Ball Aerospace's COSMOS Open Source Test System

Ryan J. Melton
Ball Aerospace & Technologies Corp.
Boulder, CO

ABSTRACT

Ball Aerospace COSMOS is a free and readily available open source test and operations system. It brings a set of functionality to Integration and Test (I&T) that has previously only been available in proprietary and expensive COTS solutions. A set of 15 applications provide automated procedures, realtime and offline telemetry display and graphing, post-test analysis and CSV extraction, limits monitoring, command and telemetry handbook creation, and binary file editing. Automated test procedures offer the full power of the Ruby programming language allowing operators to send commands, verify telemetry, read and write files, access the network, and even send an email on completion. Additional features include automated test report generation, standardized meta-data collection (unit serial number, operator name), and loop testing (executing the same test repeatedly to wring out timing and other issues). Advanced debugging functionality allows for single-stepping through procedures, setting breakpoints, and complete logging of all script and user interaction with the system under test. Detailed data visualization allows for custom screen creation, line and x-y plotting of realtime data, and easy creation of custom 3d visualizations. Post-test analysis and data extraction capabilities make narrowing down anomalies easy. This paper will discuss all the ways COSMOS can help make I&T and thus the Unit-under-test, better.

KEY WORDS: Open Source, Test Framework, Test Automation, Data Visualization, Graphing, Ball Aerospace COSMOS

INTRODUCTION

Ball Aerospace COSMOS empowers engineers to easily create their own user interface for operating and testing embedded systems. Nine years ago, I was very frustrated spending hours trying to perform what should have been a simple task: create a telemetry display that showed about 50 different values. This frustration resulted in the development of Ball Aerospace COSMOS, over eight years in the making and open sourced in January 2015. COSMOS provides a fully featured test and operations system that provides commanding, automated test scripting, data visualization and much more. This paper discusses the huge amount of functionality available in Ball Aerospace COSMOS and now freely available.

TERMINOLOGY

The COSMOS system uses several terms that are important to understand. Many may be obvious to users within the aerospace industry, but the following table attempts to define these terms clearly for everyone.

Table 1: COSMOS Terminology

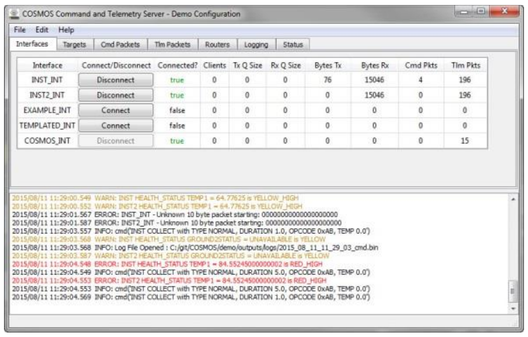
Term	Definition
Target	A COSMOS Target is an embedded system that COSMOS can send commands to and/or receive telemetry from.
Command	A packet of information telling a target to perform an action.
Telemetry Packet	A packet of information providing status from a target. Telemetry packets are either periodically received or may be received in response to a command.
Interface	A Ruby class that knows how to send commands to and/or receive telemetry from a target. COSMOS comes with interfaces that support TCP/IP, UDP, and serial connections. Custom interfaces are easy to add to the system.
Ruby	The powerful dynamic programming language used to write COSMOS applications and libraries. Also the language used in COSMOS scripts and test procedures.
Configuration Files	COSMOS uses simple plain text configuration files to define commands and telemetry packets, and to configure each COSMOS application. These files are easily human readable/editable and machine readable/editable.
Packet Log Files	Binary files containing either logged commands or telemetry packets.
Message Log Files	Text files containing messages generated by a tool.
Tool	Another name for a COSMOS application.

INCLUDED TOOLS

Ball Aerospace COSMOS comes with the following set of 15 applications that are directly available for use with minimal to no configuration.

