CFDP File Transfer

What the CCSDS File Delivery Protocol is, how it runs between the vehicle and the ground, and how to move a file from Yamcs, from COSMOS or with one button

The vehicle runs NASA’s cFS CF application; the ground runs the open-source Yamcs CFDP engine, shared by both ground stations. Class 1 and Class 2, uplink and downlink, every case checked byte for byte.

Slides 2 to 5 explain the protocol and the wiring. Slides 6 to 13 are the work instructions: before you start, uplink and downlink from Yamcs, the COSMOS page and script, the front-door test, the poll directory, and the loss demonstration. Slides 14 to 19: telemetry, troubleshooting (two slides), what was proved, limits, engineering notes. Words in bold are buttons or page names exactly as they appear on screen.

THE PROTOCOL

CFDP moves whole files over a space link, not packets

What it is

- CCSDS 727.0-B-5, the standard file delivery protocol used by NASA, ESA and JAXA missions
- A file is cut into **PDUs** (protocol data units): Metadata, File Data segments, EOF, then for Class 2 NAK, ACK and Finished
- Each side is an **entity** with a number; a transfer is a **transaction** named by the sender's entity and a sequence number
- A modular checksum travels in the EOF and is checked on arrival

What it is for on DRACO7RT

- **Uplink**: a new table, a stored-command sequence or any file to the vehicle's RAM disk
- **Downlink**: recorded data, logs or the demo files from the vehicle's file system to a ground bucket
- **Unattended**: anything dropped in the vehicle's poll directory is sent on its own
- The same PDUs ride the ordinary command and telemetry paths: no new link, no new radio

Who does the work

- On the vehicle: NASA's **CF** app (cFS CFDP, version 3.0.99), entity **25**
- On the ground: the **Yamcs CFDP service**, entity **21**, one engine for both ground stations
- COSMOS commands the vehicle (CF TX_FILE) and drives the Yamcs engine through its REST API
- You never touch a PDU: you name a file and a class

Vocabulary used in this deck: **Class 1** = unacknowledged, **Class 2** = acknowledged (reliable); **bucket** = a Yamcs file store; **tc-out** = the Yamcs command link to the vehicle.

CLASSES

Class 1 streams and hopes; Class 2 repairs until the file is whole

	Class 1, unacknowledged	Class 2, acknowledged (reliable)
PDUs	Metadata, File Data segments, EOF	The same, plus NAK (gaps), ACK, Finished
Lost segment	Stays lost; the receiver keeps a file with a hole	The receiver sends a NAK listing the gaps; the sender resends them
Feedback to the sender	None (optional “closure” Finished)	ACK of EOF and a Finished PDU: the sender knows the file arrived intact
Cost	One pass, no return traffic: fastest, needs no uplink for a downlink	Return traffic: a downlink needs the uplink too, an uplink needs the downlink
Use it for	Telemetry dumps, logs, anything you can ask for again	Tables, sequences, software: anything that must be exact
Measured 27 October	64 KiB with 20 % PDU loss: incomplete, as designed	64 KiB with 20 % PDU loss: complete and byte-exact after NAK rounds

Classes 3 and 4 of the older standard (store-and-forward through waypoints) were withdrawn in CCSDS 727.0-B-5; neither NASA CF nor Yamcs implements them. Class 1 and Class 2, both directions, is the whole protocol as flown today.

THE WIRING

PDUs ride the normal command and telemetry paths

Ground: Yamcs CFDP service

Entity 21. Files live in buckets: **cfdpUp** (to send), **cfdpDown** (received). Pages: File transfer, Buckets.

COSMOS reaches it over REST (Script Runner, the front door).

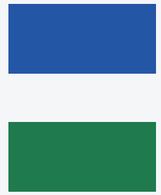


Uplink path (blue)

PDU wrapped in a cFE command, MID 0x18C8, on the Yamcs **tc-out** link (UDP to CI_LAB 10.5.5.38:1333) into the software bus, CF channel 0.

Downlink path (green)

CF emits PDUs as telemetry, MID 0x08C2; TO_LAB sends them to the downlink tee, which copies every packet to COSMOS and to Yamcs.



