Technical Report: High-Fidelity Landing Gear Digital Twin & Telemetry Pipeline

Author: Mohammed Bello Sani
Institution: Air Force Institute of Technology (AFIT), Kaduna
Date: December 2025
Repository: [Landing-Gear-Digital-Twin]

1. Executive Summary

This project represents the development of a high-fidelity **Cyber-Physical System (CPS)** for a retractable aircraft landing gear assembly (Dornier 228 class). Moving beyond isolated component analysis, this Digital Twin architecture integrates **multi-body physics simulation** with a **Hardware-in-the-Loop (HIL)** style telemetry pipeline.

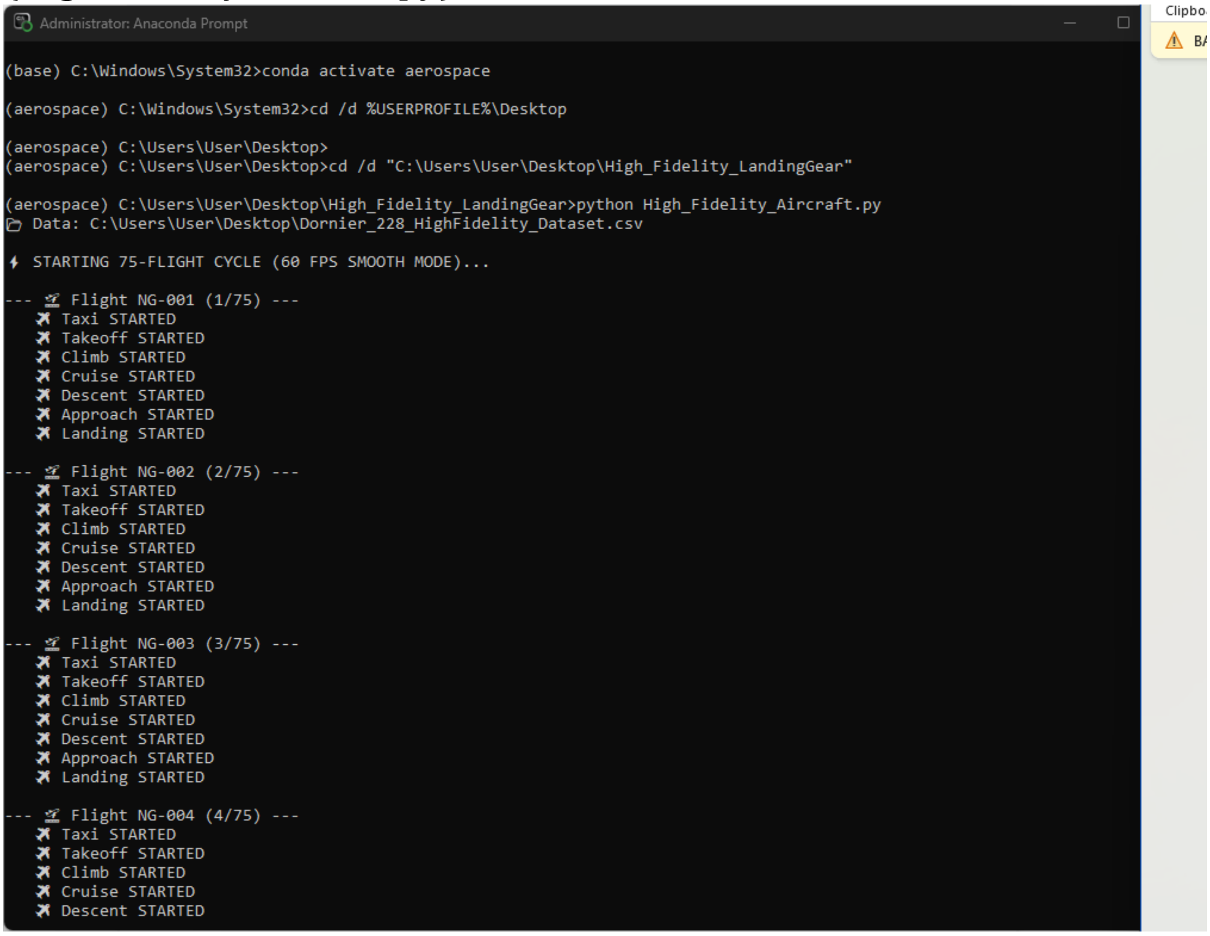
The system features a "Robot Aircraft" Flight Management System (FMS) that autonomously drives flight scenarios, injecting real-time faults (e.g., hard landings, seal degradation) into a **Simscape Multibody** physical plant. All performance metrics are streamed via **UDP** to an **InfluxDB 3** data warehouse for real-time analytics.

