CubeSat Flight Software Workshop

Ground Data System (GDS) Ground Support Equipment (GSE)

Leonard J. Reder
reder@jpl.nasa.gov
5 June 2019

Jet Propulsion Laboratory
California Institute of Technology

Ground Data System (GDS)/Ground Support Equipment (GSE) Fundamental Concept

- ***GDS – "Is an ops sort of thing..."***

RF Command & Telemetry Data Channel

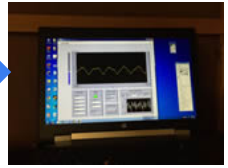
User

GDS

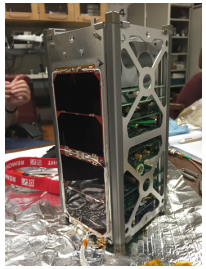
Command & Telemetry Processing/Storage/Visualization

Radio

- ***GSE – "Is a lab sort of thing…"***

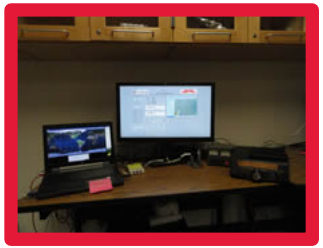


Simulations (Solar/Battery/Load)



CSUNSat1 with JPL developed battery

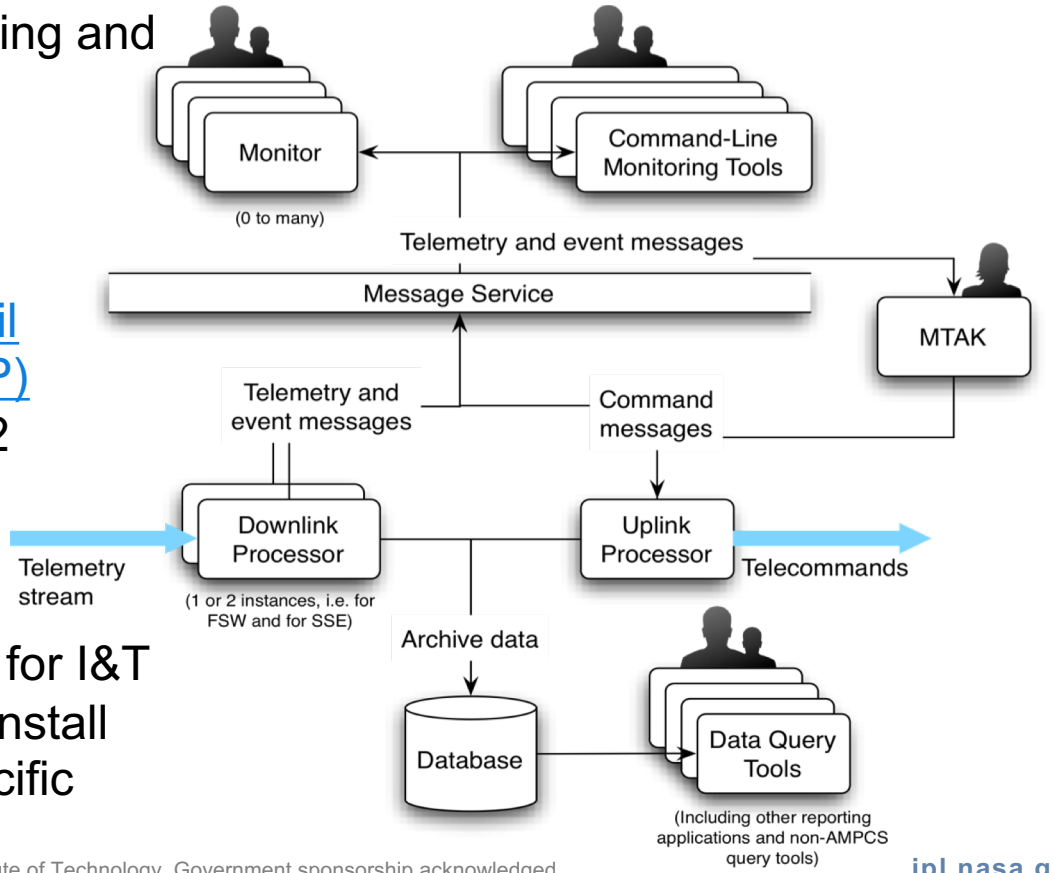
GDS



Command & Telemetry Processing/Storage/Visualization

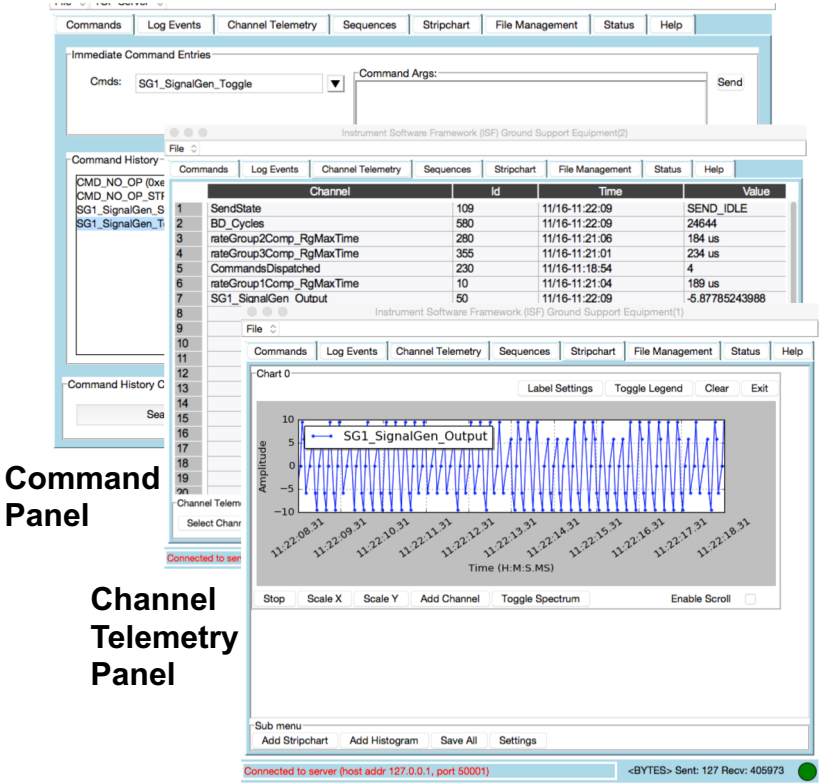
Enterprise Ground Data System (JPL AMPCS)

- AMMOS Mission Data Processing and Control System (AMPCS)
- Architecture Figure from "Cost-Effective Telemetry and Command Ground Systems Automation Strategy for the Soil Moisture Active Passive (SMAP) Mission", Josh Choi, AIAA 2012
- Key thoughts:
  - Based on software bus
  - Uses a full up database
  - MTAK is a Python adapter for I&T
  - Involved to configure and install
    - Requires mission specific adaptations



