ASAP Proof Plan for L7 Using Existing CCSDS L2 TX/RX in GRC

1 Goal

Produce, in the shortest time, a reproducible evidence package that demonstrates a working Application Layer (L7) over your existing CCSDS Data-Link (L2) TX/RX in GRC. The evidence includes:

- A "golden vector" CCSDS Space Packet (hex), its expected L7 MIC (CRC-32C), the first bytes of the corresponding CADU (with ASM), and the recovered Space Packet after the RX chain.
- Logs proving APID sequence continuity over long runs (e.g., 10 k packets) and zero L7 integrity failures at nominal bench rates.
- Optional channel-impairment or SDR results summarized as Es/N0 vs. FER, showing that the coding profile meets the link budget while maintaining zero L7 MIC failures.
- An operator demo showing a command that triggers a predictable telemetry change over the same GRC harness.

2 Directory Skeleton (Artifacts Organized)

Create a lightweight structure to keep all artifacts consistent:

- `golden/`: golden Space Packet (binary + hex), expected MIC, first 64 bytes of CADU (hex).
- `runs/run_0001/`: results for the first trial (CSV of per-packet checks, summary, CADU binaries, optional pcap).
- `runs/run_0002/`, etc.: subsequent trials (impairments or SDR).
- `tools/`: sender/receiver harness logic (see §4).

3 Golden Vector (One-Time)

Construct a deterministic, well-specified CCSDS Space Packet to serve as a golden reference:

1. **Primary header (6 bytes, big-endian):** set Version, Type, Secondary Header Flag, APID; use "standalone" sequence flags; assign a known sequence count; set packet length to (`user_data + optional_secondary_header + MIC`)−1.

2. **Deterministic user data:** include an 8-byte time tag (big-endian), a fixed 16-byte pattern (e.g., byte values 0x00 .. 0x0F), and an 8-byte counter (big-endian).

3. **MIC (L7 integrity):** compute CRC-32C over the *user data only* (not including the primary header), append as 4 bytes (big-endian).

4. **Persist artifacts:** store the raw packet (binary), a hex dump, the expected MIC value, and later the first 64 bytes of the CADU produced by the framer.

4 Sender/Receiver Harnesses (Logic Only)

Provide two minimal processes that bracket your GRC graph via UDP. These operate purely at L7 and do not inspect L2/L1 internals.

L7 Sender (Conceptual Logic)

- Inputs: APID constant, starting sequence count, output UDP host/port for the GRC Socket PDU Source.
- Loop for a configured number of packets (e.g., 10 000):
  1. Build user data: time_tag (8 B) + fixed_pattern (16 B) + payload_counter (8 B).
  2. Compute MIC = CRC-32C(user_data) and append to user_data.
  3. Assemble CCSDS primary header with current sequence count and computed packet length.
  4. Concatenate header + (user_data + MIC) → Space Packet bytes.
  5. Send Space Packet bytes as a single UDP datagram to GRC.
  6. On the first iteration, persist the Space Packet to `golden/`.
  7. Increment sequence count modulo 14 bits.
  8. Optional: throttle by a fixed inter-packet interval to set bench rate.

L7 Receiver (Conceptual Logic)

- Inputs: UDP host/port matching the GRC Socket PDU Sink, expected APID.
- For each received Space Packet:
  1. Parse primary header; extract APID, sequence count, packet length.
  2. Split user_data (packet length bytes) from the trailing 4-byte MIC.
  3. Recompute CRC-32C(user_data) and compare with received MIC.
  4. Check APID matches expectation and that sequence count increments contiguously (modulo 14 bits).

5. Log a CSV row with: packet index, APID, sequence count, MIC result (0/1), APID match (0/1), sequence continuity (0/1).

- After N packets, write a human-readable summary: total packets, MIC failures, sequence gaps, elapsed time, and derived packets-per-second.

5 Minimal GRC Plumbing (Bytes-Only First)

Use your existing CCSDS L2 blocks; start with the pure byte path to get instant results.

