F' Integration Test Framework Discussion

Presented by Kevin Oran on July 17, 2019

Jet Propulsion Laboratory
California Institute of Technology

Project Objectives

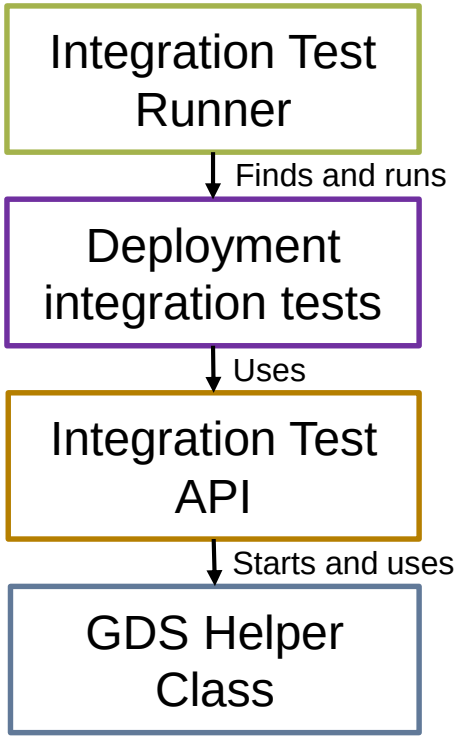
Why are we implementing a test API again?

- Provide a new implementation of an integration test API that runs on the GDS middleware.
- Ensure the new GDS Integration test API can run independently of the ground UI.
- Provide an interface to enable future support for CI/CD integration testing.
- Provide a means of getting formatted test reports.

Concept of Operations

For an Integration Test Framework

- An F’ user can use the Test Runner to run their integration tests or to incorporate tests into CI/CD for their project.
- An F’ user can develop integration tests by calling on the Integration Test API.
- The Integration Test API will use a central GDS Helper Layer to access GDS. The current implementation of this layer is called the standard pipeline.



Project Requirements

Python

- Python code shall be written to be compatible with both Python 2 and 3.
- Python code shall use named tuples if/when returning multiple results.
- Python code shall use pydoc-compatible commenting to define parameters, operation and returns.

Project Requirements

Integration Test Runner

- The Test Runner shall collect artifacts to record the condition of the tests.
  - Current list: History logs, copies of the FSW dictionaries, a copy of the FSW binary
- The Test Runner shall collect files to record the results of the tests.
  - Current list: Test logs, test reports
- The Test Runner shall support specifying a deployment directory to discover and run Integration Tests.
- The Test Runner should provide usability features to aid in CI/CD setup.
  - Test configurations, Command Line Interface

Project Requirements

Integration Test API

- The Test API shall support the following functionality for both events and telemetry:
  - Awaiting a message
  - Asserting a message is in a history
  - Asserting a message is in a history or is received before a timeout. (Await assert)
- The Test API shall provide basic assertions on event and telemetry messages.
- The Test API shall provide asserts on FSW-created timestamps.
- The Test API shall limit redundancy through the use of predicate functions for asserts.
- The Test API shall create a detailed test log while it is being used by the test cases.
- The Test API shall be performance tested to identify if it drops messages and at what point it drops messages.
- The Test API asserts shall search efficiently: Optimized for the data structures in the history and better than O(n^2).