F’ Ground Support Equipment (GSE) Capability

- ***Out-of-the-box ready to use on Linux, Mac OSX, or Windows without any mission specific tailoring.***
- Enables integration testing and quick-look telemetry monitoring
  - ✓ ***Integration test API and logging***
  - ✓ Immediate commanding
  - ✓ Event and telemetry tables
  - ✓ Sequence assembly and execution
  - ✓ File uplink/downlink
  - ✓ Stripcharts and histograms
- Lightweight portable ground support system that is ***not an Enterprise Class GDS*** such as AMPCS

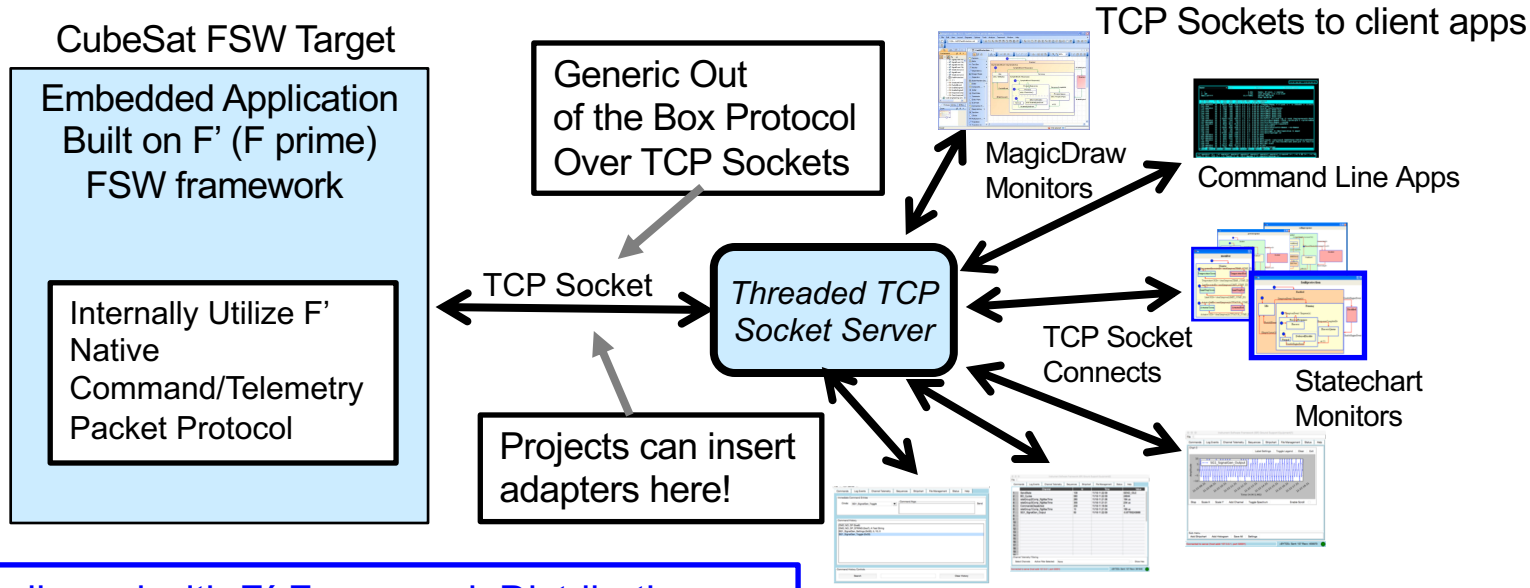


Command Panel

Channel Telemetry Panel

Plotting (Stripcharts, Histograms, etc.) Panel

F’ Ground Support Equipment (GSE) Architecture

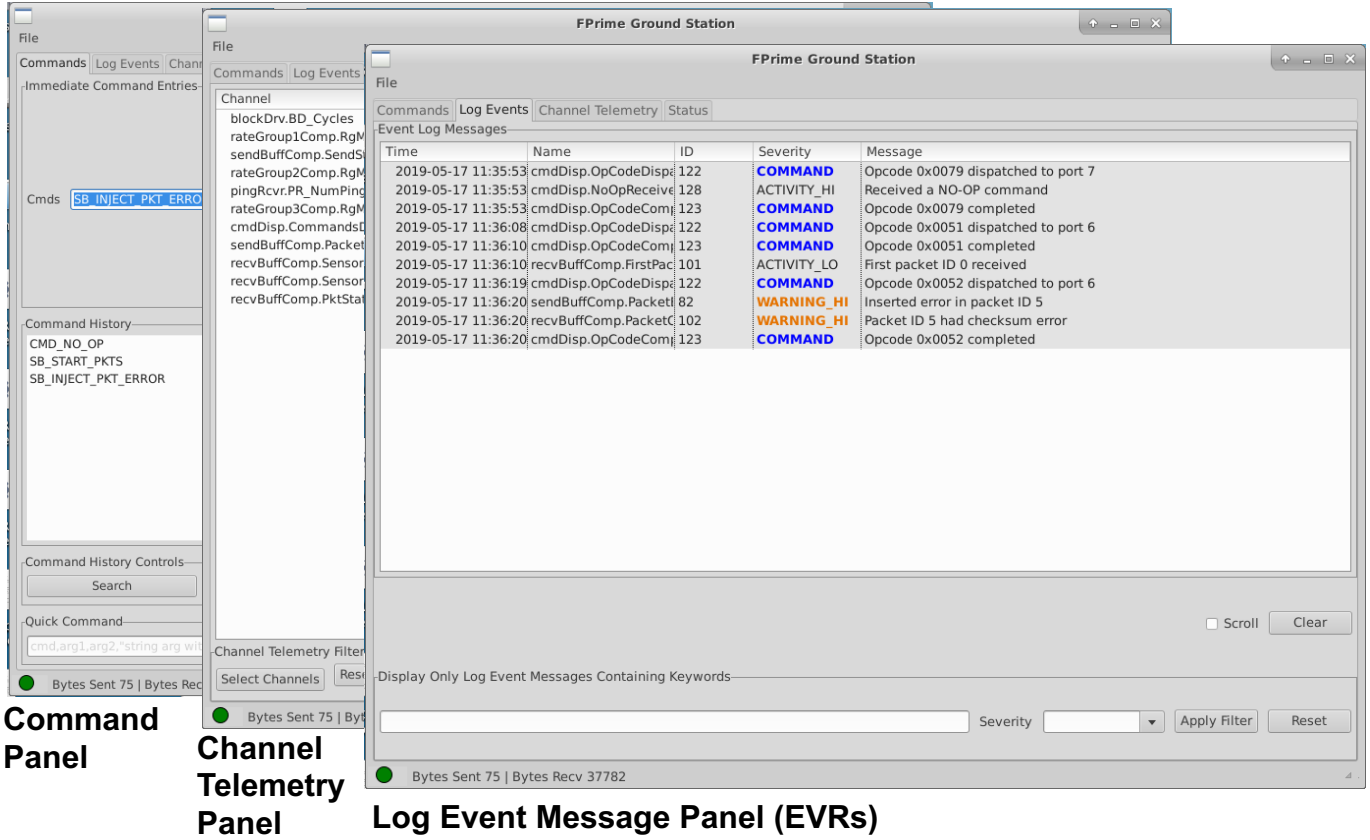


Delivered with F´ Framework Distribution:

- Threaded TCP Socket Server
- gse.py Graphical User Interface
- Integration Test Framework API
- Example command line scripts built on API

New F' Ground Data System (GDS) Client GUI

- Look is identical to legacy GSE Client GUI
- Built on WxPython for improved widgets
- Stripcharting & Histograms to be added
- Replaces GSE Client GUI for Python 3

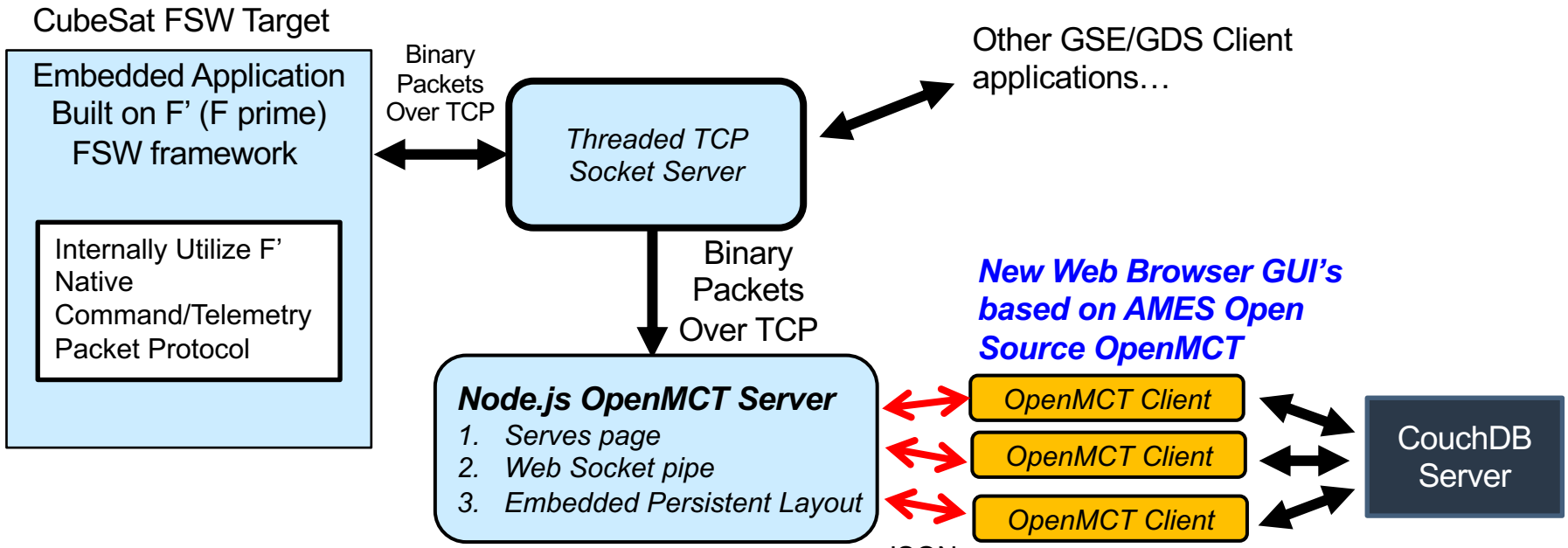


Command Panel

Channel Telemetry Panel

Log Event Message Panel (EVRs)

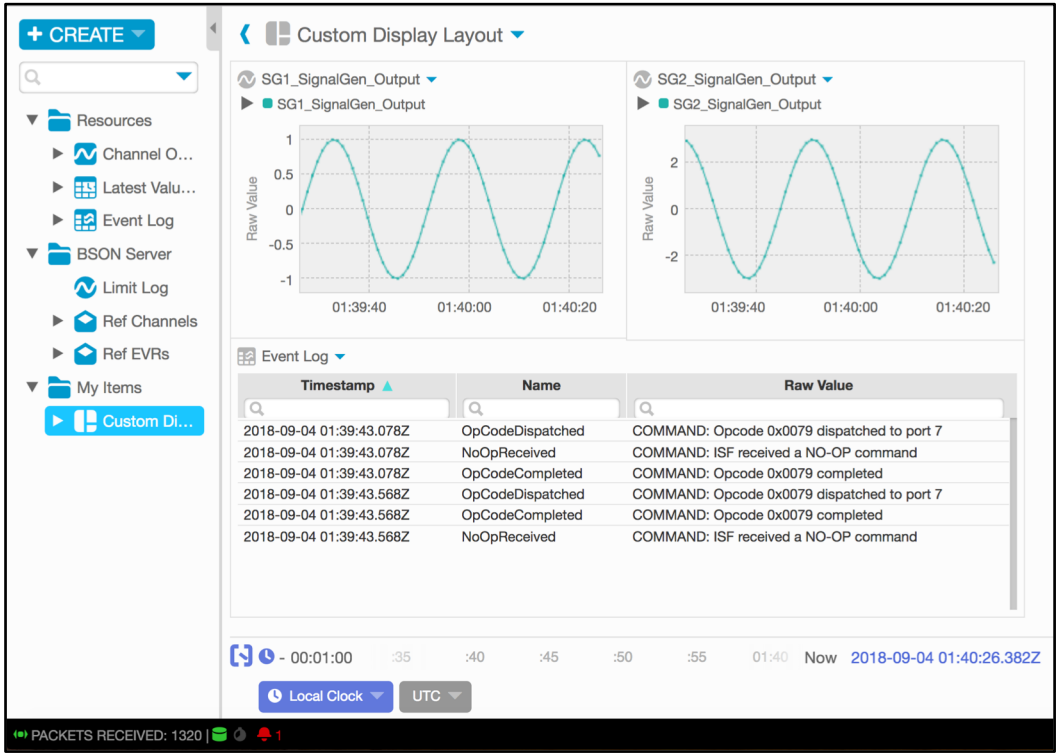
F' OpenMCT Telemetry Monitoring and Browsing Capability (Architecture)



- Node.js server converts F′ binary to BSON packets
- F´ XML used to generate JSON dictionaries