Command and Telemetry Server

Command and Telemetry Server acts as the hub of the realtime portion of COSMOS. All commands and telemetry packets pass through this tool ensuring everything that happens is logged. It provides realtime commanding, telemetry reception, logging, limits monitoring, packet routing, and system status.

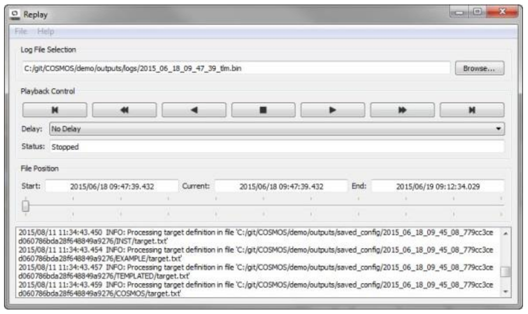

COSMOS Command and Telemetry Server - Demo Configuration
File Edit Help
Interfaces Targets Cmd Packets Tlm Packets Routers Logging Status

Interface	Connect/Disconnect	Connected?	Clients	Tx Q Size	Rx Q Size	Bytes Tx	Bytes Rx	Cmd Pkts	Tlm Pkts
INST_INT	Disconnect	true	0	0	0	76	15046	4	196
INST2_INT	Disconnect	true	0	0	0	0	15046	0	196
EXAMPLE_INT	Connect	false	0	0	0	0	0	0	0
TEMPLATED_INT	Connect	false	0	0	0	0	0	0	0
COSMOS_INT	Disconnect	true	0	0	0	0	0	0	15

```
2015/08/11 11:29:00.549  WARN: INST HEALTH_STATUS TEMP1 = 64.77625 is YELLOW_HIGH
2015/08/11 11:29:00.552  WARN: INST2 HEALTH_STATUS TEMP1 = 64.77625 is YELLOW_HIGH
2015/08/11 11:29:01.567 ERROR: INST_INT - Unknown 10 byte packet starting: 00000000000000000000
2015/08/11 11:29:01.587 ERROR: INST2_INT - Unknown 10 byte packet starting: 00000000000000000000
2015/08/11 11:29:03.557 INFO: cmd('INST COLLECT with TYPE NORMAL, DURATION 1.0, OPCODE 0xAB, TEMP 0.0')
2015/08/11 11:29:03.568  WARN: INST HEALTH_STATUS GROUND2STATUS = UNAVAILABLE is YELLOW
2015/08/11 11:29:03.568 INFO: Log File Opened : C:/git/COSMOS/demo/outputs/logs/2015_08_11_11_29_03_cmd.bin
2015/08/11 11:29:03.587  WARN: INST2 HEALTH_STATUS GROUND2STATUS = UNAVAILABLE is YELLOW
2015/08/11 11:29:04.548 ERROR: INST HEALTH_STATUS TEMP1 = 84.55245000000002 is RED_HIGH
2015/08/11 11:29:04.549 INFO: cmd('INST COLLECT with TYPE NORMAL, DURATION 5.0, OPCODE 0xAB, TEMP 0.0')
2015/08/11 11:29:04.553 ERROR: INST2 HEALTH_STATUS TEMP1 = 84.55245000000002 is RED_HIGH
2015/08/11 11:29:04.553 INFO: cmd('INST COLLECT with TYPE NORMAL, DURATION 5.0, OPCODE 0xAB, TEMP 0.0')
2015/08/11 11:29:04.569 INFO: cmd('INST COLLECT with TYPE NORMAL, DURATION 1.0, OPCODE 0xAB, TEMP 0.0')
```

Replay

Replay simulates the Command and Telemetry Server for telemetry packet log file playback. This enables use of any of the realtime tools with logged data. Replay is great for playing back scenarios and viewing them on telemetry screens.