Vehicle: cFS CF app

Entity 25, channel 0, cycled at 10 Hz by the scheduler. Receives into **/ram**; sends from **/cf/demo**, **/ram** or the poll directory **/cf/poll_dir**.

/cf is NFS from the host: files there are visible in root/cf_rtems.

Class 2 needs both paths at once. A Class 2 downlink sends ACKs and NAKs up; a Class 2 uplink sends them down. So the Yamcs **tc-out** link must be enabled for every uplink and for any Class 2 transfer. The scripts switch it on and back off themselves.

Both stations see every PDU. COSMOS decodes them as CF PDU_CH0 packets (kind, class, sequence, offset) and counts them in CF housekeeping; Yamcs reassembles the file. One engine, two consoles.

NAMES AND PLACES

Where files are, and the rules the vehicle's file system imposes

On the vehicle

- **/cf/demo/README_DRACO7RT.txt** (795 B) and **/cf/demo/sensor_dump_64k.bin** (65 536 B): the files to download; their SHA-256 is in /cf/demo/SHA256SUMS
- **/ram/<name>**: where uplinked files go (RAM disk, cleared at reboot)
- **/cf/poll_dir**: anything placed here is sent to the ground as Class 2 within 21 s, then deleted
- Entity id **25**, channel 0

On the ground (Yamcs)

- Bucket **cfdpUp**: put a file here to send it up
- Bucket **cfdpDown**: every received file lands here; poll-directory files land as poll_dir/<name>
- Entity **ground** = 21; remote entity **draco** = 25
- Service name **cfdp** (used in the REST paths)

Two hard rules

- **Uplink to /ram, never to /cf.** The vehicle's NFS client can read, list and delete on /cf but cannot create a file there; a create fails and a mkdir freezes the FM app until the next boot
- **File names: 19 characters at most** (the flight OS limit). Longer names fail with error −104 before a byte moves

A remote path on the vehicle is written in full: **/ram/mytable.tbl**. A received file on the ground is named by what the sender asked for; Yamcs numbers duplicates rather than overwriting.

WORK INSTRUCTIONS · BEFORE ANY TRANSFER

Three things to settle before you move a file

1 · Who is commanding

- The ground side of every transfer is the **Yamcs CFDP engine**; its PDUs reach the vehicle only through the Yamcs command link **tc-out**, which is **off by default**
- Any **uplink** and any **Class 2** transfer in either direction needs it on: front door › Demo control › COMMANDING STATION › **Switch to Yamcs**, **Run** (strip cell COMMANDING reads YAMCS)
- A **Class 1 downlink** is the one case that works with it off
- One commanding station at a time: switch back to COSMOS when you are done

2 · Is CF alive

- COSMOS: Viewer › Mission › **File transfer (CFDP)**: the Channel 0 counters show numbers, not null; the faults box is all zeros
- Yamcs sidebar › **Links**: udp-in **In** counting; sidebar › **File transfer** opens with **cfdp ▾** selected at the top
- The vehicle needs no particular mission state: CF runs from boot, before and after launch

3 · Name and place

- Uplinks go to **/ram/<name>**; the name is **19 characters or fewer**
- Downlinks come from a full vehicle path, for the demo **/cf/demo/README_DRACO7RT.txt** or **/cf/demo/sensor_dump_64k.bin**
- Pick the class: Reliable (Class 2) for anything that must be exact; Class 1 for a quick dump

The two scripted routes (Script Runner **cfdp_demo.rb**, front-door **CFDP file-transfer test**) switch commanding to Yamcs themselves and hand it back to COSMOS at the end, whatever happens in between.