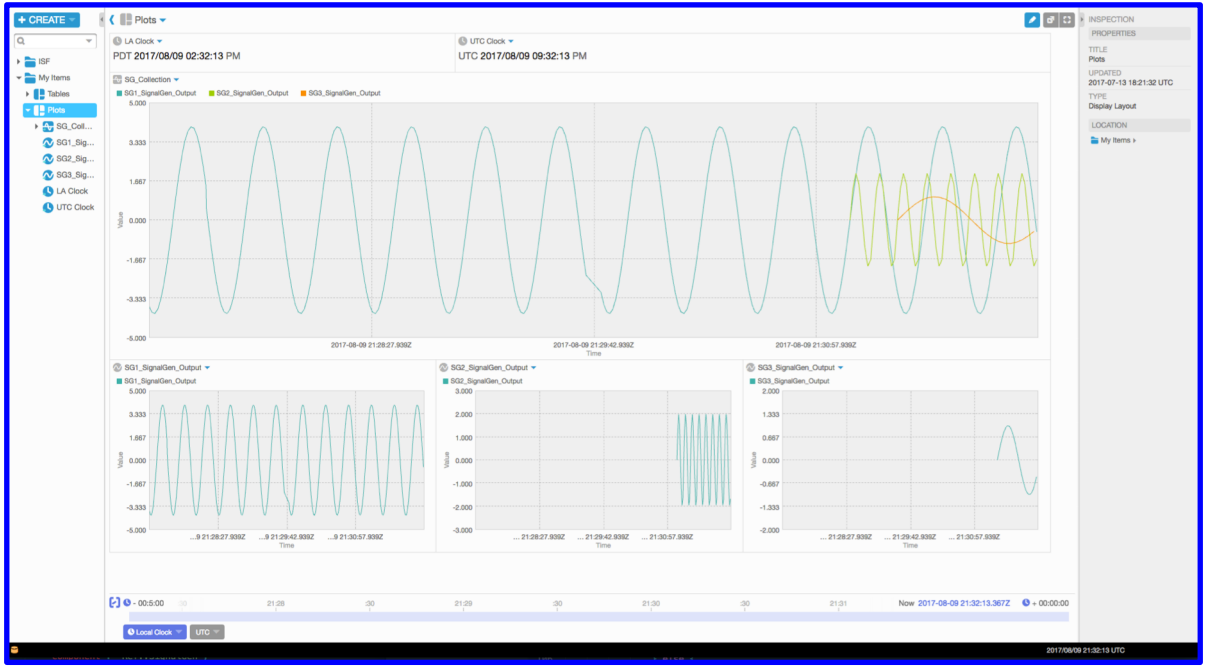
F’ OpenMCT Telemetry Monitoring and Browsing Capability (Capability)

- Open Mission Control Technologies (OpenMCT) open source web framework for visualization of telemetry (see https://github.com/nasa/openmct)
- Node.js Telemetry Server
  - Consumes streaming F´ data
  - Stores telemetry histories using LevelDb database
  - Serves web application content
  - Embedded LevelDb layout persistence
- Client web application
  - Real-time and historical plotting
  - User-customizable layouts
  - Server status indicator in client



F’ OpenMCT displaying F’ telemetry

Multiple instances of stripchart widget shown within AMES open source OpenMCT application connected to F’ demo Ref target app.



F´ Native Protocol & Radio Adaptations (Part 1 Protocol)

- Native message packets assumed within framework
  - Described by descriptors
    - 0 => Commands
    - 1 => Event Log Messages (a. k. a. EVR is Event Report)
    - 2 => Channel Telemetry
    - 3 & 4 => Uplink/Downlink Entities (e.g. RAM Data Product, RAM or DISK Files, etc.)
    - Possible to define others with packets in future
- Often packets (especially simple telemetry) are wrapped with larger packets defined to gain efficiency in transport – Not yet supported in F´
  - Sequences are constructed from commands
- Various protocols used to communicate with radios – None yet supported in F´
  - Consultative Committee for Space Data Systems (CCSDS)
  - AX.25 (Amateur X.25) often used for UHF Ham Radio point-to-point packets

F´ Native Protocol & Radio Adaptations (Part 1 Protocol)

Channel Telemetry Packet

Packet Item	Description	Type/Size
Size	Size of packet starting at end of this buffer	U32
Packet Descriptor	‘1’ Signifying channel tlm	U32
Channel ID	ID of channel	U32
Time Base		U16
Time Context		U8
Seconds	Time tag in seconds	U32
Microseconds	Time tag in microsecs.	U32
Channel Value	Value - use dictionary to parse into specific type.	(Size - 19)

Event Log Message Telemetry Packet

Packet Item	Description	Type/Size
Size	Size of packet starting at end of this buffer	U32
Packet Descriptor	‘2’ Signifying event msg	U32
Channel ID	ID of event	U32
Time Base		U16
Time Context		U8
Seconds	Time tag in seconds	U32
Microseconds	Time tag in microsecs.	U32
Event Args	Value - use dictionary to parse into specific format.	(Size - 19)

Note: Target system prepends an `A5A5 GUI` string when sending for Threaded TCP Server routing.

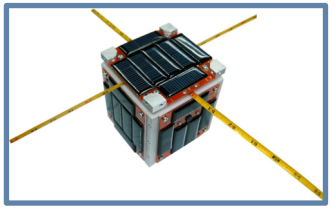
F´ Native Protocol & Radio Adaptations (Part 1 Protocol)

Command Packet

Packet Item	Description	Type/Size
Header	‘A5A5 FSW ZZZZ’	13 bytes
Size	Size of packet starting at end of this buffer	U32
Packet Descriptor	‘0’ Signifying command	U32
Opcode	ID of command	U32
Arguments	Arguments for command.	(Size - 19)

Note: Header is added by client for Threaded TCP Server routing.