Log File Selection
C:/git/COSMOS/demo/outputs/logs/2015_06_18_09_47_39_tlm.bin

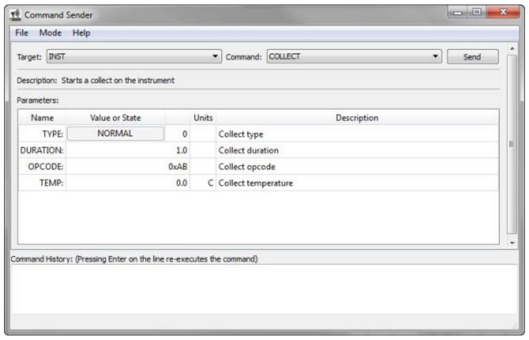
Playback Control
Delay: No Delay
Status: Stopped

File Position
Start: 2015/06/18 09:47:39.432 Current: 2015/06/18 09:47:39.432 End: 2015/06/19 09:12:34.029

```
2015/08/11 11:34:43.450  INFO: Processing target definition in file 'C:/git/COSMOS/demo/outputs/saved_config/2015_06_18_09_45_08_779cc3ce
d060786bda28f648849a9276/INST/target.txt'
2015/08/11 11:34:43.454  INFO: Processing target definition in file 'C:/git/COSMOS/demo/outputs/saved_config/2015_06_18_09_45_08_779cc3ce
d060786bda28f648849a9276/EXAMPLE/target.txt'
2015/08/11 11:34:43.457  INFO: Processing target definition in file 'C:/git/COSMOS/demo/outputs/saved_config/2015_06_18_09_45_08_779cc3ce
d060786bda28f648849a9276/TEMPLATED/target.txt'
2015/08/11 11:34:43.459  INFO: Processing target definition in file 'C:/git/COSMOS/demo/outputs/saved_config/2015_06_18_09_45_08_779cc3ce
d060786bda28f648849a9276/COSMOS/target.txt'
```

Command Sender

Command Sender provides a graphical interface for manually sending individual commands. Drop down selection of every command and command parameter in the system makes sending individual commands easy. A history pane makes resending previous commands easy.



Target: INST Command: COLLECT

Description: Starts a collect on the instrument

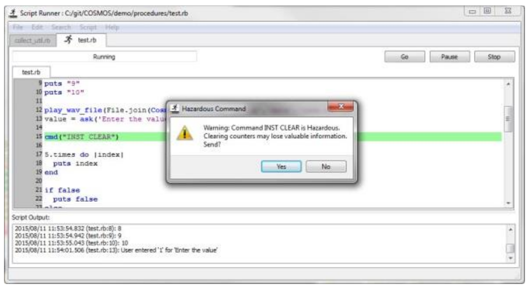
Parameters:

Name	Value or State		Units	Description
TYPE:	NORMAL	0		Collect type
DURATION:		1.0		Collect duration
OPCODE:		0xAB		Collect opcode
TEMP:		0.0	C	Collect temperature

Command History: (Pressing Enter on the line re-executes the command)

Script Runner

Script Runner executes test scripts and provides highlighting of the currently executing line. Scripts pause if any error occurs, breakpoints can be added, and lines can be reexecuted after a problem has been corrected.



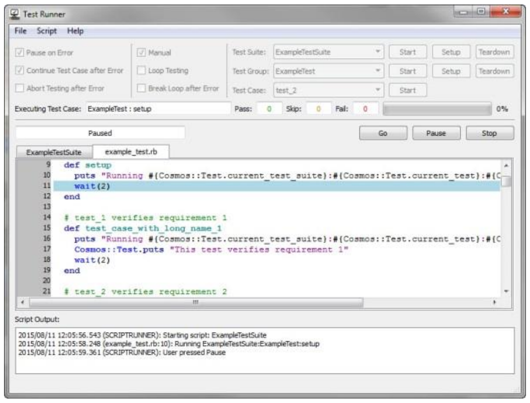
```
puts "9"
puts "10"

play_wav_file(File.join(Cos
value = ask('Enter the valu

cmd("INST CLEAR")

5.times do |index|
  puts index
end

if false
  puts false
```

