a-6640-41d9-b8e7-719fd19a2fce",
  "aircraft_type": "Helicopter",
  "operational_intent_id": "<id-of-the-operational-intent>",
  "position": {
    "timestamp": { "value": "2026-07-01T10:32:00Z", "format": "RFC3339" },
    "lat": -23.207184,
    "lng": -45.875054,
    "alt": 115.0,
    "accuracy_h": "HA10m",
    "accuracy_v": "VA10m",
    "extrapolated": false
  },
  "operational_status": "Airborne",
  "track": 270.0,
  "speed": 10.0,
  "vertical_speed": 0.0,
  "test_metadata": {
    "scenario_id": "VOL4D-EXIT-REENTRY-59S",
    "event": "exit_volume",
    "t_offset_seconds": 0
  }
}
```

6.3 Homologation/Testing Endpoints

These endpoints are defined in `flights.yaml`, `injection.yaml`, and `versioning.yaml`. They are **only called by DECEA's automated testing framework** during homologation. You implement them, but they are not used in production operations.

6.3.1 Flight Planning Testing (`flights.yaml`)

These endpoints simulate the user experience of creating and managing flight plans, allowing DECEA's test framework to exercise your OIR creation and management logic.

Method	Path	Description
`GET`	`/status`	Return readiness status of your testing interface.
`POST`	`/clear_area_requests`	Instruct your USS to cancel all flight plans in a given area.
`PUT`	`/flight_plans/{flight_plan_id}`	Create or update a flight plan (simulates a user action).
`DELETE`	`/flight_plans/{flight_plan_id}`	Delete a flight plan.
`GET`	`/user_notifications`	Return notifications received by the virtual user.

Scopes:

- `interuss.flight_planning.direct_automated_test` — For test director operations (clear area, delete, status).
- `interuss.flight_planning.plan` — For virtual user operations (create/update, notifications).

6.3.2 Remote ID Data Injection (`injection.yaml`)

These endpoints allow DECEA's test framework to inject simulated drone telemetry directly into your USS, to test your Remote ID serving logic.

Method	Path	Description
`PUT`	`/tests/{test_id}`	Create a test: inject one or more simulated flights with telemetry sequences.
`DELETE`	`/tests/{test_id}/{version}`	Delete a test and remove all injected data.
`GET`	`/user_notifications`	Return notifications received by the virtual user during the test.

Scope: `rid.inject_test_data`

Example injection request:

```
{
  "requested_flights": [
    {
      "injection_id": "test-flight-001",
```

```
    "aircraft_type": "Helicopter",
    "telemetry": [
      {
        "timestamp": { "value": "2026-07-01T10:00:00Z", "format": "RFC3339" },
        "position": { "lat": -23.2071, "lng": -45.8750, "alt": 100.0 },
        "track": 90.0,
        "speed": 10.0,
        "vertical_speed": 0.0,
        "operational_status": "Airborne",
        "accuracy_h": "HA10m",
        "accuracy_v": "VA10m"
      }
    ],
    "details_responses": [
      {
        "effective_after": "2026-07-01T09:59:00Z",
        "details": {
          "id": "test-flight-001",
          "operator_id": "OP-TEST-123",
          "uas_id": { "registration_id": "PR-TEST01" }
        }
      }
    ]
  }
 ]
}
```

6.3.3 Versioning (`versioning.yaml`)

Method	Path	Description
`GET`	`/versions/{system_identity}`	Return the version of a specific system component.

Scope: `interuss.versioning.read_system_versions`

Example:

```
GET /versions/br.mil.decea.brutm.uss.v1
```

```
{
  "system_identity": "br.mil.decea.brutm.uss.v1",
  "system_version": "1.4.2"
}
```

6.4 Summary of All Required Endpoints

Endpoint	Type	When Required
`GET /uss/v1/operational_intents/{entityid}`	Production	Always
`POST /uss/v1/operational_intents`	Production	Always
`GET /uss/v1/constraints/{entityid}`	Production	Always
`POST /uss/v1/constraints`	Production	Always
`GET /uss/v1/operational_intents/{entityid}/telemetry`	Production	When OIR is Nonconforming/Contingent
`GET /uss/flights`	Production (Remote ID)	When any flight is Activated
`GET /uss/flights/{id}/details`	Production (Remote ID)	When any flight is Activated
`GET /version`	Operations	Always
`GET /diagnostics/time`	Operations	Always
`POST /telemetry`	Operations/Testing	During homologation
`GET /status`	Testing	Homologation
`POST /clear_area_requests`	Testing	Homologation
`PUT /flight_plans/{id}`	Testing	Homologation
`DELETE /flight_plans/{id}`	Testing	Homologation
`GET /user_notifications`	Testing	Homologation
`PUT /tests/{test_id}`	Testing (Remote ID)	Homologation
`DELETE /tests/{test_id}/{version}`	Testing (Remote ID)	Homologation
`GET /versions/{system_identity}`	Testing	Homologation

Revision #1
Created 12 July 2026 20:10:07 by João Pedro Favoretti
Updated 14 July 2026 13:19:47 by João Pedro Favoretti