F´ Native Protocol & Radio Adaptations (Part 2 Adapters)



Cubsat with tape measure antenna

RF Channel

RADIO

TCP, UDP or ZeroMQ Connections

Physical Hardware Adapters

Protocol Translation Adapter

Threaded TCP Socket Server (Single Target TCP Socket Connection)

Clients

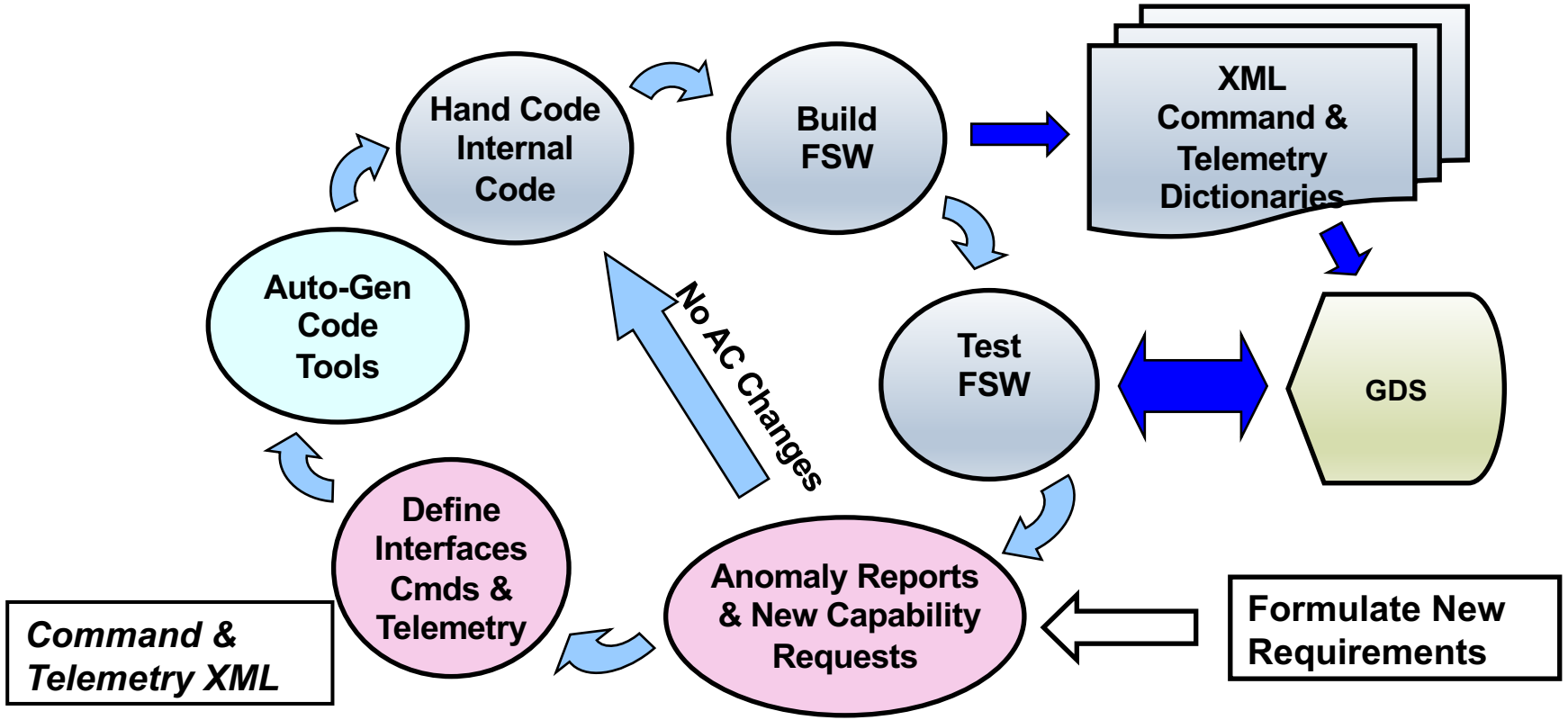
Physical Connection
USB 2.0 / 3.0
RS-422
RJ-11 Ethernet
Others

Projects insert specific adapters to support h/w interfaces and protocol translations (e.g. CCSDS for NASA/DOD, X.25 for University Ham Radio)

Fight / Ground Dictionary Concept

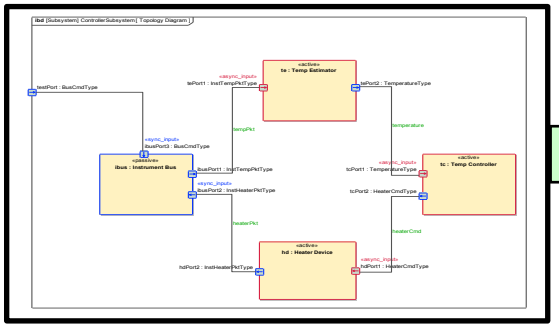
- Ground Dictionaries are the means by which FSW and GDS know how to talk with each other
  - An interface contract between FSW and GDS and visa versa
  - Often represented as flat files that are shared (e.g. XML, JSON, etc.)
  - A database approach such as the web application Dictionary Management System (DMS) (NPO-49751) at JPL is sometimes used as well
- Information shared
  - Commands => opcodes, arguments, mnemonics
  - Event Log Messages => ID's, argument values, formatting, severity
  - Telemetry Channels => ID's, value type, formatting
  - Parameters => ID's, name, type
  - Serializables => Name, data members name/type pairs

Flight / Ground Dictionaries In A Process Context



Extracted from CL#07-2539

F' Code Generation Process (C++ and Dictionaries)



MagicDraw Component Plugin

magicdraw
Architecture Made Simple

Domain Specific XML Files

SerializedData Ai.xml

Port Ai.xml

Component Ai.xml

Topology Ai.xml

F´ Code Generators (Python)

SerializedData Cpp

Port Cpp

Component Cpp

Topology Cpp

Application Flight Code

Command/Telemetry Dictionaries

AMPCS XML Dicts

F´Python Modules Dicts

OpenMCT JSON Dicts

XML Input Dicts:
Command XML
Events XML
Channel XML
Parameter XML

F´ Flight / Ground Dictionary Examples

- F´ utilizes Python Modules, various XML file formats, and JSON

Python module command dictionary entry – CMD_TEST_CMD_1.py

```
COMPONENT = "Svc::CommandDispatcher"

MNEMONIC = "CMD_TEST_CMD_1"

OP_CODE  = 0x7b

CMD_DESCRIPTION = "No-op command"

# Set arguments list with default values here.
ARGUMENTS = [
    ("arg1","The I32 command argument",IsfI32Type()),
    ("arg2","The F32 command argument",IsfF32Type()),
    ("arg3","The U8 command argument",IsfU8Type()),
    ]
```

F´ Flight / Ground Dictionary Examples

- F´ utilizes Python Modules, various XML file formats, and JSON

F´ XML Dictionary representation of CMD_TEST_CMD_1 command

```
<dictionary topology="Ref">
 *****
 <commands>
  *****
  <command component="cmdDisp" mnemonic="CMD_TEST_CMD_1" opcode="0x7b">
   <args>
    <arg name="arg1" type="I32"/>
    <arg name="arg2" type="F32"/>
    <arg name="arg3" type="U8"/>
   </args>
  </command>
  *****
 </commands>
 ****
</dictionary>
```

F´ Flight / Ground Dictionary Examples

- F´ utilizes Python Modules, various XML file formats, and JSON

F´ JSON OpenMCT Dictionary representation of CMD_TEST_CMD_1 command

```
{
   "Ref": {
      ***
      "commands": {
        ****
         "123": {
            "description": "No-op command",
            "component": "Svc::CommandDispatcher",
            "instance": "cmdDisp",
            "arguments": ["I32", "F32", "U8"],
            "id": 123,
            "name": "CMD_TEST_CMD_1"
        },
        ****
     },
```