WORK INSTRUCTION · YAMCS · UPLINK

Send a file up from the Yamcs File transfer page

Open

- Viewer › Yamcs › **File transfer (CFDP)**, Yamcs sidebar › **File transfer**, or **/yamcs/file-transfer?c=draco__realtime**
- Strip cell COMMANDING must read **YAMCS** on the front door (slide 6)
- Stage the file first: bucket page **/yamcs/storage/buckets/cfdpUp** › **Upload files** (top bar). Clear saved data and Start stack keep it (they empty only cfdpDown)

Send

1. Press **New transfer** (top bar, beside cfdp ▾). Local entity **ground [id=21]**, remote entity **draco [id=25]** are preset
2. In the bucket list choose **cfdpUp** and click the file; its name fills **Local filename**
3. Type the **Remote filename**: **/ram/** plus a name of 19 characters or fewer, e.g. /ram/mytable.tbl
4. **Reliable** on = Class 2 (the default), off = Class 1
5. Press **UPLOAD**

You should see

- A new row at the top of tab **Transfers**, Direction →, size growing, then a green check and **COMPLETED**
- COSMOS CFDP screen, box Channel 0 traffic: **RECV_PDU0** and **RECV_FD0** climb while it runs; Class 2 also moves **SENT_PDU0** (the vehicle's ACK and Finished)
- Measured 27 October: 82 B Class 1 in 1.0 s; 64 KiB Class 2 in 6.1 to 7.3 s

Prove it arrived: downlink the same path back (next slide) and compare; the scripted round trip does exactly that and checks the SHA-256. Anything in /ram is gone at the next vehicle boot.

WORK INSTRUCTION · YAMCS · DOWNLINK

Bring a file down: the vehicle sends, Yamcs receives into cfdpDown

Not the DOWNLOAD button

- The dialog's **DOWNLOAD** sends a "proxy put" request. The flight CF app (3.0) does not implement it, so that row would wait and fail
- A downlink is always started **on the vehicle** with the command **CF TX_FILE**; Yamcs then shows it as an incoming (←) transfer

Send CF TX_FILE

1. Yamcs sidebar › **Commanding › Send a command** › folder **CF/** › **TX_FILE**, fill step 2, **Send**; or Telemetry › Displays › DRACO › **cfdp.opi**; or the COSMOS CFDP screen buttons (next slide)
2. CLASS **CLASS_1** or **CLASS_2**, KEEP 1, CHAN 0, PRIO 0, DEST_ID **21**
3. SRC_FILENAME the full vehicle path, e.g. **/cf/demo/sensor_dump_64k.bin**; DEST_FILENAME the name you want on the ground (19 characters or fewer)
4. Class 2 needs Yamcs commanding on; Class 1 does not

You should see

- Yamcs › File transfer: a new ← row, **Remote file** = your SRC path, then **COMPLETED**; the **Local file** link opens the object in bucket **cfdpDown**
- COSMOS CFDP screen: **SENT_PDU0** and **SENT_FD0** climb; Last PDU from the vehicle › PDU_KIND ends on EOF
- Measured: 795 B Class 1 in 1.0 s; 64 KiB Class 2 in 3.1 to 5.1 s, SHA-256 equal to /cf/demo/SHA256SUMS

KEEP 1 leaves the source file on the vehicle (KEEP 0 deletes it after a successful send). Yamcs numbers a repeated destination name instead of overwriting it.

WORK INSTRUCTION · COSMOS · THE CFDP SCREEN

The COSMOS page: watch the vehicle, press a button to start a downlink

Open

- Viewer › Mission › **File transfer (CFDP)**: window **CF CFDP**
- or Viewer › Home › Mission home › **Open File transfer (CFDP)**
- or Telemetry Viewer: Select Target **CF**, Select Screen **CFDP**, **Show**

What is on it

- **Channel 0 traffic**: PDUs and file-data PDUs sent and received, NAKs both ways, PDU errors
- **Faults and limits (should stay 0)**: file and limit faults
- **Queues, polling, engine**: queues, POLL_COUNT0, FLAGS0, command counters
- **Last PDU from the vehicle**: PDU_KIND, mode, entities, sequence, offset
- Graph boxes **File data bytes** (sent = downlink, received = uplink) and **PDU counters**

Buttons

- Bottom row, box “Downlink the demo files”: **README, Class 1**, **README, Class 2**, **64 KiB dump, Class 1**, **64 KiB dump, Class 2**: each sends one CF TX_FILE; the file lands in bucket cfdpDown
- **Yamcs File transfer page**: the receiving side, new tab
- **Run cfdp_demo.rb (all cases)**: Script Runner on the procedure (next slide); **Home**: FOGI HOME

The Class 2 buttons need Yamcs commanding on (the vehicle waits for the ground’s ACK and Finished); the Class 1 buttons work either way. The same page is in Yamcs: Telemetry › Displays › DRACO › **cfdp.opi**, generated from this screen.