2. System Architecture

The architecture is designed as a distributed system with three distinct nodes:

2.1 Node A: The Flight Management System (FMS)

- **Core:** Python-based autonomous agent (High_Fidelity_Aircraft.py).



- **Role:** Acts as the "Robot Pilot," simulating 75+ flight cycles from Taxi to Landing.
- **Physics Engine:** Calculates flight dynamics (Roll, Pitch, Heading, G-Force) and environmental stressors.
- **Fault Injection:** Probabilistically introduces failures (e.g., seal_integrity degradation) based on gear cycle count.

2.2 Node B: The Digital Twin (Physical Plant)

- **Core:** MATLAB/Simulink & Simscape Multibody (Fancy_Landing_Gear.slx).
- **Role:** Solves the equations of motion for the landing gear mechanism.
- **Actuation:** Driven by AeroTwin_Master_Console.m, which utilizes a PostgreSQL bridge to "listen" for phase changes (e.g., "APPROACH" or "TAKEOFF") and trigger physical retraction/extension sequences.

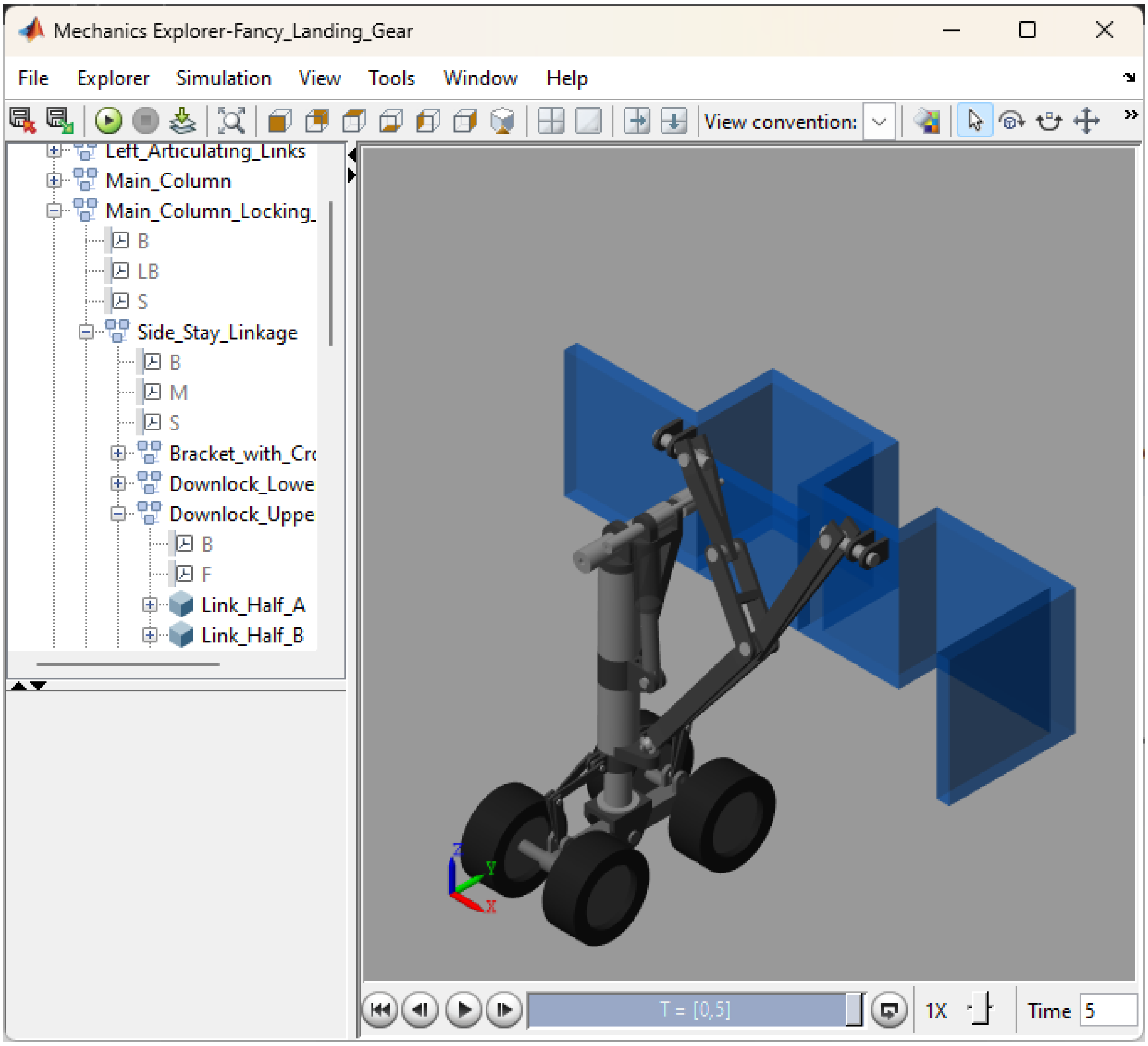
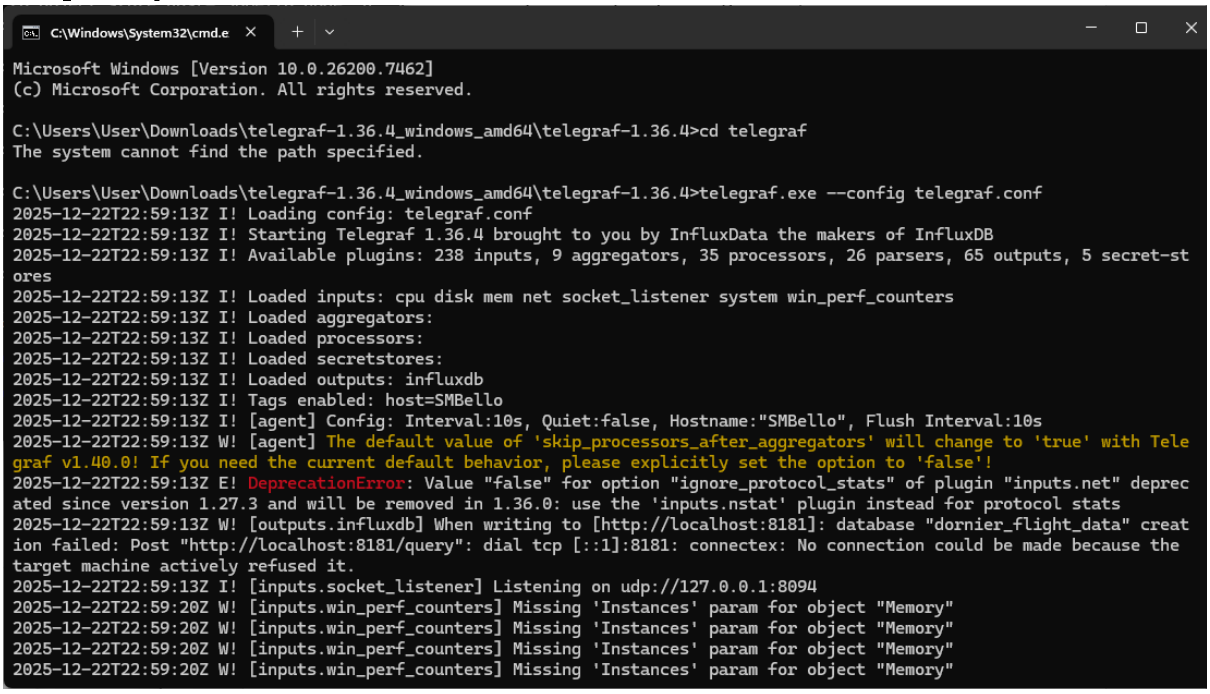


Figure 1: Simscape Multibody environment showing the active actuation of the Main Column and Side Stay linkage, driven by the AeroTwin Master Controller logic.

