DRACO7RT · TUTORIAL · UPDATED 10 NOVEMBER 2026

How to use DRACO7RT, by role

What a presenter, an operator, a flight-software developer, a ground-page developer, a reviewer and a Claude Code user each do, and where they go to do it

DRACO7RT is a prototype spacecraft simulator: real NASA flight software (cFS on RTEMS, PowerPC) running on an emulated board, commanded from a real ground system (COSMOS), with NASA 42 supplying the physics.

Everything starts at the front door, https://draco.efsi.com/. The slides, including this one, are at https://draco.efsi.com/slides/ (the blue **Presentations** button, top right of the front door) with no sign-in.

START HERE

Which one are you?

Presenter

You show the demo to an audience: fresh vehicle, launch, the autonomy firing, the GPS spoof, the IMU fault. Browser only, no terminal.

Slide 7

Operator

You fly the vehicle from COSMOS: watch telemetry, send commands, recover from faults, run the scripted mission test.

Slide 8, then the COSMOS Operator Guide

Flight-software developer

You change the cFS apps or tables, rebuild, restart the vehicle, and debug it with breakpoints from VS Code.

Slide 9

Ground-page developer

You build telemetry screens, walls and command buttons in COSMOS or Grafana, and keep packet definitions matching the flight code.

Slide 10

Reviewer (QA, SMA)

You judge the evidence: requirements, tests, run records, static analysis, the NASA standards mapping, and what still needs your concurrence.

Slide 11, then Latest for SMA

Claude Code user

You direct the AI assistant on the server to make changes, debug, or build pages, under rules a person reviews.

Slide 12

Most people are two of these at once. Slides 3 and 4 list the addresses and which ones need a sign-in; slides 5 and 6 the decks and the tutorials.

ADDRESSES AND ACCESS (1 OF 2)

Where everything is, and what needs a sign-in

Address, and how to get there	What it is	Access
https://draco.efsi.com/ (the front door)	Top to bottom: header (blue **Presentations**, **Sessions**, **Sign out**), live status strip, **Ground stations** panel, the tiles, **Demo control** buttons at the bottom	**Sign in with Microsoft**, or the shared COSMOS password in the box under it (**Sign in with password**), once
https://draco.efsi.com/slides/ (blue **Presentations** button, top right)	The fourteen decks in the browser, each with a PDF (slide 5)	Open, no sign-in
/viewer/ (tile **Mission Control**, or Ground stations › **Open COSMOS**)	The COSMOS pages with group buttons across the top: Home, Wall, GNC, Mission, cFS, Tools (Command Sender, Script Runner, Packet Viewer, Cmd/Tlm Server with its Log Messages) and more	Front-door session; the COSMOS tools open on it, no second login
/viewer/#wall/wall (tile **Telemetry Wall**)	COSMOS /tools/tlmviewer?config=wall: GNC, sensors and mission panels side by side, for a projector	Front-door session
/grafana/d/draco-wall (tile **Grafana Wall**) and /42/ (tile **Live 3D view**)	Grafana wall with 15 minutes of history; NASA 42 live 3D views of the vehicle	Front-door session
/yamcs/, /openmct/, https://plan.draco.efsi.com (Ground stations › **Open Yamcs** / **Open Plan**; Viewer › Yamcs › Open MCT)	Second ground station Yamcs, Open MCT displays on it, PlanDev mission planning: the Ground System deck	Front-door session; PlanDev then takes any name

No Microsoft account? The COSMOS password is the project’s shared operator password, not a personal account; ask the project owner. Nothing on the vehicle or in the ground system is reachable from outside except through these pages. Slide 4: the tools for learning, building and reviewing.