WORK INSTRUCTION · COSMOS · SCRIPT RUNNER

cfdp_demo.rb: five transfers with byte checks, in 37 seconds

Run

1. Viewer › Tools › **Script Runner**, menu File › Open › **FOGI/procedures/cfdp_demo.rb**; or the CFDP screen's **Run cfdp_demo.rb (all cases)** (new tab)
2. Press **Start** at the top of Script Runner. No vehicle preparation is needed; it must simply be booted
3. It switches commanding to Yamcs for the run and switches it back at the end, even on failure

What it does

- **U1** Class 1 uplink of a small text file to /ram/cos_up_c1.txt, then checks RECV_PDU0 moved
- **U2** Class 2 uplink of 64 KiB to /ram/cos_up_c2.bin
- **D1** Class 1 downlink of the README; **D2** Class 2 downlink of the 64 KiB dump; each SHA-256 compared with the vehicle's SHA256SUMS
- **R2** round trip: the U2 file comes back from /ram and must match what was sent
- Counters: PDUs moved both ways, no fault counter changed

You should see (27 Oct, 03:16 UTC)

- U1 82 B 1.0 s · U2 65 536 B 7.3 s · D1 795 B 1.0 s · D2 5.1 s · R2 5.1 s, every SHA-256 "matches"
- sent PDUs +286, received PDUs +142, faults unchanged
- Last lines: **==== CFDP DEMO PASS: U1 U2 D1 D2 R2 ====** then "Yamcs tc-out disabled again"

The script talks to the Yamcs engine over its REST API (10.5.5.202:8090) for the uplinks and the received files, and to the vehicle through COSMOS for CF TX_FILE and the housekeeping checks: the two stations working on one transfer.

WORK INSTRUCTION · FRONT DOOR

One button runs every case and records the evidence

Run

1. Front door https://draco.efsi.com, sign in, scroll to the bottom panel **Demo control**
2. Group YAMCS › **CFDP file-transfer test**, then **Run** in the dialog
3. The log streams in the job pane (tab **Job output**); strip cell COMMANDING flips to YAMCS during the run and back to COSMOS after; the job badge ends **OK**
4. About a minute and a half (91.9 s on 27 October)

The eleven checks

- **SCH**: CF is being woken 10 times a second (schedule table activity rate)
- **UP1, UP2, RT2**: Class 1 and Class 2 uplinks to /ram, the 64 KiB file back down byte-exact
- **DOWN1, DOWN2**: README and 64 KiB dump down, SHA-256 against the vehicle's SHA256SUMS
- **POLL**: a file dropped in the poll directory arrives on its own
- **LOSS2, LOSS1**: 20 % of the vehicle's PDUs discarded in the downlink tee: Class 2 completes, Class 1 fails as designed
- **PDUTLM**: PDU packets decoded in COSMOS; **FAULTS**: no CF fault counter moved

What it leaves behind

- **cFS/evidence/runs/cfdp_<UTC>/run.json**: verdict, every check with its numbers, the timings table
- Requirements dRT-CFDP-001 to 005 marked **VERIFIED-RUN** (or RUN-FAIL) in the requirements table and the trace
- The transfers themselves on the Yamcs File transfer page and the files in cfdpDown

One action at a time: while another runs, every Demo control button is greyed. From a shell: **python3 run/yamcs/cfdp_demo.py --record** (from /home/efsi/draco7_0).