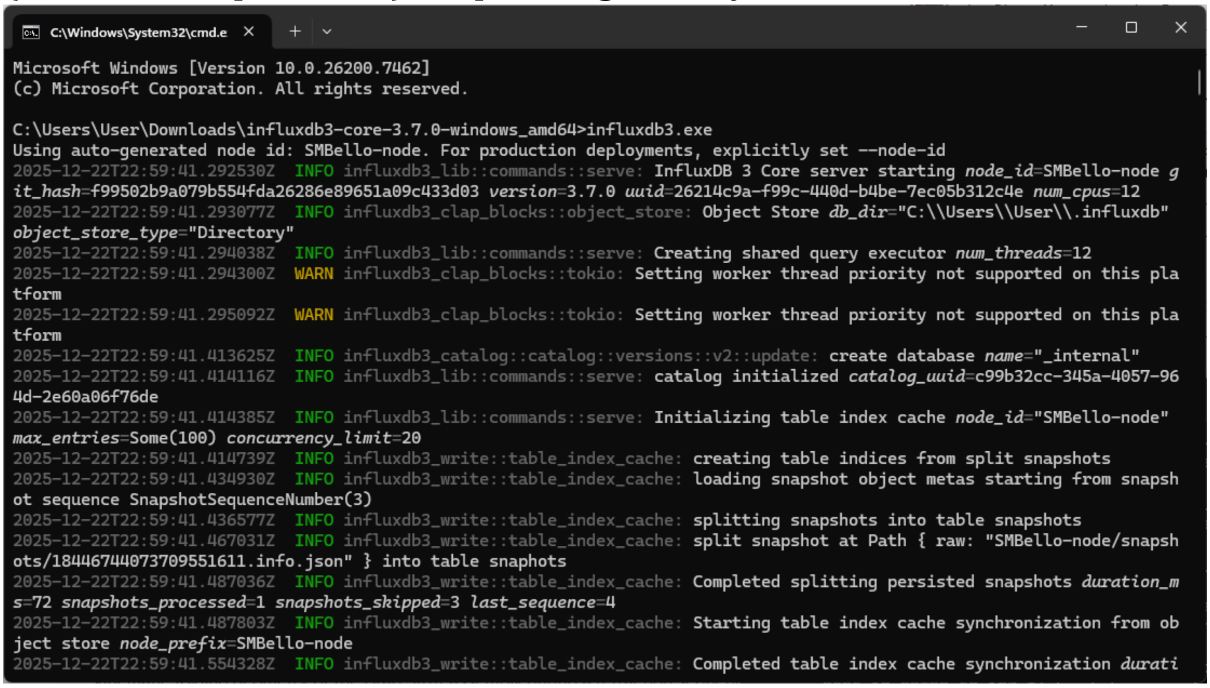
2.3 Node C: The Telemetry Pipeline

- **Visuals: FlightGear** receives UDP packets (Port 5502) for 3D visualization of the flight attitude.

- **Courier: Telegraf** listens on UDP Port 8094 to aggregate high-frequency sensor data.