ADDRESSES AND ACCESS (2 OF 2)

Learning, building and reviewing: same sign-in

Address, and how to get there	What it is	Access
/tutorials/ (tile **Tutorials**; Viewer › Mission › Tutorials)	Seven hands-on tutorials; the live ones check the telemetry for you (slide 6)	Front-door session
/training/ (tile **Training: step by step**; Viewer › Mission › Training (step by step))	Five walkthroughs, one per role, a box to tick per step and a **Print** button	Front-door session
/tables/ (tile **Table Workbench**; Viewer › Mission › Table Workbench)	Build a stored command sequence or edit an app table, check it, deliver it to the vehicle	Front-door session
/lab/ (tile **Notebooks (Python)**; Viewer › Tools › Notebooks (Python))	JupyterLab over both archives, read only: nothing there can command the vehicle	Front-door session
/evidence/DRACO7RT_RTEMS_DVR.html and /compliance/ (tile **Evidence**)	Design Verification Report (HTML and PDF) and the requirements trace and compliance site, generated from the evidence	Front-door session
SSH to the server, folder /home/efsi/draco7_0	The repository, for developers (VS Code Remote-SSH)	Server account

Every one of these is behind the same front-door session as slide 3. Only /slides/ is open to anyone.

THE PRESENTATIONS

Fourteen decks at https://draco.efsi.com/slides/

Open, no sign-in: send this link to anyone first. Click a title to open that deck; in a deck use the arrow keys or the ‹ › buttons, n for notes, f for full screen.

Start and operate

How to Use It, by Role ↗
This deck: who does what, and where

Training: Step by Step ↗
Five walkthroughs; working copy at /training/

COSMOS Operator Guide ↗
Which screen to watch at every stage; troubleshooting

Yamcs Operator Guide ↗
Flying from the second station: lists, stacks, displays

Full Mission Test ↗
The scripted end-to-end test, flown from COSMOS and from Yamcs

Ground system

Ground System ↗
Yamcs, Open MCT and PlanDev beside COSMOS

CFDP File Transfer ↗
Files up and down, Class 1 and Class 2

Archive, Events and Notebooks ↗
Where the data goes; Python over it

Changes to the Baseline Delivery ↗
What the ground gained since 22 July

Flight software

Writing and Debugging Code ↗
An app's shape, the build loop, GDB on the vehicle

Tables and Stored Commanding ↗
Constants in tables, the RTS builder, delivery

Assurance and security

Latest for SMA ↗
NPR 7150.2D, the SWE-134 scorecard, AI lessons

Flight Software Security ↗
Attack surface, SpaceCOP, the 14 scenarios

Ground Security and Zero Trust ↗
The ground scored pillar by pillar

The same list is the front door’s **Presentations** tile (each title · PDF · live copy). Each deck also lives as a claude.ai artifact at a fixed address, for the people who edit it.

LEARN BY DOING

Seven tutorials at https://draco.efsi.com/tutorials/

Each uses a different way of learning. The live ones fly the real vehicle: you command in COSMOS beside the steps, and a **Check** button reads the telemetry to tell you it worked.

1 First flight

GUIDED WALKTHROUGH

First command, one refused, launch, deployments, detumble

2 Scavenger hunt

DISCOVERY

Eight values to find with no map; answers checked live

3 Predict, observe, explain

COMMIT, THEN WATCH

Wheel momentum, the autonomous dump, what a burn costs

4 Fault case: something is wrong

PROBLEM-BASED

An unannounced IMU fault: diagnose it, recover in order

5 Put it in order

SEQUENCING PUZZLES

Five chains: telemetry path, launch, dump, recovery, table load

6 Recall check

RETRIEVAL PRACTICE

Fifteen questions; a miss comes back until you have it

7 Notebook: worked example to your own

FADED WORKED EXAMPLES

Python: one worked, one with blanks, one your own

Suggested path

Fresh vehicle, then 1 → 3 → 4 → 2 → 7. Do 5 and 6 any time, and 6 again a day later.

Front-door session needed: tile Tutorials, or Viewer › Mission › Tutorials. Tutorial 1 needs a fresh vehicle; progress is kept in your browser.