WORK INSTRUCTION · THE POLL DIRECTORY

Drop a file on the vehicle and it comes down by itself

How it works

- CF polls **/cf/poll_dir** on the vehicle; whatever appears there is sent to ground entity 21 as **Class 2** and deleted after the send succeeds
- /cf is served over NFS from the host directory **root/cf_rtems**, so a file written there from a host shell is on the vehicle at once
- This is how an application would hand files to the ground without an operator: logs, dumps, images

Do it

1. Yamcs commanding on (it is Class 2): front door › Demo control › **Switch to Yamcs**
2. On the host: **cp myfile.txt /home/efsi/draco7_0/root/cf_rtems/poll_dir/**, name 19 characters or fewer
3. Wait up to 21 s (the poll interval); nothing else to send

You should see

- Yamcs › File transfer: a ← row with Remote file **/cf/poll_dir/<name>**, then COMPLETED
- The file in bucket **cfdpDown** as **poll_dir/<name>**
- The file gone from the host directory (CF removed it)
- COSMOS CFDP screen: **POLL_COUNT0** and **SENT_FD0** moved
- Measured 27 October: a 43 B file sent 15 s after the drop

Write the file in one step (copy or move it in; do not build it in place) so CF never picks up a half-written file. The same poll directory also appears on the vehicle when the host puts files there before boot.

WORK INSTRUCTION · THE LOSS DEMONSTRATION

Throw away one PDU in five and watch Class 2 repair the file

Inject loss

1. The downlink tee (container **draco7-downlink-tee-1**) copies every vehicle packet to COSMOS and Yamcs. A file **/tmp/tee.drop** inside it makes it discard a share of one packet type
2. From the host: **docker exec draco7-downlink-tee-1 sh -c 'echo "08c2 20" > /tmp/tee.drop'** drops 20 % of the CF channel-0 PDUs (stream id 0x08C2) to every station
3. COSMOS CFDP screen: **64 KiB dump, Class 2**, then **64 KiB dump, Class 1** (or TX_FILE)
4. Stop: **docker exec draco7-downlink-tee-1 rm /tmp/tee.drop** (re-read every second)

Class 2, measured 27 October

- Yamcs sends NAKs listing the holes; the vehicle resends those segments; **32 segment requests** answered
- COSMOS: **RECV_NAK_SR0** climbs on the vehicle; **SENT_FD0** keeps rising after the first pass
- **COMPLETED**, 65 536 B, SHA-256 exact, in 20.7 s instead of 3.1 s

Class 1, same loss

- No NAKs: the vehicle sends once and forgets
- Yamcs waits for the missing bytes, then marks the row **FAILED, CHECK_LIMIT_REACHED**
- 49 920 of 65 536 B arrived; the holes stay. The protocol worked as specified: that is what “unacknowledged” means

The front-door test does all of this itself (cases LOSS2 and LOSS1), including removing the drop file; use the manual steps when you want to show it live on the screens.

TELEMETRY

What the vehicle tells you about a transfer, and where to read it

Packet	Items that matter	Where
CF HK_TLM_PKT (0x08B0), once a second	Channel 0: SENT_PDU0, SENT_FD0, RECV_PDU0, RECV_FD0, SENT_NAK_SR0, RECV_NAK_SR0, RECV_PDU_ERROR0 / DROPPED0 / SPURIOUS0; faults FAULT_FILE_OPEN0, READ0, WRITE0, CRC_MISMATCH0, FILE_SIZE_MISMATCH0, ACK_LIMIT0, NAK_LIMIT0, INACTIVITY_TIMER0; queues QPEND0, QTXA0, QTXW0, QRX0, QHIST0, QFREE0; POLL_COUNT0, FLAGS0; command counters	COSMOS screen CF/CFDP (Viewer › Mission › File transfer (CFDP)); Yamcs Telemetry › Parameters /CF/HK_TLM_PKT/… and display DRACO/cfdp.opi; Grafana not yet
CF PDU_CH0 (0x08C2), one per PDU sent	The CFDP header bits (version, type, direction, transmission mode, CRC flag, lengths) plus derived PDU_KIND (METADATA, FILE_DATA, EOF, FINISHED, ACK, NAK, …), SEQ_NUM, DST_EID, FILE_OFFSET	COSMOS Packet Viewer and the CFDP screen (“Last PDU”); in Yamcs the raw PDU goes to the engine through stream cfdp_in
CF PDU_CH1 (0x08C3)	Same layout, channel 1 (unused by the demo)	COSMOS only
Events (0x0808), app CF	Transaction start, completion and every fault with its reason, e.g. a NAK limit reached or a file that could not be opened	COSMOS Log Messages and the viewer's Events dock; Yamcs › Events (source CF)
Yamcs transfer record	Direction, state (RUNNING, COMPLETED, FAILED with reason), bytes transferred, reliable flag, remote path, local object	Yamcs › File transfer, tabs Transfers and Action log; REST /yamcs/api/filetransfer/draco/cfdp/transfers

A healthy downlink: SENT_FD0 rises, PDU_KIND ends on EOF, then (Class 2) FINISHED arrives at the ground and the vehicle’s RECV_PDU0 ticks for the ACK. A healthy uplink: RECV_FD0 rises. NAK counters move only under loss.