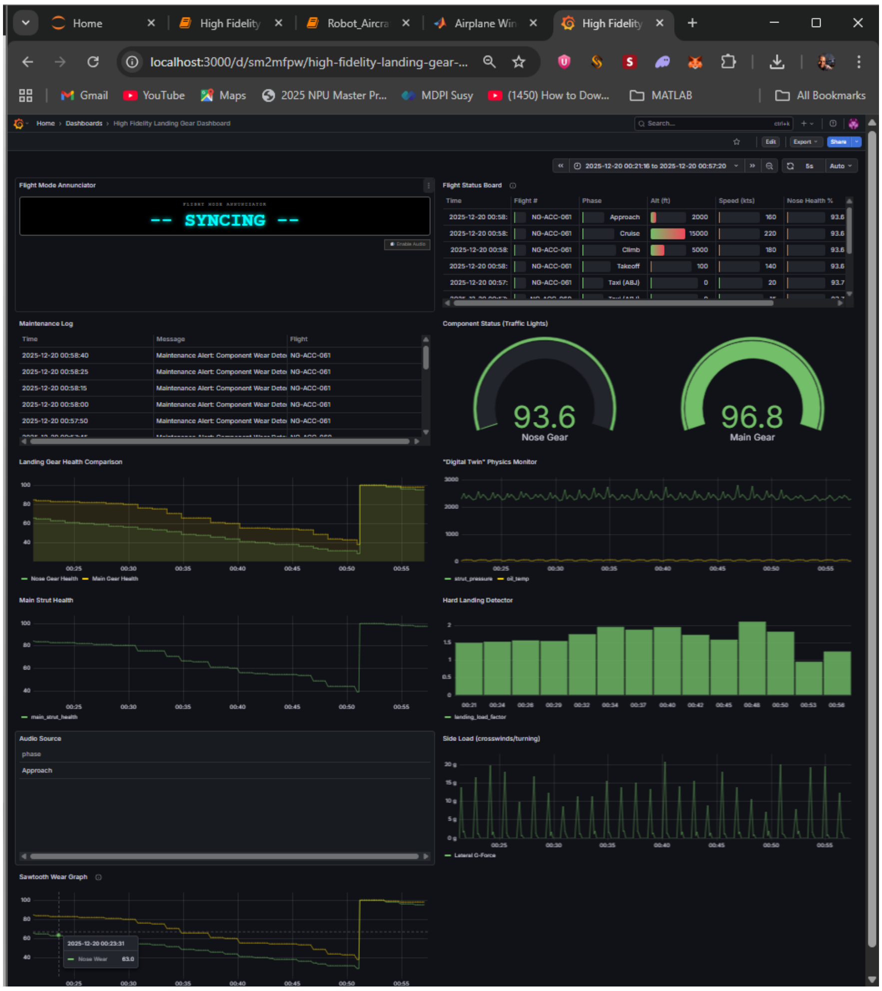
- **Warehouse: InfluxDB 3 Core** stores time-series data (nanosecond precision) for post-flight analysis.



```
Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User\Downloads\influxdb3-core-3.7.0-windows_amd64>influxdb3.exe
Using auto-generated node id: SMBello-node. For production deployments, explicitly set --node-id
2025-12-22T22:59:41.292530Z  INFO influxdb3_lib::commands::serve: InfluxDB 3 Core server starting node_id=SMBello-node g
it_hash=f99502b9a079b554fda26286e89651a09c433d03 version=3.7.0 uuid=26214c9a-f99c-440d-b4be-7ec05b312c4e num_cpus=12
2025-12-22T22:59:41.293077Z  INFO influxdb3_clap_blocks::object_store: Object Store db_dir="C:\\Users\\User\\.influxdb"
object_store_type="Directory"
2025-12-22T22:59:41.294038Z  INFO influxdb3_lib::commands::serve: Creating shared query executor num_threads=12
2025-12-22T22:59:41.294300Z  WARN influxdb3_clap_blocks::tokio: Setting worker thread priority not supported on this pla
tform
2025-12-22T22:59:41.295092Z  WARN influxdb3_clap_blocks::tokio: Setting worker thread priority not supported on this pla
tform
2025-12-22T22:59:41.413625Z  INFO influxdb3_catalog::catalog::versions::v2::update: create database name="_internal"
2025-12-22T22:59:41.414116Z  INFO influxdb3_lib::commands::serve: catalog initialized catalog_uuid=c99b32cc-345a-4057-96
4d-2e60a06f76de
2025-12-22T22:59:41.414385Z  INFO influxdb3_lib::commands::serve: Initializing table index cache node_id="SMBello-node"
max_entries=Some(100) concurrency_limit=20
2025-12-22T22:59:41.414739Z  INFO influxdb3_write::table_index_cache: creating table indices from split snapshots
2025-12-22T22:59:41.434930Z  INFO influxdb3_write::table_index_cache: loading snapshot object metas starting from snapsh
ot sequence SnapshotSequenceNumber(3)
2025-12-22T22:59:41.436577Z  INFO influxdb3_write::table_index_cache: splitting snapshots into table snapshots
2025-12-22T22:59:41.467031Z  INFO influxdb3_write::table_index_cache: split snapshot at Path { raw: "SMBello-node/snapsh
ots/18446744073709551611.info.json" } into table snaphots
2025-12-22T22:59:41.487036Z  INFO influxdb3_write::table_index_cache: Completed splitting persisted snapshots duration_m
s=72 snapshots_processed=1 snapshots_skipped=3 last_sequence=4
2025-12-22T22:59:41.487803Z  INFO influxdb3_write::table_index_cache: Starting table index cache synchronization from ob
ject store node_prefix=SMBello-node
2025-12-22T22:59:41.554328Z  INFO influxdb3_write::table_index_cache: Completed table index cache synchronization durati
```

- **Dashboard: Grafana** visualizes critical health metrics like hyd_pressure and strut_pressure.