Warning: Command INST CLEAR is Hazardous. Clearing counters may lose valuable information. Send?

```
2015/08/11 11:53:54.832 (test.rb:8): 8
2015/08/11 11:53:54.942 (test.rb:9): 9
2015/08/11 11:53:55.043 (test.rb:10): 10
2015/08/11 11:54:01.506 (test.rb:13): User entered '1' for 'Enter the value'
```

Test Runner

Test Runner provides a high level framework for system level testing including automatic test report generation. Test Runner brings the best features of software unit level testing to system level integration and test by breaking tests down into easy understandable test cases. Users can execute entire test procedures or just the specific test cases they need to run for integration or regression tests.



Test Suite: ExampleTestSuite
Test Group: ExampleTest
Test Case: test_2
Executing Test Case: ExampleTest : setup Pass: 0 Skip: 0 Fail: 0 0%
Paused

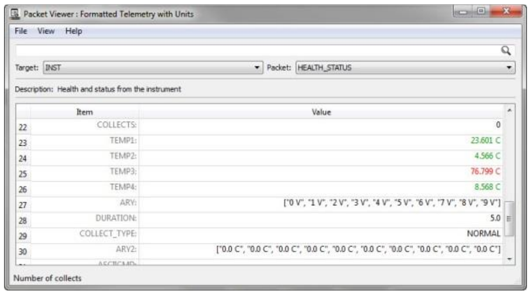
```
def setup
  puts "Running #{Cosmos::Test.current_test_suite}:#{Cosmos::Test.current_test}:#{C
  wait(2)
end

# test_1 verifies requirement 1
def test_case_with_long_name_1
  puts "Running #{Cosmos::Test.current_test_suite}:#{Cosmos::Test.current_test}:#{C
  Cosmos::Test.puts "This test verifies requirement 1"
  wait(2)
end

# test_2 verifies requirement 2
```

Script Output:

```
2015/08/11 12:05:56.543 (SCRIPTRUNNER): Starting script: ExampleTestSuite
2015/08/11 12:05:58.248 (example_test.rb:10): Running ExampleTestSuite:ExampleTest:setup
2015/08/11 12:05:59.361 (SCRIPTRUNNER): User pressed Pause
```

Packet Viewer

Packet Viewer provides realtime visualization of every telemetry packet that has been defined. Values within packets are displayed in a simple key-value format that requires no configuration. An autocomplete search bar makes finding values easy.



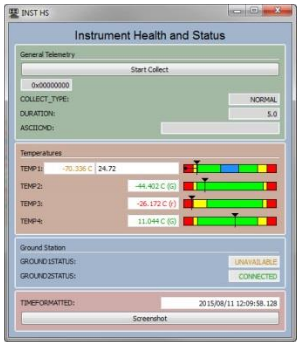
Target: INST Packet: HEALTH_STATUS
Description: Health and status from the instrument

	Item	Value
22	COLLECTS:	0
23	TEMP1:	23.601 C
24	TEMP2:	4.566 C
25	TEMP3:	76.799 C
26	TEMP4:	8.568 C
27	ARY:	["0 V", "1 V", "2 V", "3 V", "4 V", "5 V", "6 V", "7 V", "8 V", "9 V"]
28	DURATION:	5.0
29	COLLECT_TYPE:	NORMAL
30	ARY2:	["0.0 C", "0.0 C", "0.0 C", "0.0 C", "0.0 C", "0.0 C", "0.0 C", "0.0 C", "0.0 C", "0.0 C"]

Number of collects

Telemetry Viewer

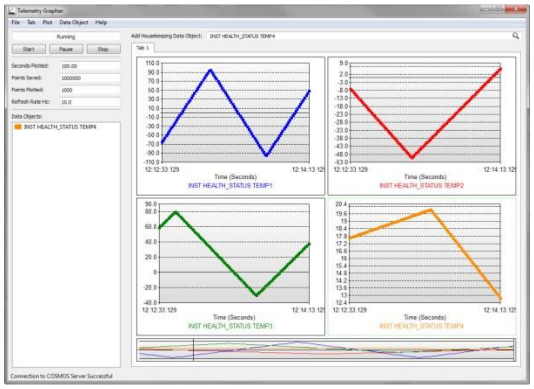
Telemetry Viewer provides custom telemetry screen functionality with advanced layout and visualization widgets. Tabs, graphs, limits bars, and other animated displays can be quickly created. Also, Telemetry Viewer can autogenerate a base set of screens for every telemetry packet that can be customized as needed.



