DRACO7RT · TRAINING · 3 NOVEMBER 2026

Training: step by step

What each role is trying to get done, and exactly how: numbered steps, what you should see after each one, and what to do when you do not

Five walkthroughs: everyone runs the mission test first, then the flight software developer, ground system developer, operator or reviewer takes their own.

The working copy, with a box to tick per step and a Print button: https://draco.efsi.com/training/

FIVE WALKTHROUGHS

Everyone takes G first, then the one for their role

Walkthrough	What you get done	Steps, time
G Everyone	Run the full mission test from COSMOS, then from Yamcs; read the result	10 steps, 15 min
F Flight software developer	Change an app in VS Code, unit-test it, fly it, stop it at a breakpoint, put it back	12 steps, 30 min
D Ground system developer	Add a value to a COSMOS page, install a new plugin version, carry it to Yamcs	9 steps, 20 min
O Operator	Send every mission command by hand, then fly the stages from Yamcs stacks	10 steps, 30 + 30 min
R Reviewer (SMA)	Follow one requirement from its text to the test, the record and the raw log	8 steps, 30 min

Each walkthrough gives the goal, what to have before you start, numbered steps with what you should see after each, and a table for when it does not go that way.

Working copy with tick boxes and Print: https://draco.efsi.com/training/ (front door tile Training: step by step; Viewer › Mission › Training (step by step)).

G · EVERYONE

Run the full mission test, from both ground stations

What you are trying to get done

Fly the whole mission end to end, from the pad to the GPS spoof response, twice: once commanded by COSMOS and once by Yamcs. Each run checks the vehicle at nine points, P0 to P8, and writes its own evidence record.

When you finish you know what the system does, where to watch it, and how to tell that it worked.

Before you start

- A sign-in to draco.efsi.com
- Only a browser: the stack runs on the server
- Nobody else mid-test: the vehicle is shared

Time: about 15 minutes: two runs of about 3 minutes each, and two vehicle resets of about 70 seconds.

Walked 3 Nov 2026 from a browser: COSMOS 9/9 in 182 s, Yamcs 9/9 in 177 s.

G · THE STEPS

1. **Open the front door**, https://draco.efsi.com/, and sign in: status strip at the top, tiles, **Demo control** at the bottom
2. **Read the strip: Vehicle** FRESH · launch-ready, **CFE_ES age** 1–2 s, **Commanding** COSMOS, **42 view** running
3. **Demo control › STACK › Fresh vehicle, Run:** the job pane ends == OK in about 70 s; MODE SAFE, 12 kg, STOWED
4. **Open something to watch** in a second tab: tile **Mission Control** (Viewer › Home › Mission home); optional tile **Live 3D view**
5. **MISSION › Mission test (COSMOS), Run:** a banner per phase in the job pane, about 3 min, ends 9/9 PASS and cosmos_<UTC>
6. **Know what P0 to P8 proved:** refused FIRE, launch, deploy, detumble, autonomous dump, burn, IMU, GPS spoof
7. **Reset again:** after a flight the strip reads **FLOWN · this boot**; **Fresh vehicle**, **Run**, wait for == OK
8. **YAMCS › Mission test (Yamcs), Run:** same nine phases; it switches Yamcs commanding on and back off; writes ground_<UTC>
9. **See both side by side:** Full Mission Test deck, slide “stations”, read from the two records; or the logs in Script Runner and Yamcs Activities
10. **Leave it clean: Fresh vehicle** once more, for the next person: Vehicle FRESH · launch-ready, Commanding COSMOS

If not: grey buttons say why on hover (FLOWN: **Fresh vehicle**). P7/P8 SKIP: 42 was not running: 3D VIEW › **42 start**, then **Fresh vehicle**. P0 fails: the vehicle was not on the pad.

F · FLIGHT SOFTWARE DEVELOPER

Change, test, fly and debug one line of flight code

What you are trying to get done

Make one small, visible change to a flight app, prove it with the unit tests, put it on the vehicle, see it from the ground, stop the running vehicle at a breakpoint on that line, then put the code back.

The example: the text of the ACS NOOP event, `acs_app.c` lines 679 to 682.

Before you start

- VS Code with Remote-SSH and C/C++
- An SSH login in group dev_draco_efsi_com
- Tell the others: a rebuild takes the vehicle for 4 minutes, a breakpoint freezes it

Time: about 30 minutes, of which about 8 are two vehicle rebuilds.

Walked 3 Nov 2026: tests 100.00% of 525 lines in 21 s; rebuild 4 min; event seen; breakpoint at acs_app.c:681 hit; reverted.