TROUBLESHOOTING

What you see, what it means, what to do

Symptom	Cause	Do
Uplink row stays RUNNING at 0 B; RECV_PDU0 does not move	Yamcs commanding (tc-out) is off: the PDUs never leave the ground	Front door › Demo control › COMMANDING STATION › **Switch to Yamcs**, then start the transfer again
Class 2 downlink never reaches COMPLETED; vehicle keeps resending EOF, then ACK_LIMIT0 ticks	Same cause: the vehicle waits for the ground's ACK and Finished, which need the uplink	**Switch to Yamcs**; or send it as Class 1 (CFDP screen: a Class 1 button)
CF event "failed to create file"; or FM stops answering after a CREATE_DIR	A write to /cf: the vehicle's NFS client cannot create there (a mkdir freezes FM until reboot)	Uplink to /ram/… ; a hung FM needs a vehicle restart (front door › Demo control › STACK › **Fresh vehicle**)
TX_FILE rejected with error −104	File name longer than 19 characters (OS limit)	Shorten the name
Row FAILED, CHECK_LIMIT_REACHED, size short of the total	Class 1 under loss: the holes cannot be asked for	Expected; resend as Class 2

Failed rows, unacknowledged transfers, pages that do not open

Symptom	Cause	Do
Row FAILED, NAK_LIMIT_REACHED	More loss than 10 NAK rounds can repair (or the pre-27-October CF snapshot, which rejected NAKs with more than 4 gaps)	Reduce the injected loss; check the build has the cf_codec.c fix
“The Finished PDU has not been acknowledged”	The file is complete on the ground; the vehicle’s ACK never arrived (commanding switched off mid-transfer, or the vehicle reset)	The received file is good; nothing to redo
Yamcs shows “404: Not Found · It looks like this page doesn't exist.”	Wrong URL; the page is /yamcs/file-transfer (no instance in the path)	Add ?c=draco__realtime; or use Viewer › Yamcs › **File transfer (CFDP)**, or Yamcs sidebar › **File transfer**
COSMOS CFDP screen all “null”	The page is still loading, or CF housekeeping is not reaching COSMOS	Wait 10 s; then front door › Demo control › HEALTH › **Verify** (= bootstrap-draco.sh --verify): tee and links

WHAT WAS PROVED · 27 OCTOBER 2026

Eleven checks from the front door, five from COSMOS, all passed, all recorded

Case (run cfdp_20261027_025937, 91.9 s)	Bytes	Time	Result
SCH: CF woken 10 Hz	–	–	35.4 activities/s = 25 + 10
UP1 Class 1 uplink, back down	81	0.1 s + 1.0 s	identical
UP2 Class 2 uplink	65 536	6.1 s	Finished PDU closed it
RT2 round trip Class 2	65 536	3.1 s	SHA-256 equal
DOWN1 README Class 1	795	1.0 s	SHA-256 = SHA256SUMS
DOWN2 64 KiB Class 2	65 536	3.1 s	SHA-256 = SHA256SUMS
POLL directory, Class 2	43	15.2 s	identical, source removed
LOSS2 Class 2 at 20 % PDU loss	65 536	20.7 s	COMPLETED, 32 NAK segments
LOSS1 Class 1 at 20 % PDU loss	49 920 of 65 536	23.5 s	FAILED, as designed
PDUTLM, FAULTS	–	–	1 325 PDU packets in COSMOS; faults 0 → 0