TX Chain (Bytes to CADU)

1. **Socket PDU Source (UDP)**: receives Space Packets from the sender harness.
2. **PDU to Tagged Stream**: key = `packet_len`.
3. **CCSDS Framer** (TM/TC or AOS): perform framing/segmentation, set FHP, prepend ASM for CADU.
4. **Randomizer**: exclude ASM from scrambling.
5. **FEC**: RS(255,223) (and interleave if used), followed by convolutional encoding (or alternate LDPC/Turbo as per profile).
6. **Artifact taps**: record post-framer bytes (CADUs) to a file; enable Tag/Message Debug after each major transform.

RX Chain (CADU to Bytes)

1. **FEC Decode**: Viterbi (with depuncturing if used) then RS decode (with deinterleave if used).
2. **De-randomizer**: exclude ASM.
3. **CCSDS Deframer**: reconstruct Space Packets from frames.
4. **Tagged Stream to PDU → Socket PDU Sink (UDP)**: emits recovered Space Packets to the receiver harness.
5. **Artifact taps**: record pre-deframer bytes (incoming CADUs) to a file.

6 First Proof Run (Tabletop, No RF)

1. Start the receiver harness; it begins logging per-packet checks and a summary.
2. Start the sender harness; it transmits a fixed number of Space Packets (e.g., 10 k).
3. Expected outcome for acceptance:
   - *Zero* MIC failures and *zero* sequence gaps across all packets.
   - Byte-identical recovery of the first golden packet (header + user data + MIC).

4. Persist the first 64 bytes of the TX CADU stream as the golden CADU snippet.

7 Add Realism: Channel Impairments or SDR

Simulated Channel

Insert a channel model *after* symbols/RRC (when you enable the waveform path) with:

- Additive noise configured to achieve target Es/N0 test points.
- Carrier frequency offset within a realistic range.
- Clock/timing offset representative of your hardware.

Collect the same run artifacts (CSV, summary, CADUs) and report FER from the deframer statistics, alongside the invariant L7 results (MIC and sequence).

SDR Loopback

When ready, replace the channel model with SDR TX/RX (e.g., HackRF/Pluto) using appropriate fixed attenuators and DC blocks. Keep gains conservative to avoid clipping, verify constellation lock, then repeat the 10 k packet run and record results as above.

8 Network/Transport Observability (Optional)

For bench traceability, capture the UDP ingress/egress legs while the run executes. Store the capture with the corresponding `run_XXXX` folder. The pcap, together with CADU binaries and CSV logs, constitutes an auditable trail from L7 ingress to L7 egress.

9 Es/N0–FER Reporting (Slide-Ready)

- In waveform mode, quote the configured Es/N0 at the channel and report measured FER at the deframer, plus the invariant L7 integrity results.
- In bytes-only mode (no symbols), simply state the intended Es/N0 test point(s) for future waveform trials and present the L7 integrity/sequence results.

10 Operator Demo (When Showing Live)

Connect your ground tool (e.g., COSMOS) to the same bench UDP ports used by the harness. Demonstrate a command that flips a known parameter and triggers a deterministic telemetry change. Confirm that the telemetry returns through the same GRC L2/L1 path, and archive the operator action alongside the packet-level artifacts.

11 Acceptance Checklist (Copy Into README)

- **Application integrity:** 10 k packets; MIC failures = 0; sequence gaps = 0.
- **Artifacts present:** golden Space Packet (hex), expected MIC, first CADU bytes, CADU TX/RX binaries, per-packet CSV, summary text, and (optionally) bench pcap.

- **Impairments/SDR:** at quoted Es/N0 points, measured FER meets budget; L7 MIC remains zero-fail.
- **Operator proof:** command → expected telemetry change via the same GRC link.

12 Notes on CRC-32C and Determinism

- Ensure the MIC covers precisely the user data bytes (not the Space Packet primary header) and is stored big-endian.
- Use a stable time tag representation (e.g., big-endian UNIX nanoseconds) to make the golden packet reproducible on demand.
- Fix inter-packet spacing (or total count) to create deterministic and comparable runs across configurations.