F´ Flight / Ground Dictionary Examples

- F´ utilizes Python Modules, various XML file formats, and JSON

F´ Python module dictionary entry for log event message TestCmd1Args.py

```
COMPONENT = "Svc::CommandDispatcher"
NAME = "TestCmd1Args"
ID  = 0x7d9
SEVERITY = "COMMAND"
FORMAT_STRING = "ISF TEST_CMD_1 args: I32: %d, F32: %f, U8: %d"
EVENT_DESCRIPTION = "This log event message returns the TEST_CMD_1 arguments."

# Set arguments list with default values here.
ARGUMENTS = [
    ("arg1","Arg1",IsfI32Type()),
    ("arg2","Arg2",IsfF32Type()),
    ("arg3","Arg3",IsfU8Type()),
    ]
```

F´ Flight / Ground Dictionary Examples

- F´ utilizes Python Modules, various XML file formats, and JSON

  F´ XML Dictionary representation of TestCmd1Args event log message

```
<dictionary topology="Ref">
  ***
  <events>
    ***
    <event component="cmdDisp" name="TestCmd1Args" id="0x82" severity="ACTIVITY_HI"
format_string="TEST_CMD_1 args: I32: %d, F32: %f, U8: %d">
      <args>
        <arg name="arg1" type="I32"/>
        <arg name="arg2" type="F32"/>
        <arg name="arg3" type="U8"/>
      </args>
    </event>
    ***
  </events>
  ***
</dictionary>
```

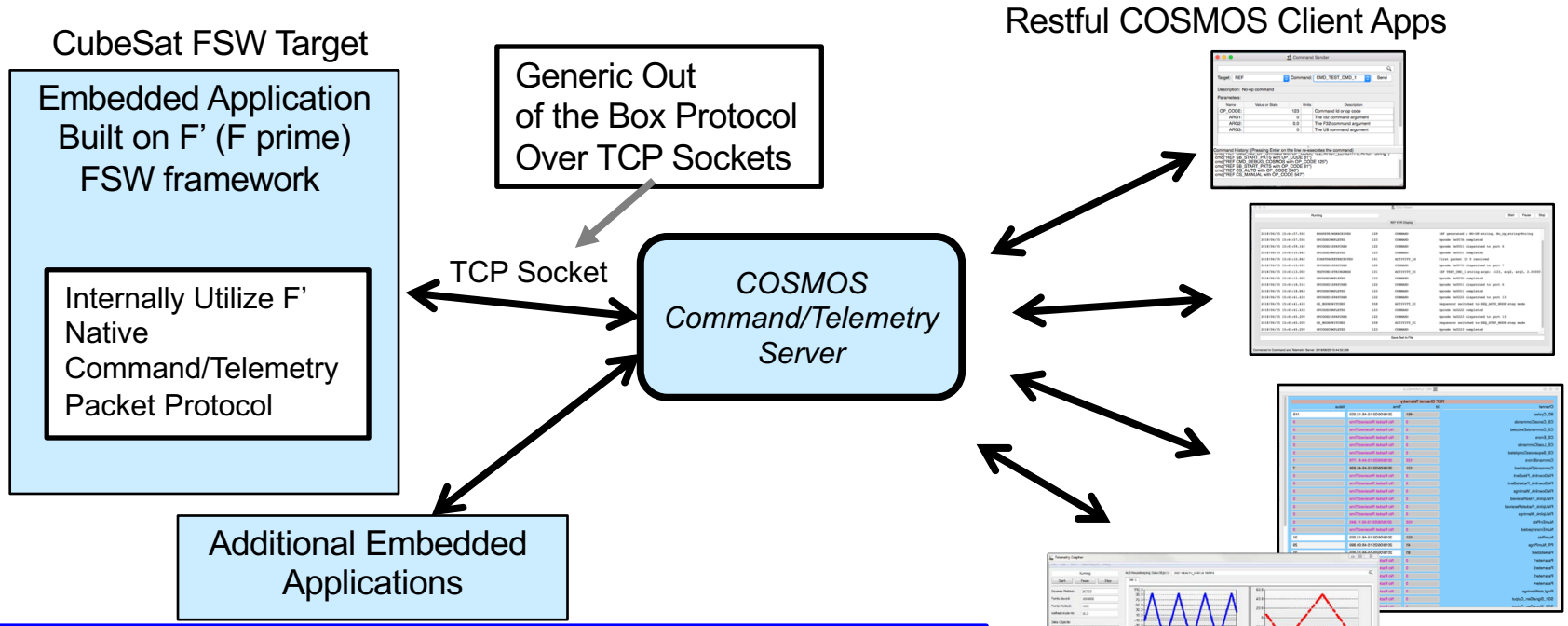
F´ Flight / Ground Dictionary Examples

- F´ utilizes Python Modules, various XML file formats, and JSON

F´ JSON OpenMCT Dictionary representation of TestCmd1Args event log message

```
{
   "Ref": {
      ***
      ”events": {
         ****
 "130": {
            "instance": "cmdDisp",
            "description": "This log event message returns the TEST_CMD_1 arguments.",
            "telem_type": "event",
            "arguments": ["I32", "F32", "U8" ],
            "severity": ”ACTIVITY_HI",
            "format_string": "ISF TEST_CMD_1 args: I32: %d, F32: %f, U8: %d",
            "component": "Svc::CommandDispatcher",
            "id": 130,
            "name": "TestCmd1Args"
         },          ****
```

F´ COSMOS GSE Adaptation & Alternative (Architecture)

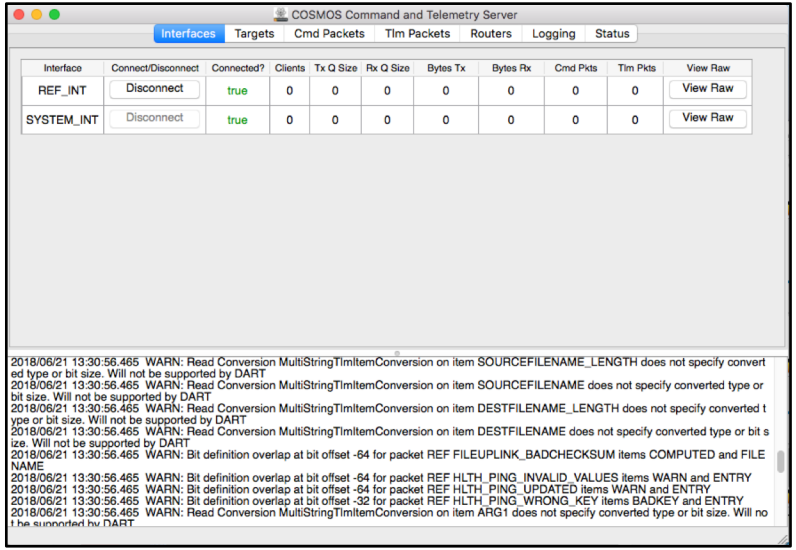


Adapted COSMOS to work with F´ Distribution:

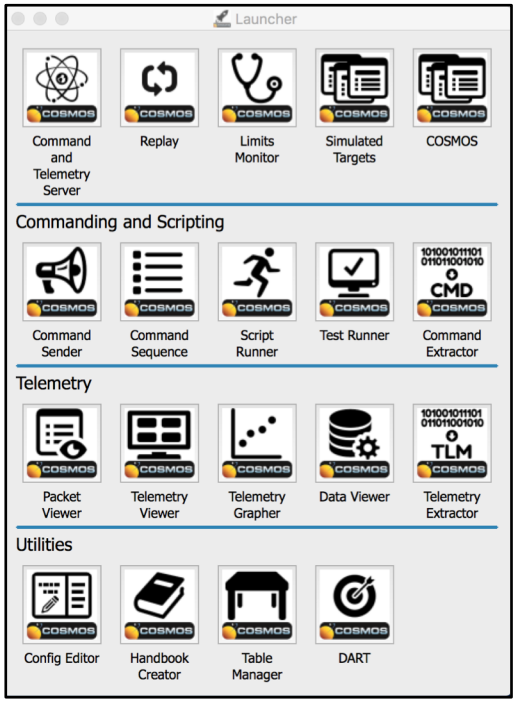
- Tool developed for XML to COSMOS configuration mapping
- Multi-target capable server
- RestFull Http/JSON client interface for building your own

F´ COSMOS GSE Adaptation & Alternative (Capability)

- COSMOS is an open source GSE (a.k.a. GDS) (see https://cosmosrb.com)
- Multi-target capability
- Configurable for both hardware i/f and software protocols
- F´ XML converted to COSMOS text files for configuration
- Over 15 client apps
- Ruby test script API
- Python API
- Telemetry history browsing capability



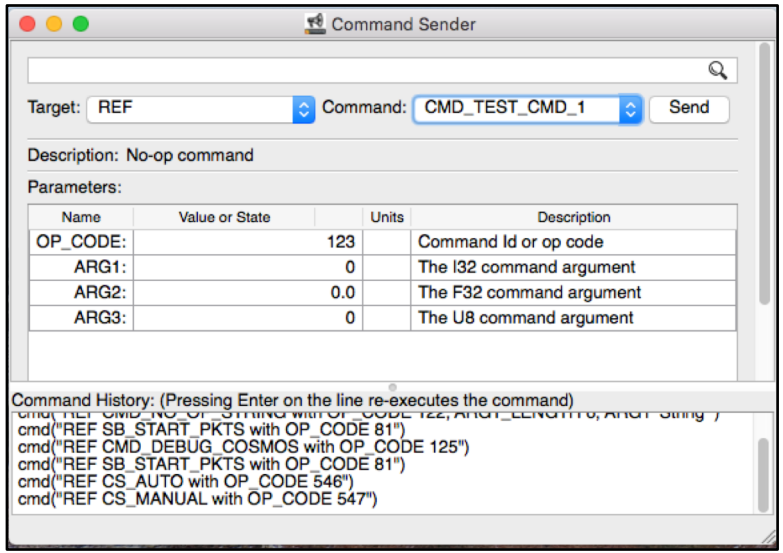
CmdTlmServer Application



Launcher Application

F´ COSMOS GSE Adaptation & Alternative (Capability)

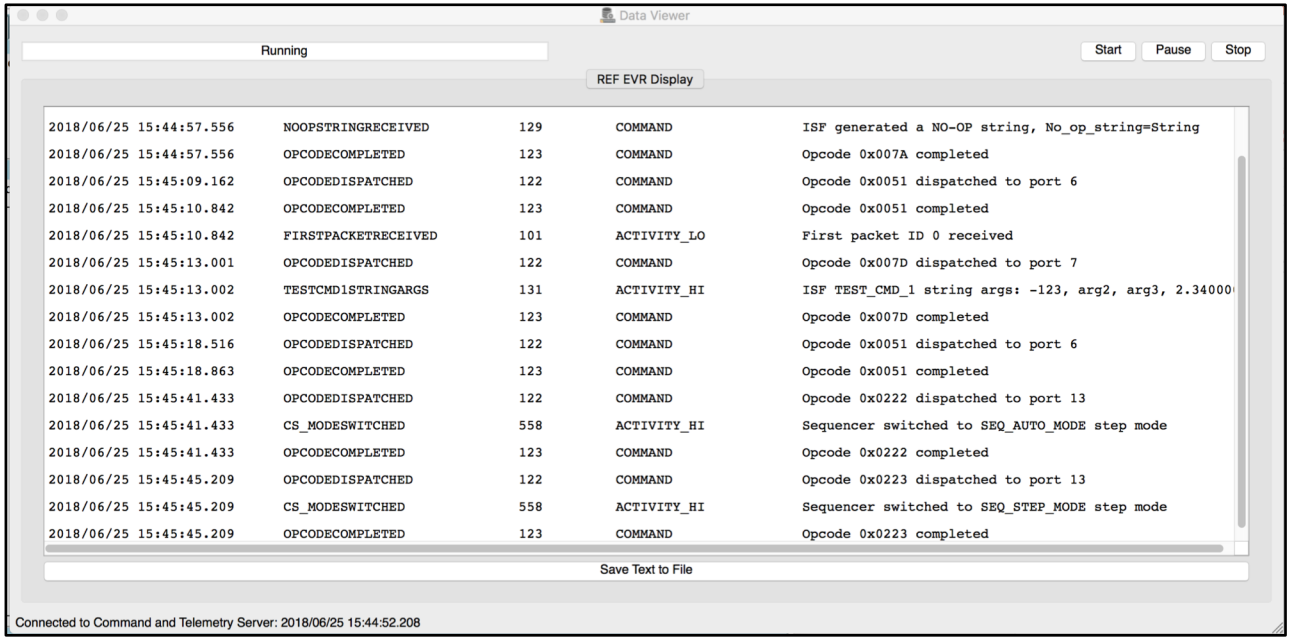
- COSMOS is an open source GSE (a.k.a. GDS) (see https://cosmosrb.com)
- Multi-target capability
- Configurable for both hardware i/f and software protocols
- F´ XML converted to COSMOS text files for configuration
- Over 15 client apps
- Ruby test script API
- Python API
- Telemetry history browsing capability



Command Sender Application

F´ COSMOS GSE Adaptation & Alternative (Capability)