PRESENTER

The ten-minute demo, from a browser

Before the audience

1. Sign in at https://draco.efsi.com/ (the front door)
2. At the bottom, Demo control › STACK › **Fresh vehicle**, then **Run** in the confirm box. Wait until the strip’s Vehicle cell (top left) reads **FRESH · launch-ready**, about 70 to 90 s
3. On the projector: tile **Telemetry Wall**; press F11
4. Optional second window: tile **Live 3D view** (/42/). Its strip cell **42 view** must read running, else Demo control › 3D VIEW › **42 start**

The show

1. Front door › Demo control › MISSION › **LAUNCH (RTS-005)**, then **Run**: separation, deployments, detumble to pointing, all onboard
2. Wait for the momentum sawtooth: at 4.0 the vehicle dumps momentum by itself. That is the autonomy
3. Or MISSION › **Mission test (COSMOS)** instead of LAUNCH: the whole scripted sequence, a pass or fail at the end, about three minutes; its log streams in the **Job output** pane under the buttons
4. GPS spoof and IMU fault: the COSMOS Operator Guide, stages 7 and 8

What to say

- Everything on the software side is real: NASA flight software, the real-time OS, the processor model, the ground system
- Only the board, the physics and the sensors are simulated. Two ground stations watch it, and a NASA planner’s prediction matched the flight (Ground System deck)
- No AI runs on the vehicle. An AI assistant helped build and test it, under rules a person reviews
- If a value looks frozen, say so and press **Fresh vehicle** again. The audience prefers honesty to a stuck demo

LAUNCH and Mission test refuse to run unless the vehicle is fresh: their buttons are greyed and say why on hover. LAUNCH’s confirm box has a tick box **Force even though the vehicle is not fresh**: leave it unticked. Deeper: the COSMOS Operator Guide, stage by stage.

OPERATOR

Flying the vehicle from COSMOS

The vocabulary

- **Target**: one thing you talk to: ACS, NAV, IMU, or FOGI for the whole vehicle
- **Packet** and **item**: ACS HK_TLM_PKT H_MAG is target, packet, item
- **Screen**: a Telemetry Viewer page. FOGI HOME (Viewer › Home › Mission home) is the hub; most FOGI pages have a **HOME** button
- **Command**: Viewer › Tools › Command Sender: **Select Target**, **Select Packet** (the command), fill the **Value or State** column, then **Send** at the top right

The mission, by screen

- Pre-launch: Home › Mission home, everything live, MODE SAFE, FUEL 12.0
- Launch and deploy: Mission › Events narrates, Mission › Deployments ladders step
- Pointing and the dump: GNC › Attitude & rates graph; Mission › Limits & autonomy shows the limit checker watching
- Burn and fuel: Mission › Propulsion buttons (**Station-Keep 0.5 m/s**, **Refuel (12 kg)**)
- GPS and IMU: GNC › NAV (GPS gate) and GNC › IMU: gates, latches, recovery buttons

Three rules of thumb

- Trust a value only if its received time is now: a first query after a restart can show the previous boot
- A rejected GPS fix or IMU sample is the gate doing its job; after a 42 restart it rejects on purpose until you reset it
- After an autonomous response, recover in order: clear the latch, wait for it to read clear, re-arm the actionpoint, return the source to AUTO

The COSMOS Operator Guide walks every stage with the values to expect. COSMOS is the commanding station; Yamcs sees the same telemetry and commands only when switched on at front door › Demo control › COMMANDING STATION › **Switch to Yamcs** (Ground System deck). Step by step: Training walkthrough O.

