Chapter 8 — Homologation

This chapter describes the homologation (validation) process that a company must complete before receiving a production API Key and being authorized to operate in the BR-UTM ecosystem.

8.1 What Is Homologation?

Homologation is DECEA's process for validating that a USS implementation:

- Correctly implements all required APIs (as defined in the OpenAPI specifications).
- Behaves correctly in all defined operational scenarios.
- Meets all non-functional requirements (timing, precision, synchronization).
- Is ready to operate safely alongside other USSs in the live ecosystem.

The process is **primarily manual** — conducted by DECEA engineers who run test scenarios against your deployed system. DECEA may also use **internal automated testing tools** (`uss_qualifier` or equivalent) to exercise specific scenarios programmatically.

8.2 Prerequisites for Homologation

Before requesting homologation, your USS must:

1. **Implement all mandatory production endpoints** (see Chapter 6):
   - `GET /uss/v1/operational_intents/{entityid}`
   - `POST /uss/v1/operational_intents`
   - `POST /uss/v1/constraints`
   - `GET /uss/flights`
   - `GET /uss/flights/{id}/details`
   - `GET /version`
   - `GET /diagnostics/time`
   - `POST /telemetry`
2. **Implement all testing endpoints**:
   - All `flights.yaml` endpoints
   - All `injection.yaml` endpoints
   - The `versioning.yaml` endpoint
3. **Be deployed and reachable from the internet** with a valid HTTPS base URL.

4. **Be registered in the Sandbox** with a development API Key and able to interact with the Sandbox DSS.
5. **Satisfy all non-functional requirements** (see Chapter 7), especially NTP synchronization.

8.3 Requesting Homologation

To initiate the homologation process:

1. Contact DECEA through the official support channels:
   - **Mattermost** (developer channel)
   - **Central de Ajuda** (ticketing system)
2. Provide:
   - Your company name and CNPJ.
   - The version of your software being submitted.
   - The publicly accessible base URL of your USS.
   - Contact details for the technical team.
3. DECEA will schedule the homologation session and provide:
   - A testing API Key (granting access to the test ecosystem).
   - The URL of the test ecosystem (if separate from the main Sandbox).
   - The test scenario list to prepare for.

8.4 What DECEA Tests

DECEA validates your USS against 18 defined scenarios. Each scenario tests one or more functional and non-functional requirements.

Scenario Matrix

#	Scenario	Key Requirements Tested
1	Register an operation in an empty area	Auth (8.1), NTP, notifications (8.2), OIR creation (8.3), conflict check (8.4), DSS discoverability
2	Register an operation near a non-intersecting Constraint	+ Constraint query and processing (8.7)

#	Scenario	Key Requirements Tested
3	Register an operation near an intersecting Constraint	+ 4D intersection detection, deconfliction (8.5), operator notification
4	Register an operation near a non-intersecting OIR	+ Peer-to-peer OIR details fetch
5	Register an operation near an intersecting OIR	+ Deconfliction, operator notification
6	Delete an operation (OIR)	+ Deletion notification to subscribers
7	Activate an operation conflicting with another active OIR at the same priority	+ Pre-activation conflict check (8.6), blocking activation
8	Activate an operation conflicting with an active OIR at higher priority	+ Deconfliction before activation
9	Activate an operation conflicting with an active OIR at lower priority	+ Correctly allows activation when lower priority conflict exists
10	DECEA creates a Constraint over an existing Accepted OIR	+ Reaction to constraint notification (8.11), deconfliction or state change
11	DECEA creates a Constraint over an existing Activated OIR	+ Immediate reaction (Nonconforming or deconfliction)
12	DECEA's USS activates a higher-priority OIR over an active OIR	+ Priority-based conflict resolution, deconfliction
13	USS creates an ISA when the OIR is activated	+ Remote ID ISA lifecycle (8.8)
14	USS shares drone position during an active operation	+ Telemetry serving at `/uss/flights` and `/uss/flights/{id}/details`
15	USS updates drone position every 10 seconds	+ Update frequency requirement
16	Drone position exits OIR volume → transition to Nonconforming	+ Conformance monitoring timing (≤10s detection, ≤5s DSS update)
17	Drone position exits OIR → Nonconforming → returns → back to Activated	+ Recovery from non-conformance
18	Drone position exits OIR → Nonconforming for 60s → Contingent	+ Full emergency state machine

Requirement Mapping

Code	Full Requirement
RF 1	4D geospatial intersection calculation, 1 cm precision

Code	Full Requirement
RF 2	DSS discoverability before state transitions
RF 3	Automatic high priority for critical situations
RF 4	Transition to Accepted only if no higher-priority conflict
RF 5	Pre-activation final conflict check
RF 6	Continuous situational awareness via subscriptions
RF 7	Automatic conflict notification to operators
RF 8	CMSA-only role for Nonconforming/Contingent transitions
RF 9	Constraint intersection analysis before OIR creation
RF 10	Continuous constraint monitoring during operations
RF 11	Automatic propagation of constraint notifications to affected operators
RF 12	Full telemetry recording during conformance monitoring
RF 13	Aircraft position updated and available within ≤10 seconds
RNF 1	NTP synchronization to `ntp.decea.gov.br`
RNF 2	Consistent timestamps from synchronized clock
RNF 3	Notification latency ≤5s in ≥95% of cases
RNF 4	ISO 8601 / RFC 3339 with UTC timezone
RNF 5	Audit logs with NTP-derived timestamps

8.5 How the Automated Testing Works

For scenarios involving Remote ID (scenarios 13–15), DECEA's test framework calls your `injection.yaml` endpoints to inject simulated telemetry:

1. DECEA's framework calls `PUT /tests/{test_id}` with a sequence of simulated aircraft positions.
2. Your USS processes these as if they were real drone telemetry and makes them available via `GET /uss/flights`.
3. DECEA's framework queries `GET /uss/flights` and `GET /uss/flights/{id}/details` to verify the data is correct and timely.
4. After the test, `DELETE /tests/{test_id}/{version}` clears the injected data.

For scenarios involving flight planning (scenarios 1–12), DECEA's framework uses your `flights.yaml` endpoints:

1. DECEA's framework calls `PUT /flight_plans/{id}` to simulate a user creating a flight plan.
2. Your USS translates this into an OIR creation in the DSS and notifies subscribers.
3. DECEA validates the DSS state and your subscriber notifications.
4. `POST /clear_area_requests` may be used to reset the test environment between scenarios.

For conformance monitoring scenarios (16–18), DECEA combines both `injection.yaml` (to inject positions outside the volume) and `flights.yaml` (to set up the OIR), then monitors whether your USS correctly transitions states and notifies within the required time limits.

8.6 Testing Environment Access

During homologation, DECEA provides a dedicated test environment:

- A test API Key with access to the test ecosystem.
- Possibly a separate test DSS instance.
- Access credentials for DECEA's monitoring systems (Interface UTM) to observe your operations during the test.

This testing key and environment are separate from your normal development (Sandbox) key.

8.7 After Successful Homologation

Upon passing all scenarios:

1. DECEA grants your software the **U1 permission level** for the **production environment**.
2. You receive a **production API Key** tied to your validated software version.
3. You can now:
   - Create UTM Zones in the production Portal UTM.
   - Begin operating flights in production.
   - Generate sub-keys for third-party operators using your software.

Third-Party Sub-Keys

If your USS software is a platform sold to third-party drone operators:

- Your company (as the validated software owner) creates **sub-API Keys** for each third-party client.
- The third-party client uses their sub-key to authenticate and to create their own UTM Zones.
- All operations under sub-keys are still associated with your validated software version.

8.8 Re-Homologation

If you release a **new major version** of your USS software with significant architectural changes, a new homologation process may be required. Contact DECEA to determine whether re-validation is needed for your specific changes.

Minor updates and bug fixes that do not affect the API contract or safety-critical behaviors typically do not require re-homologation.

8.9 Homologation Checklist

Use this checklist before requesting homologation:

APIs

- ☐ `GET /uss/v1/operational_intents/{entityid}` implemented and tested
- ☐ `POST /uss/v1/operational_intents` implemented (correctly handles creation, update, deletion)
- ☐ `GET /uss/v1/constraints/{entityid}` implemented
- ☐ `POST /uss/v1/constraints` implemented
- ☐ `GET /uss/v1/operational_intents/{entityid}/telemetry` implemented (active in Nonconforming/Contingent)
- ☐ `GET /uss/flights` implemented with correct data structure
- ☐ `GET /uss/flights/{id}/details` implemented
- ☐ `GET /version` implemented
- ☐ `GET /diagnostics/time` implemented with NTP status
- ☐ `POST /telemetry` implemented
- ☐ All `flights.yaml` endpoints implemented
- ☐ All `injection.yaml` endpoints implemented

- [ ] `versioning.yaml` endpoint implemented

Functional

- [ ] OIR creation → DSS → subscriber notification flow works end-to-end
- [ ] OIR state transitions (Accepted → Activated → Nonconforming → Contingent) work correctly
- [ ] Pre-activation conflict check is performed
- [ ] Constraint queries and processing are implemented
- [ ] ISA is created on activation and deleted on flight closure
- [ ] Incoming `POST /uss/v1/operational_intents` notifications trigger conflict re-evaluation
- [ ] OIR deletion triggers subscriber notification (with `operational_intent` omitted)
- [ ] Nonconforming detected within 10 seconds
- [ ] DSS updated within 5 seconds of Nonconforming detection
- [ ] Contingent triggered after 60 seconds of Nonconforming

Non-Functional

- [ ] NTP synchronized to `ntp.decea.gov.br` with ≤5s offset
- [ ] All timestamps in ISO 8601 / RFC 3339 / UTC
- [ ] Subscriber notifications sent within 5 seconds in ≥95% of cases
- [ ] Telemetry data updated at most every 10 seconds
- [ ] 4D intersection precision at 1 cm
- [ ] Audit logs in place with NTP-derived timestamps
- [ ] All inbound JWT tokens validated (signature, expiry, audience, scope)
- [ ] System deployed publicly on HTTPS

Revision #2
Created 12 July 2026 20:11:30 by João Pedro Favoretti
Updated 14 July 2026 14:52:34 by João Pedro Favoretti