Also

- COSMOS Script Runner **cfdp_demo.rb** at 03:16 UTC: U1 U2 D1 D2 R2 PASS in 37 s, sent +286 / received +142 PDUs, faults unchanged
- Requirements **dRT-CFDP-001** (10 Hz wake-up), **002** (PDUs in telemetry, Class 1), **003** (Class 2 uplink), **004** (downlink and poll directory, byte-exact), **005** (Class 2 recovery under loss): all **VERIFIED-RUN**
- Trace: 80 requirements, 80 traced, 0 untagged; DVR gate PASS
- Stack health check (--verify) clean after the plugin upgrades

LIMITS

What this CFDP does not do, and the numbers that bound it

By design of the flight app

- **No ground-initiated download**: CF 3.0 has no proxy put, so a downlink is a TX_FILE command (or the poll directory)
- **No directory listing** over CFDP; you must know the path
- CF emits **no CRC** on its PDUs; integrity comes from the file checksum in EOF and, on the ground, the SHA-256 checks the scripts make
- One-byte entity ids (ground 21, vehicle 25), PDUs of at most 512 B (480 B of file data)

By the vehicle’s file system

- Uplinks land only in **/ram**, which is cleared at every boot; moving a file into /cf is not possible from the vehicle
- Names of **19 characters** or fewer
- Class 2 in either direction, and any uplink, need **Yamcs commanding on**; there is one commanding station at a time and nothing enforces it

Measured bounds

- Downlink about **21 kB/s** (64 KiB in 3.1 s); uplink about **10 kB/s** (6.1 s), set by the engine’s 40 ms gap between PDUs
- Class 2 recovers 20 % loss with **10 NAK rounds** of 5 s; beyond that the transfer fails cleanly
- Inactivity timeout 60 s on both sides; a stalled transfer is cancelled, not left hanging
- Both channels are on the same 10 Hz wake-up; channel 1 is wired but unused

None of these is a defect of the integration: they are the flight app’s feature set and the simulator’s file system, stated so that a demonstration never promises more than the stack does.

ENGINEERING RECORD

What was changed to make it work, and where it lives

Flight (cFS)

- **Schedule**: CF_WAKE_UP at 10 Hz (sch_def_msgtbl.c entry 26, sch_def_schtbl.c); without it CF never cycles
- **Downlink**: TO_LAB subscribes to the channel-0/1 PDU message ids 0x08C2/0x08C3 (to_lab_sub.c)
- **Table** cf_def_config.c: entity 25, poll directory, 5 s ACK/NAK timers, 480 B chunks, temp files in /ram
- **Fix** cf_codec.c: NAK segment requests were capped at 4 (CF_PDU_MAX_TLV) instead of CF_PDU_MAX_SEGMENTS; any NAK with more gaps was “nak pdu too short”. Matches nasa/CF main

Ground

- **Yamcs**: CfdpService “cfdp” in yamcs.draco.yaml; streams cfdp_in (PDUs out of packet 0x08C2) and cfdp_out (PDUs wrapped as CI_LAB commands) in etc/extra_streams.sql; buckets cfdpUp, cfdpDown
- **COSMOS**: CF PDU_CH0/PDU_CH1 packets, cfdp_pdu_conversion.rb, screen CF/CFDP, HOME button, procedure cfdp_demo.rb (plugins openc3-cosmos-draco 1.0.11, openc3-fogi-open-source 1.1.9)
- **Yamcs display** DRACO/cfdp generated from the COSMOS screen; XTCE regenerated (cosmos2xtce)
- **Tee**: loss injection through /tmp/tee.drop
- **Front door**: action cfdp_demo; viewer pages in both groups

Evidence and documents

- Requirements **dRT-CFDP-001..005** (verify/requirements.yaml, requirements.csv), checks V-CFDP-* in run/yamcs/cfdp_demo.py --record, verdicts applied by apply_verdicts.py
- Demo files and their checksums: tools/cfdp/make_demo_files.py → root/cf_rtems/demo
- Plan and findings: claude-notes/CFDP-PLAN.md, CFDP-RESEARCH.md; conventions in CLAUDE.md (file-system rules, NAK fix, modified-screen trap)
- This deck: powerpoint/decks/cfdp-guide, also at /slides/

No evaluation-licensed software is involved: NASA CF (Apache 2.0) on the vehicle, Yamcs (AGPL) as the engine, COSMOS open-source edition on the other station.