F · STEPS 1 TO 6: FROM THE CODE TO THE VEHICLE

1. **Connect:** F1 › **Remote-SSH: Connect to Host...**; File › Open Folder /home/efsi/draco7_0 (never the /var/www path: Docker cannot see it). Explorer shows cFS, run, tools, web
2. **Find the code:** Ctrl+P acs_app.c, Ctrl+G 670, ACS_ProcessCmd; `case ACS_NOOP_CC:` at 679, the event text at 681 to 682
3. **Baseline tests:** Terminal › New Terminal, the commands below: RESULT: PASS, 100.00% of 525 lines, about 21 s
4. **Change line 682** to end “- XX” (your initials), save. Tests again: still PASS, since the test (acs_app_tests.c:154) checks the event is sent, not its words
5. **Ctrl+Shift+B** (task **vehicle: build + restart vehicle**): cross-build, sync /cf, new QEMU, OPERATIONAL, downlink, 42 back; about 4 min, ends “vehicle restarted”
6. **See it from the ground:** Viewer › GNC › ACS, or Viewer › Tools › Command Sender: send ACS NOOP; CMD_COUNT +1 and the Events dock along the bottom of the viewer shows your text

```
cd /home/efsi/draco7_0/cFS
bash apps/acs/unit-test/run_coverage.sh
bash apps/acs/unit-test/run_app_coverage.sh
```

If not: “docker: not found” on Ctrl+Shift+B: the terminal lacks /snap/bin; `export PATH=/snap/bin:$PATH` and press it again.

F · STEPS 7 TO 12: STOP THE VEHICLE ON YOUR LINE, THEN PUT IT BACK

7 **Breakpoint:** click in the margin left of line 681 (red dot); Run and Debug (Ctrl+Shift+D) › **Vehicle: attach to QEMU (PowerPC/RTEMS, GDB server)** › F5

8 **Hit it:** send ACS NOOP again from COSMOS; VS Code stops on 681 (yellow). Call Stack: ACS_ProcessCmd, ACS_AppMain, CFE_ES_TaskEntryPoint

9 **Let it go:** F5 (Continue), click the red dot to remove it, Shift+F5 (Disconnect); the event arrives in the Events dock

10 **Optional, a terminal:** tools/dev/gdb-vehicle.sh, then break, continue, bt, print, detach

11 **Put it back:** undo your one line, tests, Ctrl+Shift+B; NOOP shows the original text. Not “Discard Changes”

12 **For real:** a branch, unit tests, constants in a table, a closed evidence run, /review, a pull request

While stopped, the whole vehicle is stopped: telemetry freezes, the ground sees the age climb. Keep it under a minute or two, or IMU STALE can trip.

Hover A->CmdCount at the stop: it already counts this NOOP, because line 680 ran; the event on 681 has not been sent yet.

D · GROUND SYSTEM DEVELOPER

Add a value to a page, on both ground stations

What you are trying to get done

Add one telemetry value to a COSMOS page, install the change the supported way (a new plugin version), prove it is live, and carry the same change to the Yamcs station.

The example: ACS RATE_SOURCE_CMD, the commanded rate-source policy, in the ACS page’s Mode & Rates box.

Before you start

- VS Code on /home/efsi/draco7_0 (F, step 1)
- A sign-in to draco.efsi.com
- Tell the others: an install restarts the COSMOS interface for about a minute

Time: about 20 minutes.

Walked 3 Nov with RATE_SOURCE, now kept on the page (1.2.2 to 1.2.3, 58 s). Installed since 10 Nov: 1.2.4 (1.2.3 to 1.2.4 took 66 s); the exercise now raises it to 1.2.5.

D · THE STEPS

1. **Open** in VS Code (Ctrl+P) cosmos/openc3-fogi-open-source/targets/ACS/screens/acs.txt: box “Mode & Rates”
2. **Find the name** in targets/ACS/cmd_tlm/tlm.txt (RATE_SOURCE_CMD 8 UINT); only names there, audited against the binary
3. **Add the line** under RATE_SOURCE (below)
4. **Raise the version** (installed one: COSMOS Admin Console › Plugins) in build.sh line 6 and bootstrap-draco.sh line 32 (1.2.4 to 1.2.5)
5. **Install as an upgrade,** then --per-boot (below): about a minute
6. **Check COSMOS:** Viewer › GNC › ACS, box Mode & Rates: RATE_SOURCE_CMD reads AUTO in green
7. **Carry it to Yamcs:** cosmos2opi.py, load_operator.py, selfcheck.py; Yamcs › Telemetry › Displays › DRACO › acs.opi shows the row
8. **Never Save in the screen editor:** its copy hides every later plugin version of the page
9. **Keep or put back:** review and PR, or remove it with a further version (versions only go up)

```
LABELVALUE ACS HK_TLM_PKT RATE_SOURCE_CMD
run/upgrade-plugin.sh openc3-fogi-open-source 1.2.5
cd run && ./bootstrap-draco.sh --per-boot
```

O · OPERATOR

Fly the mission by hand, from COSMOS and from Yamcs

What you are trying to get done

Send each command of the mission yourself and watch the vehicle answer: launch, the autonomous deployments and momentum dump, a burn, and the two fault responses (a forged GPS fix and a bad gyro) with the recovery for each.

Then fly the same stages from the second station, Yamcs, with its command stacks.

Before you start

- Walkthrough G done once
- 42 running, for the GPS and IMU stages
- Nobody else commanding

Time: about 30 minutes for COSMOS and 30 for Yamcs.

The commands and answers are the mission test’s (9/9 from both stations, 3 Nov); stacks flown by hand 25 Oct; a NOOP via Yamcs 3 Nov.