0x00000000
COLLECT_TYPE: NORMAL
DURATION: 5.0
ASCIICMD:
TEMP1: -70.336 C 24.72
TEMP2: -44.402 C (G)
TEMP3: -26.172 C (r)
TEMP4: 11.044 C (G)
GROUND1STATUS: UNAVAILABLE
GROUND2STATUS: CONNECTED
TIMEFORMATTED: 2015/08/11 12:09:58.128

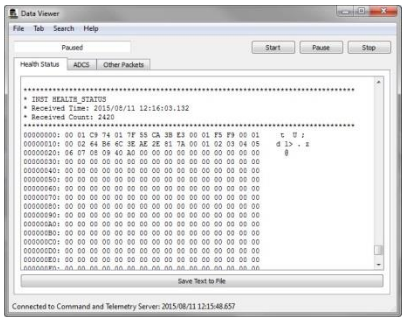
Telemetry Grapher

Telemetry Grapher provides realtime and offline graphing of telemetry data. Supports both line and x-y style plotting, with multiple tabs, plots, and items per plot. Includes built-in analysis functionality to graph min, max, difference, and standard deviation.



Data Viewer

Data Viewer provides text based telemetry visualization for items that don’t fit into other data visualization paradigms. Great for scrolling log displays and memory dumps.

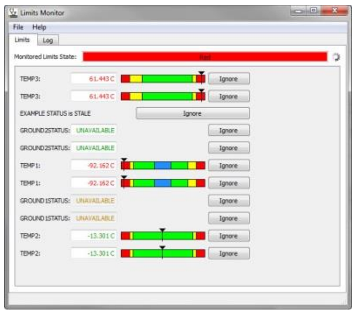


```
******************************************************************************
* INST HEALTH_STATUS
* Received Time: 2015/08/11 12:16:03.132
* Received Count: 2420
******************************************************************************
00000000: 00 01 C9 74 01 7F 55 CA 3B E3 00 01 F5 F9 00 01     t  U ;
00000010: 00 02 64 B6 6C 3E AE 2E 81 7A 00 01 02 03 04 05    d l> . z
00000020: 06 07 08 09 40 A0 00 00 00 00 00 00 00 00 00 00      @
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Connected to Command and Telemetry Server: 2015/08/11 12:15:48.657

Limits Monitor

Limits Monitor monitors telemetry with defined limits and shows items that are currently out of limits or have violated limits since the tool was started. Expected violations can be easily ignored.

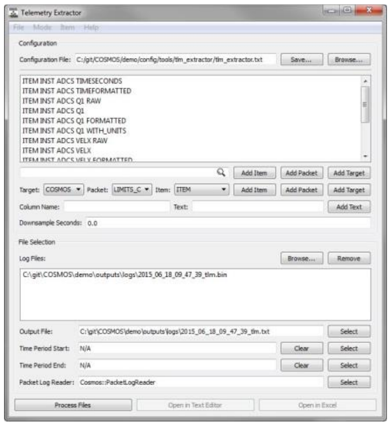


Monitored Limits State: Red

TEMP3:	61.443 C	Ignore
TEMP3:	61.443 C	Ignore
EXAMPLE STATUS is STALE		Ignore
GROUND2STATUS:	UNAVAILABLE	Ignore
GROUND2STATUS:	UNAVAILABLE	Ignore
TEMP1:	-92.162 C	Ignore
TEMP1:	-92.162 C	Ignore
GROUND1STATUS:	UNAVAILABLE	Ignore
GROUND1STATUS:	UNAVAILABLE	Ignore
TEMP2:	-13.301 C	Ignore
TEMP2:	-13.301 C	Ignore

Telemetry Extractor

Telemetry Extractor extracts telemetry packet log files into CSV data. Highly configurable and supports batch processing to output multiple files at once.



Configuration File: C:/git/COSMOS/demo/config/tools/tlm_extractor/tlm_extractor.txt