3. Technical Implementation

3.1 The "Sync Bridge" (PostgreSQL)

A critical innovation in this project is the **Asynchronous Sync Bridge**. Since Python (FMS) and MATLAB (Physics) run on different time steps, they share state via a **PostgreSQL** database.

- **Python** logs a "Phase Change" event (e.g., Phase: LANDING, G-Force: 2.1G).
- **MATLAB** polls the database in a "Fast Restart" loop. When it detects a LANDING event, it dynamically calculates stiffness (k) and damping (b) coefficients based on the live health data and executes the strut compression simulation.

3.2 Physics Modeling (Simscape Multibody)

The mechanical assembly is modeled with high fidelity:

- **Kinematics:** Side Stay and Drag Strut linkages are modeled using revolute and spherical primitives to replicate exact folding geometry.
- **Hydraulics:** Actuators are not ideal motion sources; they are modeled as hydraulic pistons ($F = P \times A$) sensitive to hyd_pressure drops simulated by the FMS.
- **Control:** A PID Controller (PID Deploy/Ret) regulates actuator pressure to prevent structural ringing during high-speed retraction.

3.3 Data Telemetry Stack

To emulate modern avionics, the system avoids local CSV logging in favor of a network-based approach:

1. **Generation:** Python generates telemetry at 60 Hz.
2. **Transport:** Data is serialized into **Influx Line Protocol** and broadcast via UDP.
3. **Ingestion: Telegraf** buffers the stream and writes to **InfluxDB 3**.
4. **Query: Grafana** queries the bucket for real-time "Red/Green" health status visualization.

4. Operational Scenarios

Scenario A: The "Hard Landing"

1. **Trigger:** The FMS detects gear_cycles > 50 and injects a random G-Force spike of 2.5G.
2. **Response:** The Python script triggers an email alert (SMTP) to the maintenance team.
3. **Physics:** MATLAB reads the 2.5G impact, updates the strut stiffness parameters (k_strut_current), and simulates the shock absorption capability to check for bottoming out.

Scenario B: Seal Failure

1. **Trigger:** seal_integrity drops below 0.8.
2. **Response:** The FMS drops the target hydraulic pressure from 3000 PSI to 2500 PSI.
3. **Physics:** The Simscape model receives reduced force, potentially causing a "Failure to Retract" or "Slow Retraction" event, observable in the Deploy Actuator Force scope.

5. Directory Structure

- /Models: Contains Fancy_Landing_Gear.slx (The Physics Twin).
- /Scripts:
  - High_Fidelity_Aircraft.py: The Python FMS and Telemetry Generator.
  - AeroTwin_Master_Console.m: The MATLAB Logic Controller.
- /Docs: Architecture diagrams and this Technical Report.

6. Conclusion & Future Work

This Digital Twin successfully demonstrates the fusion of **Operational Technology (OT)** - the physics simulation - with **Information Technology (IT)** - the InfluxDB/Python stack. Future iterations will replace the PostgreSQL bridge with a direct **ZeroMQ** low-latency socket for sub-millisecond synchronization between the Flight Model and the Landing Gear Physics.

Future Roadmap: Current research is focused on integrating an **Observer** into the control loop. This Virtual Sensor would compare the "Digital Twin" expected position against the "Real" sensor data to generate a "Residual Signal," which acts as an early warning system for mechanical drift or incipient failure.

6. Future Roadmap

Current research is focused on integrating an **Observer** into the control loop. This Virtual Sensor would compare the "Digital Twin" expected position against the "Real" sensor data to generate a "Residual Signal," which acts as an early warning system for mechanical drift or incipient failure.