0 · SEND, THEN WATCH FOR

Stage	Send (Viewer › Tools › Command Sender)	Watch for
Pad check	ACS NOOP	SAFE, 12 kg, STOWED; CMD_COUNT +1
Refusal	ACS FIRE_DEPLOY with CHANNEL 0	ERR_COUNT +1, still STOWED (no ARM)
Launch	SC START_RTS with RTS_ID 5	DETUMBLE, both DEPLOYED, then POINT
Autonomy	nothing	H_MAG past 4.0, DELTA_H, DUMP_COUNT +1
Burn	ACS STATIONKEEP with TARGET_DV_CMS 50	STATIONKEEP, DV_MS 0.5, back to POINT
GPS spoof	NAV GPS_FIX (defaults); then NAV CLEAR_REJECT, LC SET_AP_STATE AP_ID 1 NEW_STATE 1	REJECT_LATCHED SET, RTS-006 runs; after recovery CLEAR
Bad gyro	IMU SAMPLE with GYRO_X 5.0; then IMU CLEAR_FAULT, LC SET_AP_STATE AP_ID 2 NEW_STATE 1, ACS SET_RATE_SOURCE SOURCE AUTO	FAULT_LATCHED SET, RTS-007, RATE_SOURCE INTERNAL; after recovery IMU

Then Yamcs: Fresh vehicle, then front door › Demo control › COMMANDING STATION › **Switch to Yamcs** (strip Commanding reads YAMCS). In Yamcs (Viewer › Yamcs): Telemetry › **Parameter lists** DRACO 0 Pad check to DRACO 8 IMU; **Stacks**, folder DRACO: 0-check, 1-launch, 5-stationkeep, 7-gps-spoof, 8-imu-fault, one step at a time. At the end **Switch to COSMOS**, then **Fresh vehicle**.

R · REVIEWER (SMA, ASSURANCE)

Follow one requirement to its evidence

What you are trying to get done

Follow one requirement from its text to the test that verified it, to the run record and its raw log, and see how a failed or deviated check is shown. You change nothing on the system.

The example: dRT-ACS-008, the autonomous momentum dump.

Where things stand

- 83 requirements
- 79 VERIFIED-RUN
- 3 PENDING (unit-test evidence)
- 1 RUN-FAIL, an accepted deviation

Time: about 30 minutes. A sign-in is enough; repository access lets you open the files directly.

Counts read from cFS/evidence/rtems_reqs/requirements.csv on 3 Nov 2026.

R · THE STEPS

1. **Verification report:** front door › tile **Evidence** › **Design Verification Report (HTML)** or (PDF)
2. **Trace and compliance site:** same tile, **Requirements trace and compliance site**: each requirement to its source, verification and evidence
3. **Pick dRT-ACS-008** in requirements.csv: shall text, method, evidence, VERIFIED-RUN
4. **Open the record:** runs/ground_<UTC>/run.json, check V-YAMCS-P4P5 covers it, PASS, raw log beside it
5. **How verdicts are applied:** apply_verdicts.py reads the newest records and writes VERIFIED-RUN or RUN-FAIL
6. **The one failure:** verify/config.yaml deviations: V-RMT-SERSHELL, who accepted it, when, why
7. **Classification:** CLASSIFICATION.md, class E, not safety-critical, PROPOSED until SMA concurs
8. **Optional:** watch a run make a new record (walkthrough G)

The rule behind it: verdicts are read from logs by scripts, never typed; a failed check is accepted only as a recorded deviation, never silenced any other way.

WHAT WALKING THE STEPS FOUND

Two steps did not work until we walked them, so both were fixed

A breakpoint never stopped the vehicle

The cross debugger loads the toolchain’s “RTEMS GDB Support” Python scripts, and with them loaded the vehicle ran straight past every breakpoint. The distro debugger stopped at once.

Fix: `set auto-load python-scripts off` in the VS Code launch configuration and gdb-vehicle.sh. Then it stopped at acs_app.c:681 from both.

Every Yamcs display was blank

Yamcs draws a display inside a sandboxed frame. The browser sends that frame no sign-in cookie, so the gate turned away its renderer script and the page stayed empty.

Fix: that one file, Yamcs’s open-source renderer with no mission data in it, is served without the gate. The data still reaches the frame only from the signed-in page.

Both found and fixed on 3 November 2026 while walking walkthroughs F and D on the live system.

WHERE TO DO IT

The working copy is a page, with a box to tick per step

Training pages

https://draco.efsi.com/training/: every step in full, commands with a Copy button, the troubleshooting tables, a **Print** button at the top right. Ticks stay in your browser.

Tutorials

https://draco.efsi.com/tutorials/: seven hands-on lessons (front door › tile Tutorials) whose **Check** buttons read the live telemetry to tell you it worked.

Decks

https://draco.efsi.com/slides/: this one and the thirteen others (the blue **Presentations** button, top right of the front door; no sign-in). Each walkthrough's last line names the deck that goes deeper.

The vehicle is shared. Before F or D, tell the others; after O, press Fresh vehicle; R changes nothing.

Front door › tile Training: step by step, and Viewer › Mission › Training (step by step), open the pages.