```
ITEM INST ADCS TIMESECONDS
ITEM INST ADCS TIMEFORMATTED
ITEM INST ADCS Q1 RAW
ITEM INST ADCS Q1
ITEM INST ADCS Q1 FORMATTED
ITEM INST ADCS Q1 WITH_UNITS
ITEM INST ADCS VELX RAW
ITEM INST ADCS VELX
ITEM INST ADCS VELX FORMATTED
```

Target: COSMOS Packet: LIMITS_C Item: ITEM

Column Name: Text:

Downsample Seconds: 0.0

Log Files: C:\git\COSMOS\demo\outputs\logs\2015_06_18_09_47_39_tlm.bin

Output File: C:\git\COSMOS\demo\outputs\logs\2015_06_18_09_47_39_tlm.txt

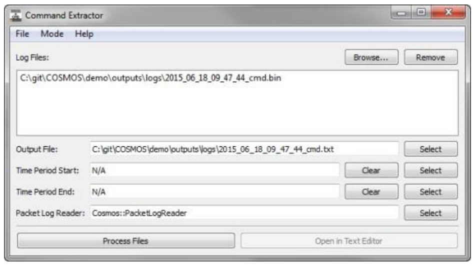
Time Period Start: N/A

Time Period End: N/A

Packet Log Reader: Cosmos::PacketLogReader

Command Extractor

Command Extractor extracts command packet logs into human readable text.



Log Files: C:\git\COSMOS\demo\outputs\logs\2015_06_18_09_47_44_cmd.bin

Output File: C:\git\COSMOS\demo\outputs\logs\2015_06_18_09_47_44_cmd.txt

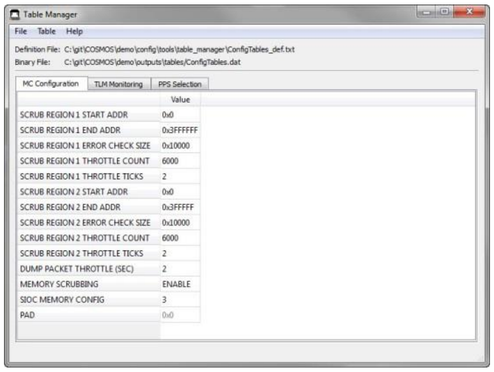
Time Period Start: N/A

Time Period End: N/A

Packet Log Reader: Cosmos::PacketLogReader

Table Manager

Table Manager is a binary file editor that can be used to create or edit configuration tables or other binary data.

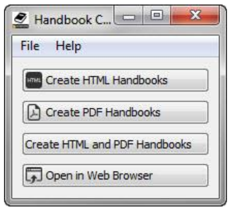


Definition File: C:\git\COSMOS\demo\config\tools\table_manager\ConfigTables_def.txt
Binary File: C:\git\COSMOS\demo\outputs\tables/ConfigTables.dat

	Value
SCRUB REGION 1 START ADDR	0x0
SCRUB REGION 1 END ADDR	0x3FFFFFF
SCRUB REGION 1 ERROR CHECK SIZE	0x10000
SCRUB REGION 1 THROTTLE COUNT	6000
SCRUB REGION 1 THROTTLE TICKS	2
SCRUB REGION 2 START ADDR	0x0
SCRUB REGION 2 END ADDR	0x3FFFFF
SCRUB REGION 2 ERROR CHECK SIZE	0x10000
SCRUB REGION 2 THROTTLE COUNT	6000
SCRUB REGION 2 THROTTLE TICKS	2
DUMP PACKET THROTTLE (SEC)	2
MEMORY SCRUBBING	ENABLE
SIOC MEMORY CONFIG	3
PAD	0x0

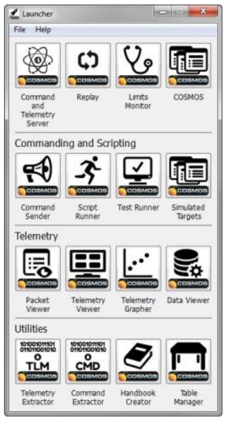
Handbook Creator

Handbook Creator creates html and pdf documentation of available commands and telemetry packets.



Launcher

Launcher provides a graphical user interface for launching each of the tools that make up the COSMOS system. Supports launching any application that can be started from the command line.



SYSTEM ARCHITECTURE

The following diagram shows how the 15 applications that make up the COSMOS system relate to each other and to the targets that COSMOS is controlling.

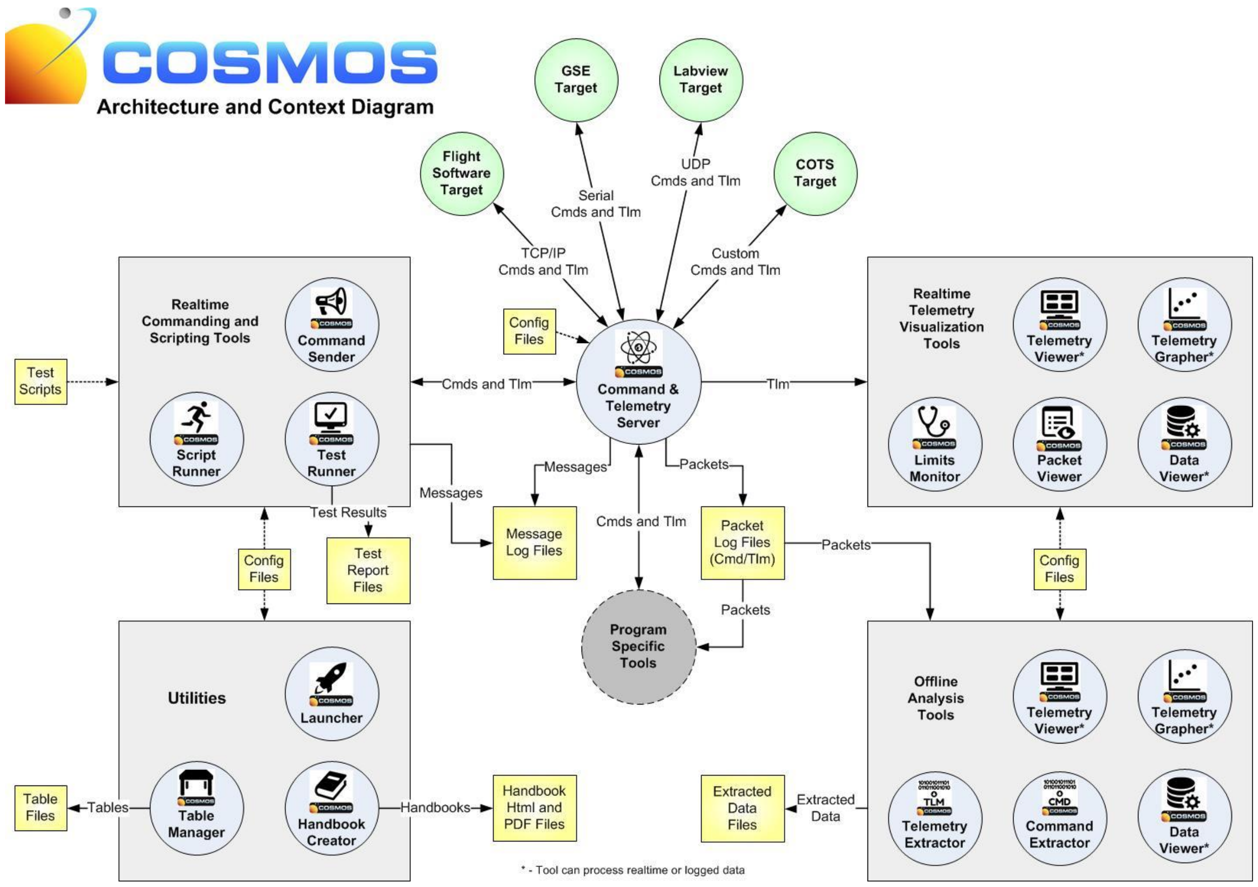


Figure 1: COSMOS Architecture and Context Diagram

Key aspects of this architecture:

1. The COSMOS tools are grouped into four broad categories
    a. Realtime Command and Scripting
    b. Realtime Telemetry Visualization
    c. Offline Analysis
    d. Utilities

2. COSMOS can interface with many different kinds of targets. The examples shown in this diagram include Flight Software (FSW), Ground Support Equipment (GSE), Labview, and a Commercial Off-The-Shelf (COTS) target such as an Agilent Power Supply. Any embedded system that provides a communication interface can be connected to COSMOS.

3. COSMOS ships with interfaces for connecting over TCP/IP, UDP, and serial

connections. COSMOS also supports custom interfaces to connect to anything that a computer can talk to.

4. All realtime communication with targets flows through the Command and Telemetry Server. This ensures all commands and telemetry are logged.

5. Every tool is configured with plain text configuration files (if any configuration is needed).

6. Project specific tools can be written using the COSMOS libraries that can interact with the realtime command and telemetry streams through the Command and Telemetry Server and can also do offline analysis of packet log files.

7. Cross Platform – COSMOS supports Windows, Linux, and Mac OSX.

KEY BENEFITS TO INTEGRATION AND TEST

Full Lifecycle System - Supports board level test, box level test, I&T, and operations providing a consistent user interface throughout the full lifecycle of a product.

Everything is logged – And even more importantly, tools are provided to easily interpret and use the logs. Whenever an anomaly occurs there are tools already written that are ready to dig into the logs and help figure out what happened.

Superb Data Visualization – Anyone can create great telemetry displays, graph data in realtime, and provide an excellent sense of situational awareness – all without any programming required.

Powerful Test Scripting – COSMOS comes with a simple API that makes sending commands and checking telemetry easy. However, you are not constrained by your test scripting language. COSMOS scripts are written in Ruby, a modern, fully functional scripting language. This allows you to read and write files, and perform live processing that most other systems force you to run offline.

Powerful Test Reporting and Organization – COSMOS Test Runner can produce very reliable test procedures that allow the user to easily execute the entire procedure or only a subset needed for a regression test. Automated test reports created at the end of every run make it very clear that everything passed successfully or where problems occurred.

SUMMARY

Ball Aerospace COSMOS is a free and open source test and operations system that is immediately available for use. It provides a wealth of functionality much of which is not even available in expensive proprietary tools. For more information and to get started with Ball Aerospace COSMOS please see http://cosmosrb.com.

REFERENCES

Melton, Ryan, "Ball Aerospace COSMOS" *Retrieved from http://cosmosrb.com August 9, 2015*

BIOGRAPHIES

Ryan Melton has been working at Ball Aerospace in Boulder Colorado and helping to develop, integrate, and test aerospace products for the past 14 years. Notable programs on which Ryan has worked include Kepler, GPM, and CALIPSO. Ryan is very active in the open source community with the recent release of Ball Aerospace COSMOS as well as for many years working with the Ruby bindings to the Qt GUI framework, qtbindings. He holds a Bachelor's of Science in Computer Engineering from Purdue University and an MBA from Regis University. Please address any questions on this paper to rmelton@ball.com.