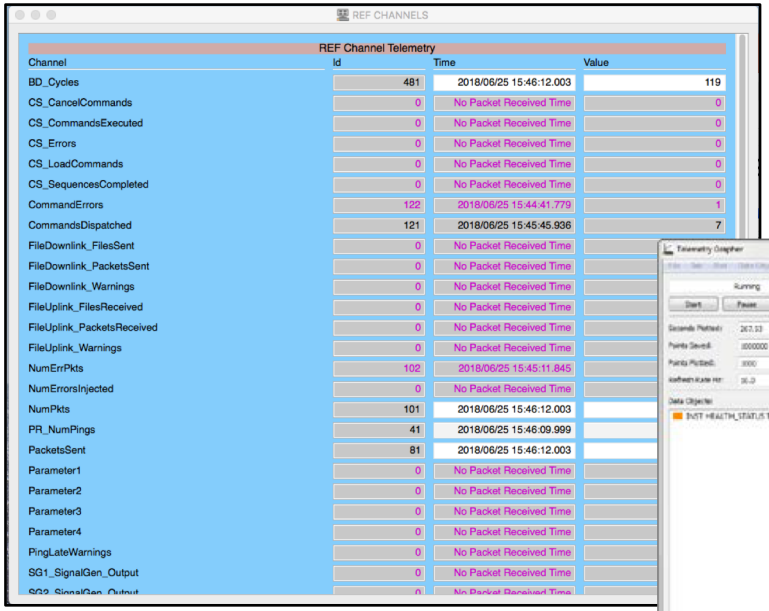
- COSMOS is an open source GSE (a.k.a. GDS) (see https://cosmosrb.com)
- Multi-target capability
- Configurable for both hardware i/f and software protocols
- F´ XML converted to COSMOS text files for configuration
- Over 15 client apps
- Ruby test script API
- Python API
- Telemetry history browsing capability



Data Viewer

F´ COSMOS GSE Adaptation & Alternative (Capability)

- COSMOS is an open source GSE (a.k.a. GDS) (see https://cosmosrb.com)
- Multi-target capability
- Configurable for both hardware i/f and software protocols
- F´ XML converted to COSMOS text files for configuration
- Over 15 client apps
- Ruby test script API
- Python API
- Telemetry history browsing capability



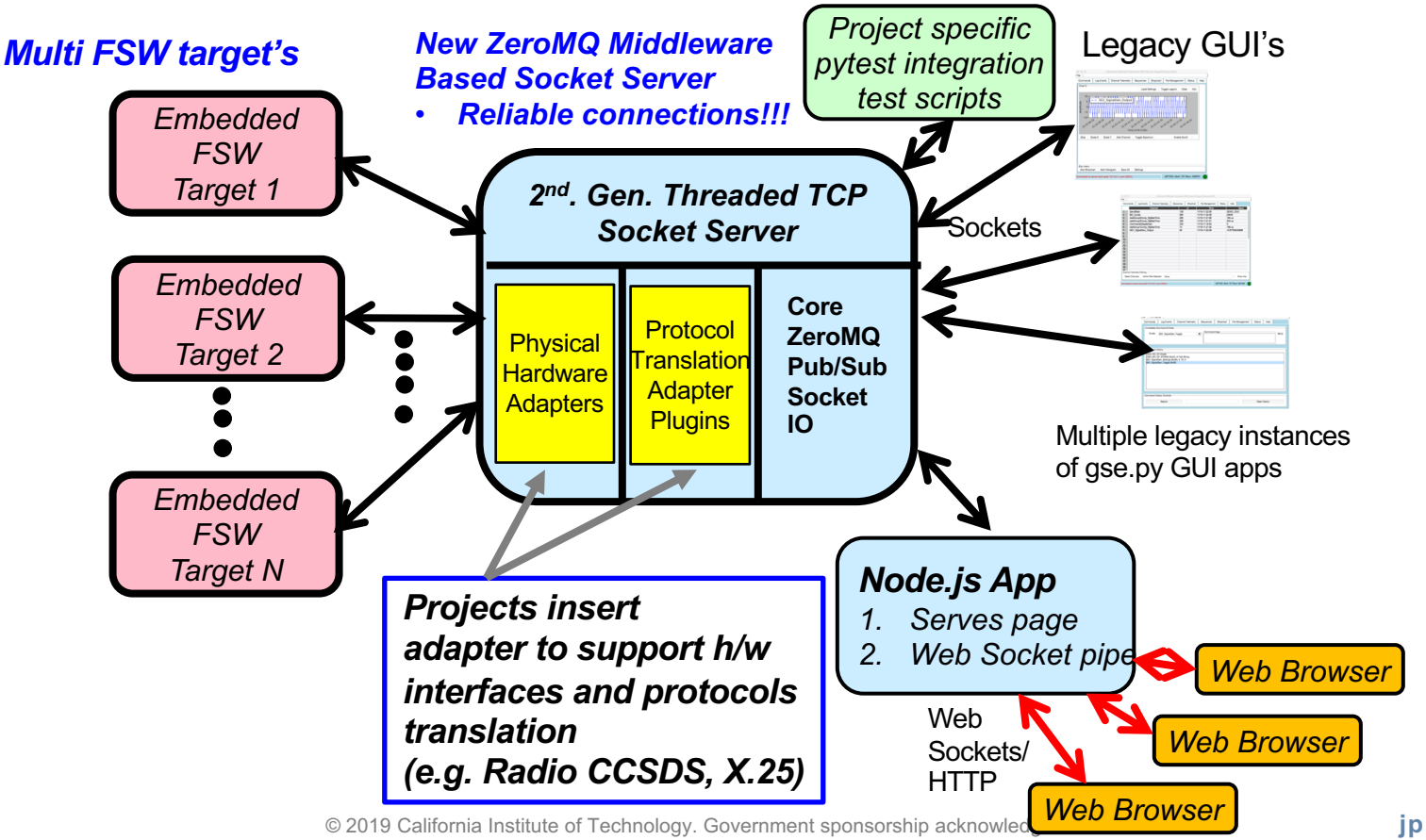
Telemetry Viewer

Telemetry Grapher

F´ GDS Summary

- Intention is to provide F´ user community an out-of-the-box ready to use GDS solution that can run on Linux, Mac OSX, or Windows without any mission specific tailoring
  - User defines mission specific dictionaries (commands & telemetry)
  - Dictionaries are automatically generated from topology model
- GDS provides both an end-user GUI tool and an integration test API
  - Work is going on to improve the capabilities
- Other GDS solutions such as COSMOS can be connected to F´ developed systems
- ***Contribute!!!***
  - *We are interested in your improvements, new GUIs, new client tools, new servers, or just ideas*

Next Generation F' GSE Concept



Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov