Operational Workflows

Managed documentation for Operational Workflows.

Track-to-Task Workflow

Track-to-Task Workflow

Field	Value
Distribution	BSS — OpenSOF
Product	OpenSOF
Release	0.15.1
Deployment	`opensof.bss.dev`
Source	`/srv/bss/releases/OpenSOF-v0.15.0-20app`
Evidence	Reference cross-component acceptance test
Source fingerprint	`cda583b42f2a0d296b268ab5b174932a378c8be6fb52ffa1af14af896e990f6e`
Status	Generated baseline — human review required

> **Verification boundary:** This page combines platform design guidance with static evidence from the release. It does not prove that every detected interface is enabled, reachable, secure, or operational in the deployed environment.

Objective

Demonstrate the operational chain from track ingestion through visualization, semantic association, collaboration and an acknowledged/completed task.

Prerequisites

- ☐ OpenTrack, OpenCOP, OpenTask and required OpenBus/knowledge services are healthy.
- ☐ A controlled track generator or recorded input is available.

- [ ] Two test users/roles are available for assignment and acknowledgement.

Procedure

1. Create or ingest a uniquely identified test track with known position, time, type and source.
2. Verify OpenTrack receives/updates it and OpenCOP displays it in the correct location/type.
3. Open the track details and confirm source time, provenance and semantic type.
4. Create an OBSERVE, INVESTIGATE, TRACK or other allowed task from the track.
5. Assign it to the second user/asset; acknowledge, execute and complete it.
6. Verify task state is visible in OpenTask/OpenCOP and correlated to the original track.
7. Capture messages/events/logs proving the state transitions.

Pass criteria

Track position/type/time are correct; task creation and authorization succeed; all state transitions are visible and correlated; duplicate updates do not create duplicate tasks; evidence is retained.

Cleanup

Delete or mark the test track/task complete according to the test-data policy and verify no generator continues publishing.

Maintainer Notes

Add human-reviewed deployment notes, corrections, decisions, screenshots, and links here. Content outside the generated block is preserved on future runs.

OpenEyes Sensor Product Workflow

OpenEyes Sensor Product Workflow

Field	Value
Distribution	BSS — OpenSOF
Product	OpenSOF
Release	0.15.1
Deployment	`opensof.bss.dev`
Source	`/srv/bss/releases/OpenSOF-v0.15.0-20app`
Evidence	Reference operational demonstration
Source fingerprint	`cda583b42f2a0d296b268ab5b174932a378c8be6fb52ffa1af14af896e990f6e`
Status	Generated baseline — human review required

> **Verification boundary:** This page combines platform design guidance with static evidence from the release. It does not prove that every detected interface is enabled, reachable, secure, or operational in the deployed environment.

Objective

Demonstrate that imagery captured by OpenEyes or uploaded from a phone persists as a sensor product, appears in the OpenEyes product strip, carries source/time/context metadata, and can be associated or pushed to an OpenFiles workspace/release group.

Prerequisites

☐ OpenEyes asset/device session is healthy and authorized.

☐ Local sensor-product storage is writable and persistence survives page refresh/service restart.

☐ Phone upload script has endpoint/token configured outside the script or uses a safe prompt.

☐ OpenFiles target workspace/release group exists and permissions are verified.

Procedure

1. Capture one image from the FPV view and record the asset/session ID and capture time.
2. Verify a thumbnail appears in Sensor Products and opens the full image.
3. Refresh the page and restart only the OpenEyes service; verify the product remains.
4. Run the phone upload script with a uniquely named image and verify it appears in the same product strip.
5. Inspect metadata: product ID, device/uploader, source time, receipt time, MIME type, size, checksum and optional position/track/task.
6. Push the product to the selected OpenFiles workspace/release group and verify recipient access.
7. Verify audit/provenance links from OpenEyes product to the OpenFiles artifact.

Pass criteria

Both captured and uploaded images persist, render correctly, have accurate metadata/checksum, are access-controlled, transfer once without corruption, and retain the source-to-share provenance trail.

Negative tests

- Unauthorized upload or workspace push is rejected and audited.
- Unsupported/oversize/corrupt content is rejected safely.
- Duplicate retry does not create uncontrolled duplicate products.
- Metadata does not expose token, local file-system path or sensitive EXIF beyond policy.

Maintainer Notes

Add human-reviewed deployment notes, corrections, decisions, screenshots, and links here.
Content outside the generated block is preserved on future runs.

OpenRF Spectrum Workflow

OpenRF Spectrum Workflow

Field	Value
Distribution	BSS — OpenSOF
Product	OpenSOF
Release	0.15.1
Deployment	`opensof.bss.dev`
Source	`/srv/bss/releases/OpenSOF-v0.15.0-20app`
Evidence	Reference SDR/edge demonstration plus technology indicators
Source fingerprint	`cda583b42f2a0d296b268ab5b174932a378c8be6fb52ffa1af14af896e990f6e`
Status	Generated baseline — human review required

> **Verification boundary:** This page combines platform design guidance with static evidence from the release. It does not prove that every detected interface is enabled, reachable, secure, or operational in the deployed environment.

Objective

Demonstrate registered SDR discovery, synchronized/located collection, edge spectrum processing and a low-latency spectrum or detection product visible through OpenRF without transporting unnecessary raw samples.

Procedure

1. Register the radio agent/device with stable ID, driver, capabilities, location profile and time source.
2. Confirm agent health, radio enumeration and selected center frequency/sample/bandwidth/gain settings.
3. Start a bounded scan or known FM/spectrum example and observe local processing load/drop indicators.
4. Open the OpenRF view and verify live spectrum/detection product, labels, units and timestamps.
5. Change one authorized setting and verify acknowledgement plus visible result.
6. Record latency, update rate, sample/drop quality and time/PNT status.
7. Stop the task and verify radio resources are released.

Pass criteria

Correct radio/profile is used; settings are bounded and authorized; product units/time are correct; view remains responsive; loss/degradation is visible; stop releases hardware; no raw bandwidth flood occurs across the WAN.

Observed technology indicators

Indicator	Occurrences
PTP/PPS	1064
camera/video	362
UHD/USRP	324
AIS	154
FlightGear	121
DIS	85
PlutoSDR	56
HackRF	55
SoapySDR	54
ADS-B	46

Maintainer Notes

Add human-reviewed deployment notes, corrections, decisions, screenshots, and links here.
Content outside the generated block is preserved on future runs.

OpenLVC and FlightGear Workflow

OpenLVC and FlightGear Workflow

Field	Value
Distribution	BSS — OpenSOF
Product	OpenSOF
Release	0.15.1
Deployment	`opensof.bss.dev`
Source	`/srv/bss/releases/OpenSOF-v0.15.0-20app`
Evidence	Reference LVC integration test
Source fingerprint	`cda583b42f2a0d296b268ab5b174932a378c8be6fb52ffa1af14af896e990f6e`
Status	Generated baseline — human review required

> **Verification boundary:** This page combines platform design guidance with static evidence from the release. It does not prove that every detected interface is enabled, reachable, secure, or operational in the deployed environment.

Objective

Demonstrate a virtual FlightGear entity entering the same operational representation used by live tracks, then being observed, correlated, tasked and replayed without hiding its simulation provenance.

Procedure

1. Start a controlled FlightGear scenario with known aircraft, position, heading, speed and scenario time.
2. Start the OpenLVC bridge and verify its input/output endpoint and entity identifier.
3. Verify OpenTrack receives updates with correct coordinates, altitude, velocity, time and LVC provenance.
4. Display the entity in OpenCOP and compare against the FlightGear state.
5. Create an observation/follow/intercept task and verify state transitions without issuing unsafe real-world effects.
6. Record scenario/event stream for OpenAAR-MR and replay the segment.
7. Verify live and virtual entities remain distinguishable by source/provenance while sharing common semantics.

Pass criteria

Track error remains within test tolerance; update rate/time mapping is stable; provenance is explicit; task workflow completes; replay reproduces the operational sequence; no virtual input crosses a prohibited live-effect boundary.

Maintainer Notes

Add human-reviewed deployment notes, corrections, decisions, screenshots, and links here. Content outside the generated block is preserved on future runs.