FLIGHT-SOFTWARE DEVELOPER

Edit, rebuild, restart, debug: the real image from VS Code

Set up once

- VS Code with the free Remote-SSH and C/C++ extensions
- F1 › **Remote-SSH: Connect to Host...**, then File › Open Folder: /home/efsi/draco7_0 (never the /var/www path)
- The repo's .vscode folder supplies the tasks, debug configurations and include paths
- Where the code is: cFS/apps/acs, nav, imu; tables and mission config in cFS/draco_rtems_defs

The loop

- Edit. Run the app's unit tests on the host first: unit-test/run_coverage.sh and run_app_coverage.sh, seconds each, 100% line coverage expected
- **Ctrl+Shift+B** (task **vehicle: build + restart vehicle**): cross-build, sync, reboot the QEMU vehicle, re-map COSMOS, 42 back up. About four minutes
- Run and Debug (Ctrl+Shift+D) › **Vehicle: attach to QEMU (PowerPC/RTEMS, GDB server)** › F5: breakpoints, stepping, variables, call stack in the real image
- Tag the lines you add with their requirement id, as the surrounding code does

Know before you debug

- GDB sees one thread, the CPU. RTEMS tasks are not listed; the stack is whatever task was running
- A breakpoint stops the whole vehicle; COSMOS goes stale until you continue. Keep stops short
- Per-task breakpoints and hot reload: **Native: launch core-cpu1 (all apps as threads)** runs the same mission as a Linux process
- Before committing: gen_trace.py reports 0 untagged, verify/dvr_gate.py passes

Step by step: Training walkthrough F (https://draco.efsi.com/training/#fsw). Full notes in tools/dev/README.md. Apps are linked into the image, so new code always goes through rebuild and restart.

GROUND-PAGE DEVELOPER

Telemetry pages, walls and buttons that match the flight code

Where things live

- Plugin: cosmos/openc3-fogi-open-source. Targets ACS, NAV, IMU, FOGI, plus the inject-only GPS42, IMU42 and SPACECOP; the base plugin owns the CFS_INT interface and the cFE packets
- Screens: targets/FOGI/screens/*.txt (composite pages, walls) and targets/<app>/screens (per app)
- Packet and command definitions: targets/<T>/cmd_tlm/*.txt, written from the flight structs
- Grafana walls: run/grafana/dashboards, generated by run/grafana/tools/gen_dashboards.py

Making a page

1. Confirm each item exists with a telemetry query before drawing it
2. Copy the style of wall_sensors.txt: VERTICAL wrapper, MATRIXBYCOLUMNS grids, LED for states, FORMATVALUE for floats, SPARKLINE for trends, a HOME button
3. Bump VERSION in the plugin's build.sh and FOGI_VER in run/bootstrap-draco.sh (same value), then run/upgrade-plugin.sh <plugin dir> <version> and ./bootstrap-draco.sh --per-boot: it installs as an upgrade and re-maps ACS, NAV and IMU
4. Open the page in the viewer and check every value is live before calling it done

Traps that cost us a day each

- A comment line ending in + or \ swallows the next line of a plugin or target file
- A plain install of a new version, or a reinstall too soon after removal, breaks the targets; upgrade-plugin.sh avoids both
- Never map GPS42 or IMU42 to the interface: they are ground-side inject targets
- "received with actual packet length" in Log Messages means a definition drifted from the flight struct: fix the definition
- Never **Save** from the Telemetry Viewer's screen editor: its copy hides every later plugin version

Changed a packet definition? Yamcs reads the same list: run python3 tools/ground/cosmos2xtce.py, then restart Yamcs (--check fails if you forget). Changed a screen? cosmos2opi.py and load_operator.py carry it to the Yamcs displays. Do not reinstall during a demo.

REVIEWER: QA AND SMA

Where the evidence is, and what needs your signature

You want to see	Where it is	State today
Software class and safety determination	cFS/evidence/rtems_reqs/CLASSIFICATION.md: Class E, not safety-critical, mapping, AI-code approach, simulation credibility	Proposed; needs SMA concurrence
Requirements and trace	requirements.csv and TRACE.md: 83 requirements, all 83 traced to code; verdicts from the newest run records	79 VERIFIED-RUN, 3 PENDING, 1 RUN-FAIL
Test results with logs	cFS/evidence/runs/run_20261023_201111/run.json: 66 checks with requirement, verdict and quoted console lines; mission-test records cosmos_<UTC> and ground_<UTC>; unit-test logs in cFS/apps/*/unit-test	64 pass, 1 skip, 1 dispositioned deviation
Static analysis	cFS/evidence/static: cppcheck findings, deviation log, complexity per function	All findings dispositioned
The report and the site	Front door › tile **Evidence** › **Design Verification Report (HTML)** or **(PDF)**, every number citing a hashed artifact; **Requirements trace and compliance site** (/compliance/) from the same data	Rebuilt 10 Nov; regenerate after each run: gen_report.py, gen_site.py
Deviations	verify/config.yaml, deviations: who, when, why; the gate warns instead of failing	One: the serial-shell check
Standards mapping and AI rules	Latest for SMA deck: scorecard across NPR 7150.2D, NASA-STD-8739.8B, 7009B, 7123.1D; NASA's AI guidance quoted	Self-assessment; plans missing

The one thing only you can do: concur with or correct the Class E determination and the not-safety-critical finding, and accept or reject the deviation. Everything else is readable without the project. Step by step: Training walkthrough R.

CLAUDE CODE USER

Directing the AI assistant, inside the rules

What it is

- A program on the server that reads the repository, runs commands and tests, writes code and notes, and reports back in a text conversation
- Start it with the claude command in the VS Code terminal (Terminal › New Terminal) or any SSH session, from /home/efsi/draco7_0
- It wrote most of the new code, the tests, the evidence tooling and these decks, under a person’s direction

Three commands

- **/fsw-change** what you want changed: it reads the requirements, edits, tests, rebuilds, restarts, verifies in COSMOS, and shows the diff
- **/fsw-debug** the question: it attaches GDB to the vehicle, sets the breakpoint or watch, reports what it saw versus what it infers, and resumes
- **/cosmos-screen** the page: it writes it from the flight structs, reinstalls the plugin, and checks every value live

The rules it works under

- A pass needs positive evidence the path ran, never the absence of an error
- It never bypasses a gate to make a demo green; if a gate blocks it, it stops and says so
- Workarounds and feature drops are written down as decisions for you
- You review every change before it is committed. NASA treats AI-written code as auto-generated code: same tests, same review, your responsibility

The rules live in the skills themselves (.claude/skills) and in CLASSIFICATION.md section 5. The Latest for SMA deck shows where the assistant’s judgment went wrong, what caught it, and why these rules exist.

WHEN IT DOES NOT LOOK RIGHT

First moves, and where the written record is

Everyone

- Every value frozen or empty: front door › Demo control › HEALTH › **Verify**, then **Downlink / map**; if still dead, STACK › **Fresh vehicle**
- Cannot sign in: the slides need no sign-in; sign in from the front door (https://draco.efsi.com/), not from a deep link
- A page says a packet does not exist: a plugin is reinstalling; wait a minute and hard-refresh
- GPS or IMU rejecting everything after a 42 restart: expected; the COSMOS Operator Guide (stage 7) has the reset

Developers

- Build or stack command fails strangely: you ran it from /var/www; use /home/efsi/draco7_0
- New code not on the vehicle: the sync step was skipped; Ctrl+Shift+B does build, sync and restart together
- GDB says the architecture is not recognised: use the provided launch configurations, which set it before loading
- Unit tests fail after a 42 or vehicle change: they do not touch the vehicle; the failure is in the code

The record

- CLAUDE.md: the conventions and the traps, kept current
- claude-notes.md: the engineering record; read the NOW section first
- claude-notes/ONBOARDING.md: the full ramp-up for a new person
- The fourteen decks at /slides/ (slide 5), the seven tutorials at /tutorials/ (slide 6) and the five walkthroughs at /training/

The troubleshooting table in the COSMOS Operator Guide covers the symptoms an operator sees; the developer README covers the build and debug loop.