Project Requirements

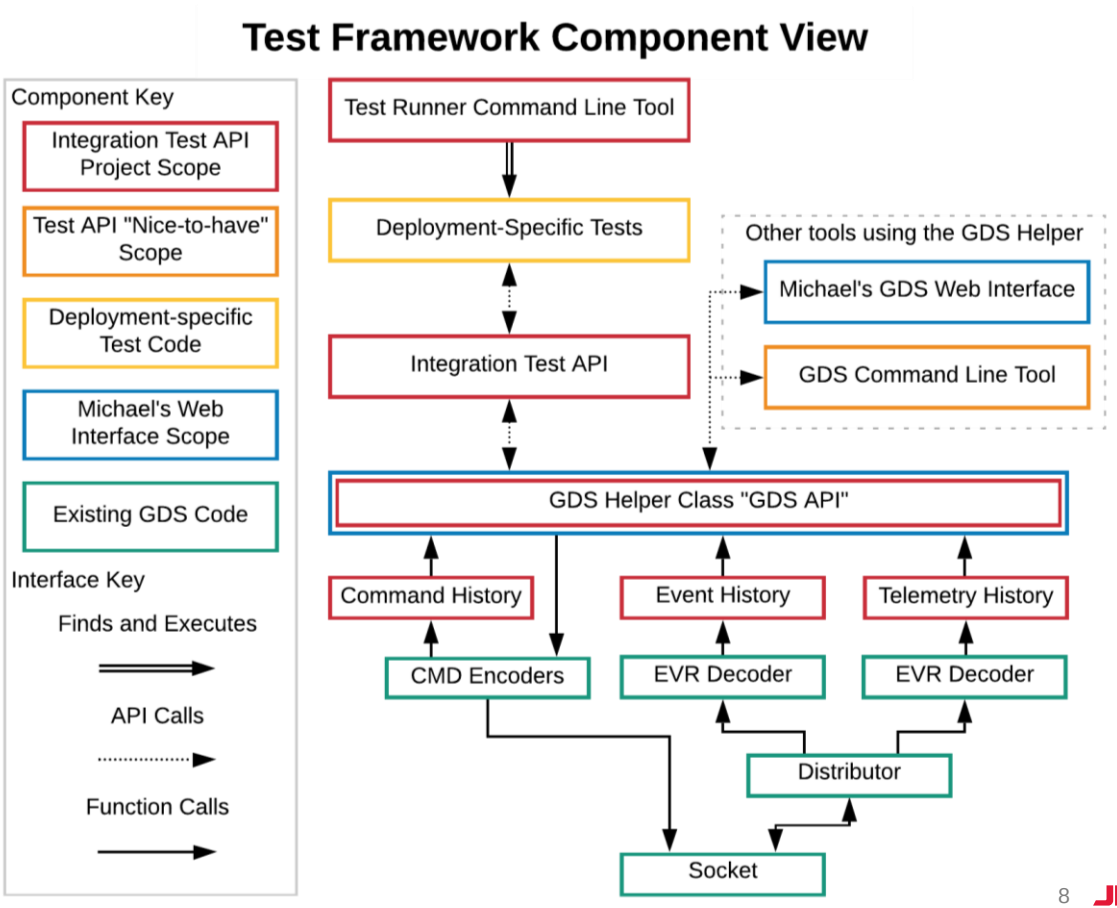
GDS Helper Layer “GDS API”

- The GDS helper layer shall provide the ability to initialize the GDS.
- The GDS helper layer shall provide the ability to Send commands.
- The GDS helper layer shall provide access to FSW Dictionaries.
- The GDS helper layer shall provide and expose histories for EVRs, Telemetry and Commands.

Test API Design

Notes

- Current history implementation needs additional functionality not captured by requirements.
- Standard pipeline may be modified, if tests need more access to GDS.
- The old GSE had a command line tool to poke at an F' Application, this is included as a Nice-to-have.



Test API Design

The following are in a skeleton API on my fork of F’. Username: koran

Currently-defined functions

- History getters
- Latest flight timestamp getter
- Clear Test Histories
- Send and await
- Translate mnemonics to IDs
- Await
- Await sequence

Currently-defined asserts

- Send assert
- Receive assert
- Receive sequence assert
- Count received assert

Test API Design

These are features that have yet to be defined in the API

Functions

- Start test case
- Log test message
- Translate ID to mnemonic

Asserts

- Send and assert

History Searching and Sorting

- How to handle out-of-order events
- What searches to make available
- Where to make these searches available

Project Schedule

“Nice-to-haves”

- Support for rules-based testing.
- Examples of rules-based testing.
- GDS command line interface

F' Integration Test Framework Project Schedule **2019**

Wk	Task	Days	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	Learn how the F' Framework works	1	**June 2**	3 F' Workshop	4 F' Workshop	5 F' Workshop	6	7 RDO	8
2	Initial design work and project planning	5	9	10	11	12	13	14	15
3	Design review and begin coding	3.5	16	17	18	19	20 Half Day	21 RDO	22
4	Implement Integration Test API	5	23	24	25	26	27	28	29
5	Test API Done	2	30	**July 1**	2	3 JPL Holiday	4 JPL Holiday	5 RDO	6
6	Ref app example tests	5	7	8	9	10	11	12	13
7	Example tests done	4	14	15	16	17	18	19 RDO	20
8	Implement Test Runner CLI	5	21	22	23	24	25	26	27
9	Unit tests for the Test API	4	28	29	30	31	**August 1**	2 RDO	3
10	Presentation and final documentation	5	4	5	6	7	8	9 Last Day	10

Jet Propulsion Laboratory
California Institute of Technology