

**NASA/TM–20210000619/Rev 1**



**Core Flight System (cFS) Training**

**Integration with COSMOS**

*Flight Software Systems Branch, Code 582  
Goddard Space Flight Center, Greenbelt, MD*

---

**March 2021**

## NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question to [help@sti.nasa.gov](mailto:help@sti.nasa.gov)
- Phone the NASA STI Information Desk at 757-864-9658

Write to:

NASA STI Information Desk Mail, Stop 148  
NASA Langley Research Center  
Hampton, VA 23681-2199

**NASA/TM–20210000619/Rev 1**



**Core Flight System (cFS) Training**

**Integration with COSMOS**

*Flight Software Systems Branch, Code 582  
Goddard Space Flight Center, Greenbelt, MD*

---

**March 2021**

### Notice for Copyrighted Information

This manuscript is a work of the United States Government authored as part of the official duties of employee(s) of the National Aeronautics and Space Administration. No copyright is claimed by the United States under Title 17, U.S. Code. All other rights are reserved by the United States Government. Any publisher accepting this manuscript for publication acknowledges that the United States Government retains a non-exclusive, irrevocable, worldwide license to prepare derivative works, publish, or reproduce this manuscript, or allow others to do so, for United States Government purposes.

Trade names and trademarks are used in this report for identification only. Their usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

**Level of Review:** This material has been technically reviewed by technical management.

Available from

NASA STI Program  
Mail Stop 148  
NASA's Langley Research Center  
Hampton, VA 23681-2199

National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161  
703-605-6000



National Aeronautics and Space Administration



# Core Flight Executive (cFS) Training

## Integration with COSMOS



# Course Agenda

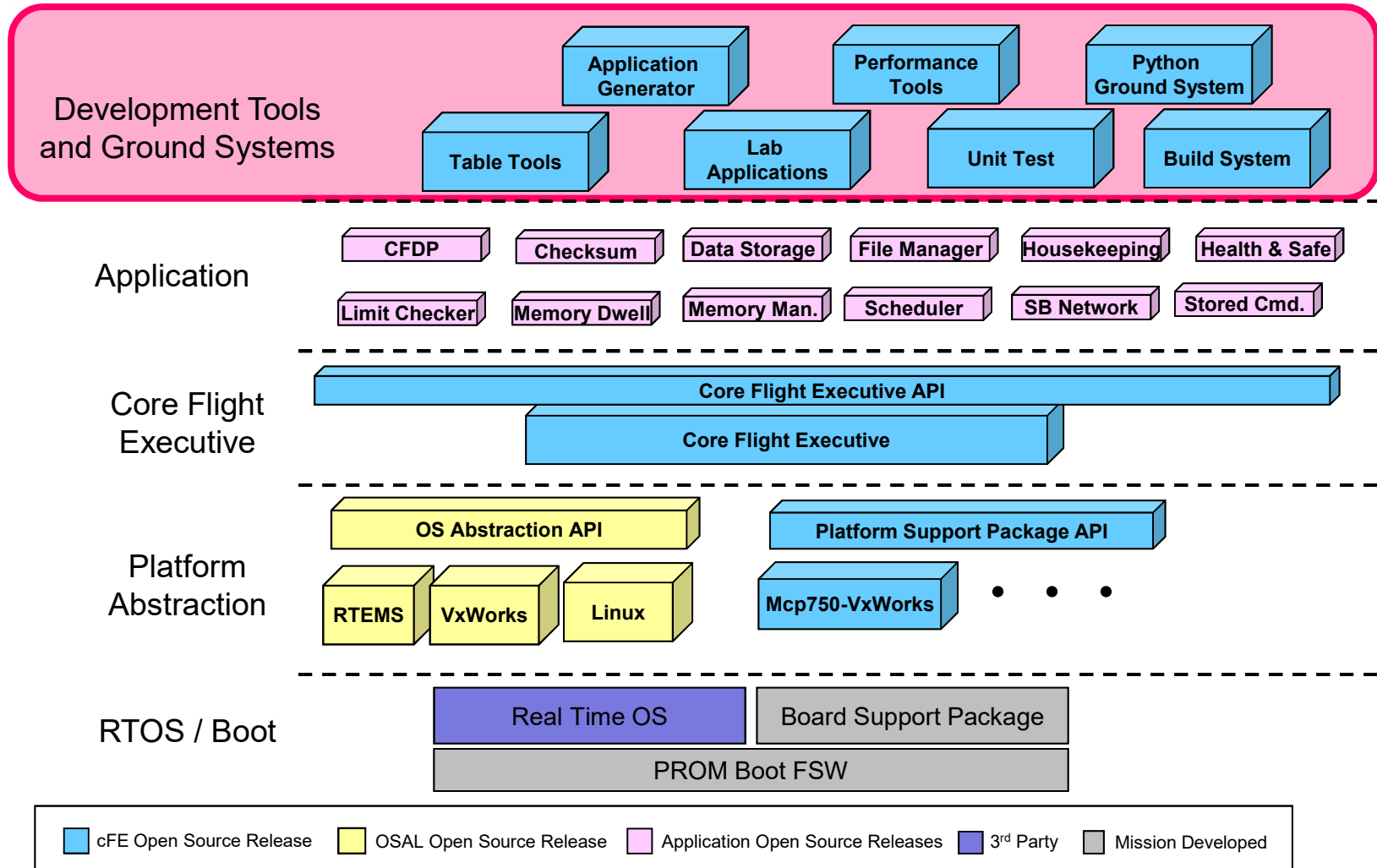


- 1. Introduction**
- 2. cFE Services**
  - a) Executive Services
  - b) Software Bus
  - c) Event Services
  - d) Time Services
  - e) Table Services
- 3. Application Layer**
  - a) cFS Applications
  - b) cFS Libraries

## **4. [Optional] Integration with COSMOS**



# COSMOS - cFS Context





# cFS and COSMOS



- **cFS has been used with several ground systems**
  - **ASIST**
  - **ITOS**
  - **COSMOS**
- **COSMOS is an open-source ground system solution**
  - <https://cosmosrb.com/>

**This module will show how to  
operate cFS with COSMOS**





# Module Agenda



- **Getting Started**
- **Defining Commands**
- **Defining Telemetry**
- **Creating Telemetry Displays**
- **Basic Scripting**
- **Test Runner**



National Aeronautics and Space Administration



# Prerequisites



# Module Prerequisites



- **Have a running cFS build environment that includes the cFS sample\_app**
  - This is the result of completing Exercise 1 in the main cFS training package
- **Have COSMOS installed on development machine**
  - Installation instructions here: <https://cosmosrb.com/docs/installation/>



# Exercise 0 – Build and Run the cFE



## Part 1 - Setup

To setup the cFS Bundle directly from the latest set of interoperable repositories:

```
git clone https://github.com/nasa/cFS.git
cd cFS
git checkout bootes-rc2
git submodule init
git submodule update
```

Subsequent exercises assume that cFS was cloned into the home directory (“~/cFS”)

Copy in the default makefile and definitions:

```
cp cfe/cmake/Makefile.sample Makefile
cp -r cfe/cmake/sample_defs sample_defs
```

If running on a standard Linux build as a normal user, allow OSAL “permissive mode” for best effort message queue depth and task priorities.

- Open the sample\_defs/default\_osconfig.cmake file
- Find the “OSAL\_CONFIG\_DEBUG\_PERMISSIVE\_MODE” parameter and set it to TRUE



# Exercise 0 – Build and Run the cFE



## Part 2 – Build and Run

The cFS Framework, including sample applications, will build and run on the pc-linux platform support package (should run on most Linux distributions), via the steps described in <https://github.com/nasa/cFE/tree/master/cmake/README.md>. Quick-start is below:

To prep, compile, and run (from cFS directory above):

```
make prep
make
make install
cd build/exe/cpu1/
./core-cpu1
```

Should see startup messages and CFE\_ES\_Main entering OPERATIONAL state. Note the code must be executed from the build/exe/cpu1 directory to find the startup script and shared objects.



# Exercise 0 Recap



cFE Version

cFE Services Started

```
ejtimmon@gs580s-582cfs6: ~/training/cFS/build/exe/cpu1
File Edit View Search Terminal Help
1980-013-04:03:58.22853 ES Startup: Calling CFE_ES_CDSEarlyInit
1980-013-04:03:58.22856 ES Startup: Calling CFE_EVS_EarlyInit
1980-013-04:03:58.22857 Event Log cleared following power-on reset
1980-013-04:03:58.22857 ES Startup: Calling CFE_SB_EarlyInit
1980-013-04:03:58.22862 SB internal message format: CCSDS Space Packet Protocol version 1
1980-013-04:03:58.22862 ES Startup: Calling CFE_TIME_EarlyInit
1980-012-14:03:20.00000 ES Startup: Calling CFE_TBL_EarlyInit
1980-012-14:03:20.00010 ES Startup: Calling CFE_FS_EarlyInit
1980-012-14:03:20.00017 ES Startup: Core App: CFE_EVS created. App ID: 0
EVS Port1 42/1/CFE_EVS 1: cFE EVS Initialized. cFE DEVELOPMENT BUILD v6.7.0+dev292 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 42/1/CFE_EVS 14: No subscribers for MsgId 0x808, sender CFE EVS
1980-012-14:03:20.05037 ES Startup: Core App: CFE_SB created. App ID: 1
1980-012-14:03:20.05042 SB:Registered 4 events for filtering
EVS Port1 42/1/CFE_SB 1: cFE SB Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808, sender CFE SB
1980-012-14:03:20.10066 ES Startup: Core App: CFE_ES created. App ID: 2
EVS Port1 42/1/CFE_ES 1: cFE ES Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808, sender CFE_ES
EVS Port1 42/1/CFE_ES 2: cFS Versions: cfe v6.7.0+dev292, osal v5.0.0+dev247, psp v1.4.0+dev76. cFE chksm 7319
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808, sender CFE_ES
EVS Port1 42/1/CFE_ES 91: Mission bootes-rc2.sample, cFE git version: v6.8.0-rc1-1-gef5291a, OSAL git version: v5.1.0-rc1-1-gf7f39f1
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808, sender CFE_ES
EVS Port1 42/1/CFE_ES 92: Build 202012071255 ejtimmon@gs580s-582cfs6
1980-012-14:03:20.15101 ES Startup: Core App: CFE_TIME created. App ID: 3
EVS Port1 42/1/CFE_TIME 1: cFE TIME Initialized
1980-012-14:03:20.20120 ES Startup: Core App: CFE_TBL created. App ID: 4
EVS Port1 42/1/CFE_TBL 1: cFE TBL Initialized. cFE DEVELOPMENT BUILD v6.7.0+dev292 (Codename: Bootes), Last Official Release: cfe v6.7.0
1980-012-14:03:20.25155 ES Startup: Finished ES CreateObject table entries.
1980-012-14:03:20.25157 ES Startup: CFE_ES_Main entering CORE READY state
1980-012-14:03:20.25197 ES Startup: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:03:20.25264 ES Startup: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.1.0+dev27, Last Official Release: v1.1.0
1980-012-14:03:20.25377 ES Startup: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25448 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25572 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.1.0+dev65, Last Official Release: v1.1.0
1980-012-14:03:20.25652 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25705 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25879 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.25961 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
1980-012-14:03:20.26029 ES Startup: SCH_LAB_APP loaded and created
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.3.0+dev44, Last Official Release: v2.3.0, Awaiting enable command.
1980-012-14:03:20.25994 CI_LAB listening on UDP port: 1234
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.3.0+dev36, Last Official Release: v2.3.0
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.3.0+dev37, Last Official Release: v2.3.0
1980-012-14:03:20.31055 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.31058 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
```



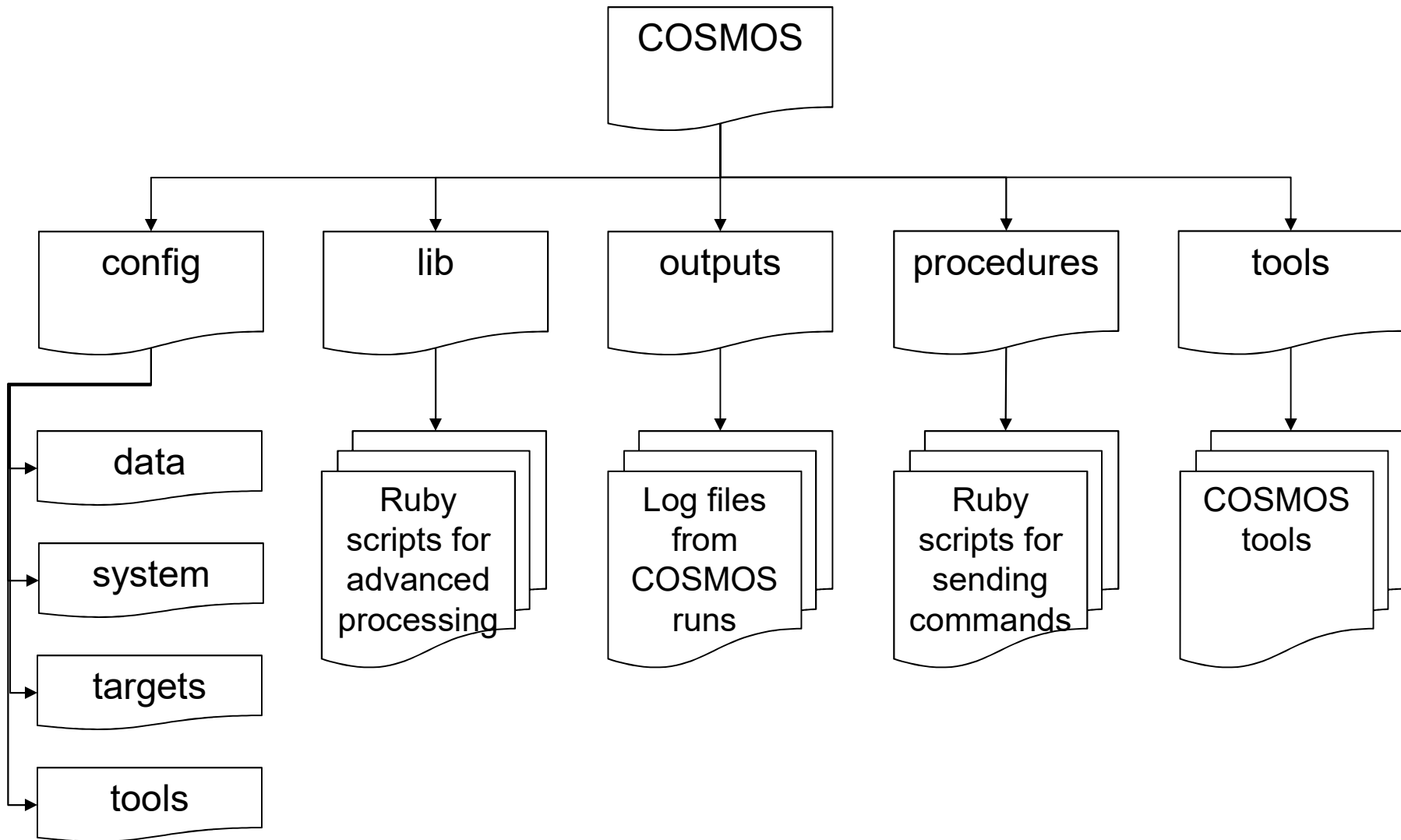
National Aeronautics and Space Administration



# Getting Started



# Navigating COSMOS



Reference: <https://cosmosrb.com/docs/structure/>





# Terms



- **Target – a destination for commands and/or a source of telemetry**
  - When communicating with cFS, each cFS app is typically a target
  - Much of COSMOS is organized around targets
- **Tool – one of the main “out of the box” components of COSMOS**
  - Everything on the “Launcher” screen is a tool
  - Each tool can be configured
  - The “Command and Telemetry Server” must be running in order to use the other tools
- **Interface – mechanism by which COSMOS communicates with a given target**



# System Configuration (system.txt)



Must either auto declare (all targets)  
or declare each target individually.  
Typically needs to be modified in a  
real system

Connection details. Can often be  
left unchanged.

Scripts that read and write log files.  
Can be changed or left as-is.

Default locations of log files. Can  
often be left unchanged.

```
ejtimmon@gs580s-582cfs: ~/cosmosdemo
1 # Declare Targets that make up the system
2 # DECLARE_TARGET target_name [substitute_name]
3
4 # AUTO_DECLARE_TARGETS
5 DECLARE_TARGET INST
6 DECLARE_TARGET INST_INST2
7 DECLARE_TARGET EXAMPLE
8 DECLARE_TARGET TEMPLATED
9 DECLARE_TARGET DART
10 DECLARE_TARGET SYSTEM
11 DECLARE_TARGET SAMPLE
12 DECLARE_TARGET TO_LAB
13
14 # Listen Hosts - Ip addresses or hostnames to listen on when running the tools
15 LISTEN_HOST CTS_API 127.0.0.1
16 LISTEN_HOST TLMVIEWER_API 127.0.0.1
17 LISTEN_HOST CTS_PREIDENTIFIED 0.0.0.0 # 127.0.0.1 is more secure if you don't need external connections
18 LISTEN_HOST CTS_CMD_ROUTER 0.0.0.0 # 127.0.0.1 is more secure if you don't need external connections
19 LISTEN_HOST REPLAY_API 127.0.0.1
20 LISTEN_HOST REPLAY_PREIDENTIFIED 0.0.0.0 # 127.0.0.1 is more secure if you don't need external connections
21 LISTEN_HOST REPLAY_CMD_ROUTER 0.0.0.0 # 127.0.0.1 is more secure if you don't need external connections
22 LISTEN_HOST DART_STREAM 0.0.0.0 # 127.0.0.1 is more secure if you don't need external connections
23 LISTEN_HOST DART_DECOM 0.0.0.0 # 127.0.0.1 is more secure if you don't need external connections
24 LISTEN_HOST DART_MASTER 0.0.0.0 # 127.0.0.1 is more secure if you don't need external connections
25
26 # Connect Hosts - Ip addresses or hostnames to connect to when running the tools
27 CONNECT_HOST CTS_API 127.0.0.1
28 CONNECT_HOST TLMVIEWER_API 127.0.0.1
29 CONNECT_HOST CTS_PREIDENTIFIED 127.0.0.1
30 CONNECT_HOST CTS_CMD_ROUTER 127.0.0.1
31 CONNECT_HOST REPLAY_API 127.0.0.1
32 CONNECT_HOST REPLAY_PREIDENTIFIED 127.0.0.1
33 CONNECT_HOST REPLAY_CMD_ROUTER 127.0.0.1
34 CONNECT_HOST DART_STREAM 127.0.0.1
35 CONNECT_HOST DART_DECOM 127.0.0.1
36 CONNECT_HOST DART_MASTER 127.0.0.1
37
38 # Ethernet Ports
39 PORT CTS_API 7777
40 PORT TLMVIEWER_API 7778
41 PORT CTS_PREIDENTIFIED 7779
42 PORT CTS_CMD_ROUTER 7780
43 PORT REPLAY_API 7877
44 PORT REPLAY_PREIDENTIFIED 7879
45 PORT REPLAY_CMD_ROUTER 7880
46 PORT DART_STREAM 8777
47 PORT DART_DECOM 8779
48 PORT DART_MASTER 8780
49
50 # Default Packet Log Writer and Reader
51 DEFAULT_PACKET_LOG_WRITER packet_log_writer.rb
52 DEFAULT_PACKET_LOG_READER packet_log_reader.rb
53
54 # Paths
55 PATH LOGS ./outputs/logs
56 PATH TMP ./outputs/tmp
57 PATH SAVED_CONFIG ./outputs/saved_config
58 PATH TABLES ./outputs/tables
59 PATH HANDBOOKS ./outputs/handbooks
60 PATH PROCEDURES ./procedures
61 PATH SEQUENCES ./outputs/sequences
62 PATH DART_DATA ./outputs/dart/data
63 PATH DART_LOGS ./outputs/dart/logs
```



# Command/Telemetry Server Configuration (cmd\_tlm\_server.txt)



Configure the log writer. Note that the ruby script is the same one specified in system.txt

```
1 TITLE 'COSMOS Command and Telemetry Server - Demo Configuration'
2
3 # PACKET_LOG_WRITER Parameter Notes
4 # nil:use default log names
5 # true: logging enabled
6 # nil: Don't cycle logs based on time
7 # 2000000000: Create new log after 2 Billion bytes
8 # nil: Use the default log directory
9 # false: Log synchronously - more efficient
10 PACKET_LOG_WRITER DEFAULT packet_log_writer.rb nil true nil 2000000000 nil false
11 # PACKET_LOG_WRITER SYSTEMLOG packet_log_writer.rb system
```

Alternate ways of defining interfaces. Each target must be associated with an interface in order to be used. Interfaces are often customized, though there are some built-in choices.

```
12
13 # Explicitly declare these interfaces since we're using name substitution
14 INTERFACE_TARGET INST cmd_tlm_server.txt # Use cmd_tlm_server.txt in targets/INST
15 INTERFACE_TARGET INST2 cmd_tlm_server2.txt # Use cmd_tlm_server2.txt in targets/INST
16
17 # Here is an example of declaring the interface directly
18 INTERFACE SYSTEM_INT cmd_tlm_server_interface.rb
19     TARGET SYSTEM
20     DISABLE_DISCONNECT
21     # LOG SYSTEMLOG
22     # DONT_LOG
23     # DONT_CONNECT
24     # DONT_RECONNECT
25     # RECONNECT_DELAY 15.0
26     # LOG_RAW
27
28 # Auto interface the rest of the targets by using their cmd_tlm_server.txt file
29 AUTO_INTERFACE_TARGETS
```

Optional router specification

```
30
31 ROUTER INST_ROUTER tcpip_server_interface.rb 2055 2055 10.0 nil LENGTH 32 16 7
32     OPTION LISTEN_ADDRESS 127.0.0.1
33     ROUTE INST_INT
34     # DONT_CONNECT
35     # DONT_RECONNECT
36     # DISABLE_DISCONNECT
37     # RECONNECT_DELAY 15.0
38     # LOG_RAW
```

Optional background task

```
39
40 BACKGROUND_TASK example_background_task.rb
41     STOPPED
42 BACKGROUND_TASK limits_groups.rb 5 # Initial delay to allow interfaces to connect
```



# Exercise 1 – Create a new target



**Objective: Create a new target for the sample app**

## Part 1 – Add sample\_app to COSMOS

1. Navigate to the config/targets directory in COSMOS
2. Create a directory called “SAMPLE”
3. Enter the “SAMPLE” directory
4. Create a file called “target.txt”
5. Navigate to the directory “cosmosdemo/config/system”
6. Open the file “system.txt” and add the line “DECLARE\_TARGET SAMPLE”
  - This tells COSMOS to look for the target you just created
7. Navigate to the directory “cosmosdemo/config/tools/cmd\_tlm\_server”
8. Open the “cmd\_tlm\_server.txt” file
9. Under the PACKET LOG WRITER section, create a LOCAL interface, add the line “TARGET sample” under it

```
INTERFACE LOCAL udp_interface.rb 127.0.0.1 1234 1235 nil nil 128 nil nil
                TARGET sample
```

- For simplicity you can delete everything in the “cmd\_tlm\_server.txt” file below the LOCAL interface

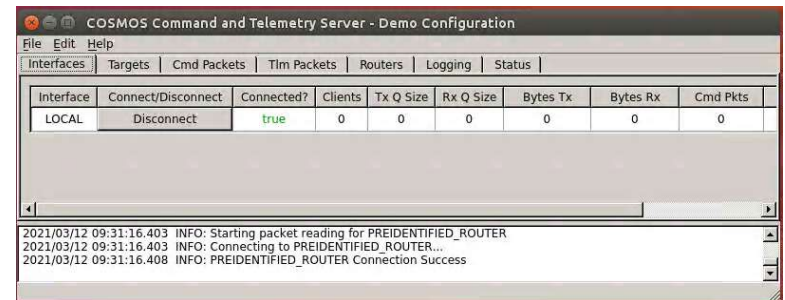
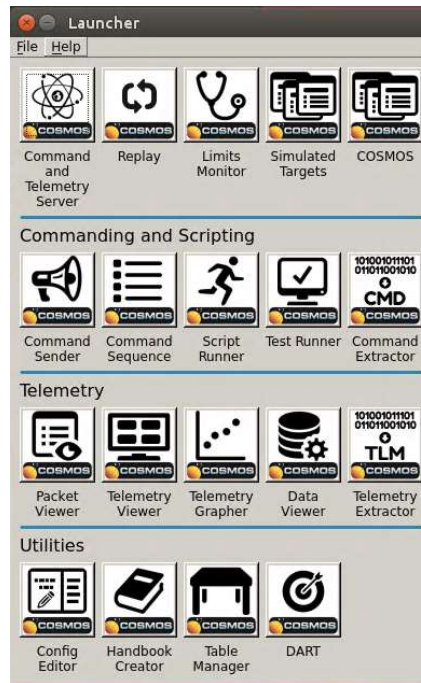


# Exercise 1 – Part 2



## Part 2 – Launch COSMOS

1. Enter the main Cosmos directory and launch COSMOS with “ruby Launcher”
  - You may need to click “Update Project CRCs” when COSMOS starts up
2. Click on “Command and Telemetry Server” and click “OK” on the dialog that pops up



Successful startup confirms that Part 1 was done correctly.



# Exercise 1 - Recap



```
ejtimmon@gs580s-582cfs: ~/cosmosdemo
File Edit View Search Terminal Help
1 # Declare Targets that make up the system
2 # DECLARE_TARGET target_name [substitute_name]
3
4 # AUTO_DECLARE_TARGETS
5 DECLARE_TARGET INST
6 DECLARE_TARGET INST INST2
7 DECLARE_TARGET EXAMPLE
8 DECLARE_TARGET TEMPLATED
9 DECLARE_TARGET DART
10 DECLARE_TARGET SYSTEM
11 DECLARE_TARGET SAMPLE
12
13 # Listen Hosts - Ip addresses or hostnames to listen on when running the tools
1,1 Top
```

system.txt

cmd\_tlm\_server.txt

```
ejtimmon@gs580s-582cfs: ~/cosmosdemo
File Edit View Search Terminal Help
1 TITLE 'COSMOS Command and Telemetry Server - Demo Configuration'
2
3 # PACKET_LOG_WRITER Parameter Notes
4 # nil: use default log names
5 # true: logging enabled
6 # nil: Don't cycle logs based on time
7 # 2000000000: Create new log after 2 Billion bytes
8 # nil: Use the default log directory
9 # false: Log synchronously - more efficient
10 PACKET_LOG_WRITER DEFAULT packet_log_writer.rb nil true nil 2000000000 nil false
11 # PACKET_LOG_WRITER SYSTEMLOG packet_log_writer.rb system
12
13 INTERFACE LOCAL udp_interface.rb 127.0.0.1 1234 1235 nil nil 128 nil nil
14 TARGET SAMPLE
1,1 Top
```



National Aeronautics and Space Administration



# Defining Commands



# Command Databases

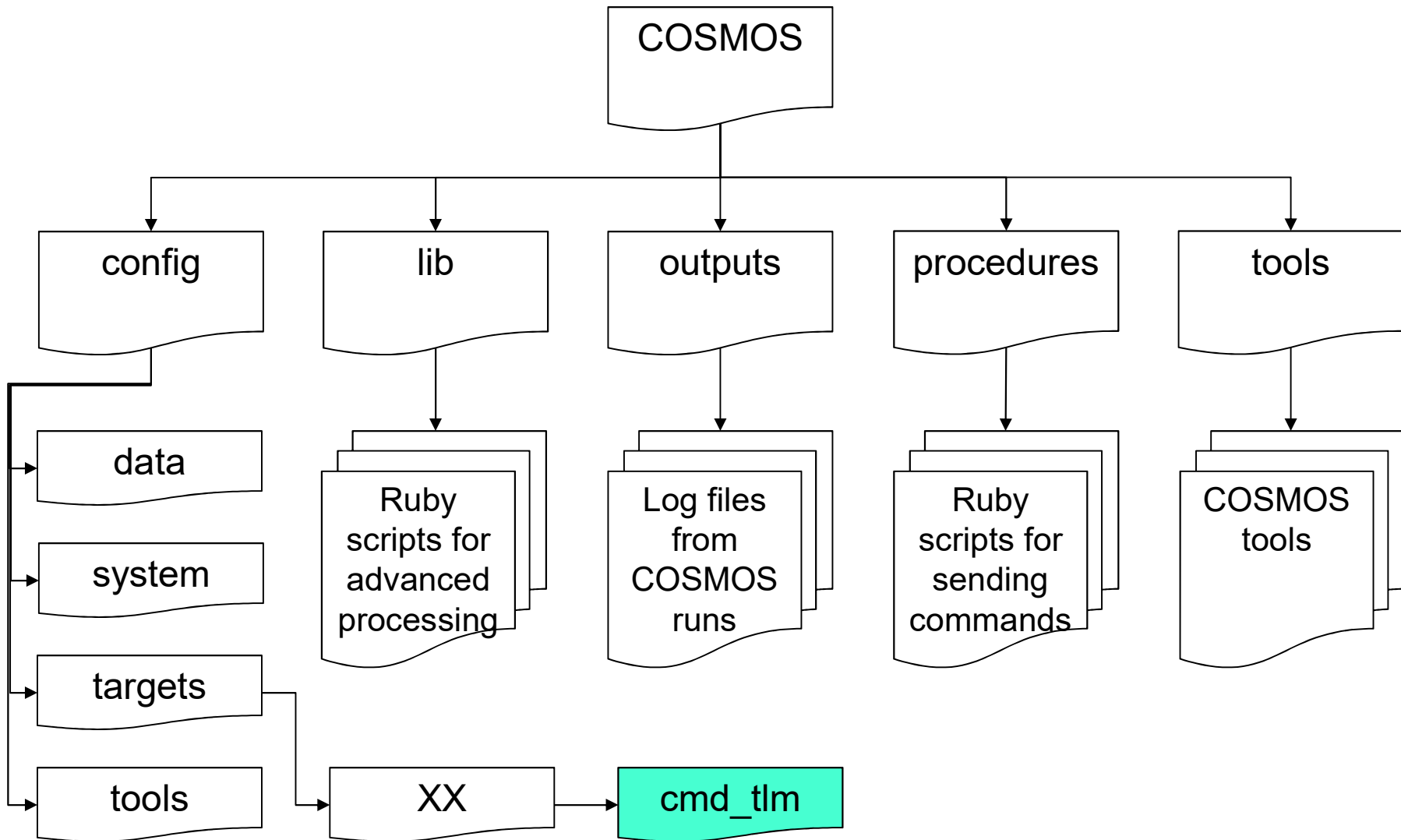


- **A command database defines the commands that can be sent to flight software**
- **COSMOS uses text-based command databases**
- **These databases must specify every field in a command (even those that don't change)**





# Location of Command Database





# Relationship with cFS



- **The COSMOS command database generally relies on the following files in a cFS app:**
  - XX\_msg.h
  - XX\_msgids.h
  - XX\_msgdefs.h
- **Each command message structure defined in XX\_msg.h should be defined in the command database**
- **The XX\_msgids.h and XX\_msgdefs.h files are used to find arguments to commands**
  - MsgID, Command Code, etc.



# Defining a Command Database



- **The command database file resides in the cmd\_tlm folder under the target**
- **In the file, each command starts with a COMMAND tag**  
COMMAND <Target> <Command Name> <Endianness> <Description>
- **Under the COMMAND tag, each parameter is appended to the command**
  - Using APPEND\_ID\_PARAMETER or APPEND\_PARAMETER

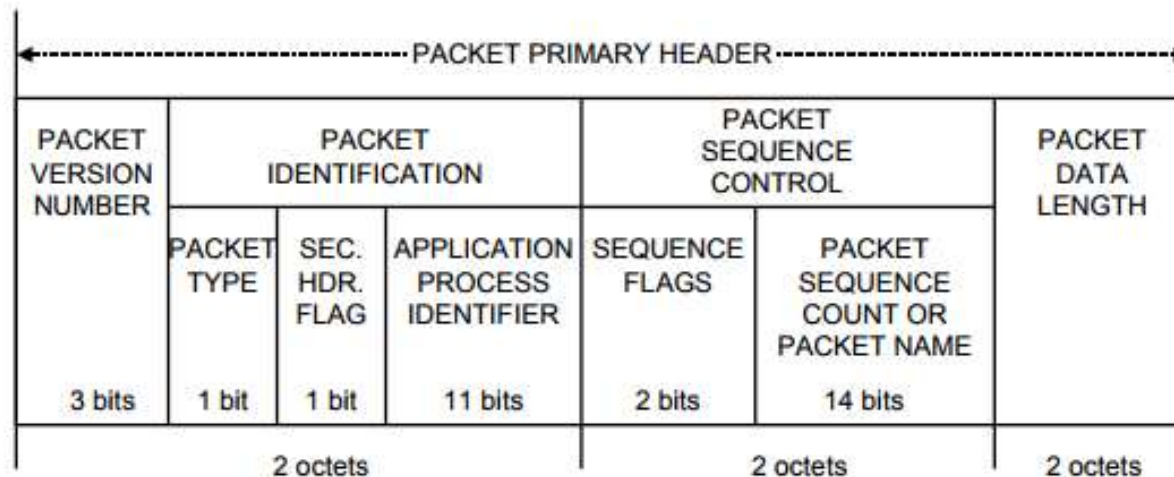
Reference: <https://cosmosrb.com/docs/command/>



# Review: cFE Software Bus Messages



- By default Consultative Committee for Space Data Systems (CCSDS) packets used to implement messages
- CCSDS Primary Header (Always big endian)



- **CCSDS Command Packets**
  - Secondary packet header contains a command function code
- **CCSDS Telemetry Packets**
  - Secondary packet header contains a time stamp of when the data was produced

```
typedef struct{
    CCSDS_PriHdr_t    Pri;
    CCSDS_CmdSecHdr_t Sec;
} CFE_SB_CmdHdr_t;
```

```
typedef struct{
    CCSDS_PriHdr_t    Pri;
    CCSDS_TlmSecHdr_t Sec;
} CFE_SB_TlmHdr_t;
```



# No-Op Example



- **Sample\_app No-Op Command in cFS:**

```
typedef struct
{
    uint8    CmdHeader[CFE_SB_CMD_HDR_SIZE];
} SAMPLE_NoArgsCmd_t;
```

- **Sample\_app No-Op Command in COSMOS:**

cFS CMD Secondary Header	cFS Primary Header	COMMAND	SAMPLE	SAMPLE_NOOP	BIG_ENDIAN	"Sample app NOOP Command"				
		APPEND_ID_PARAMETER	STREAM_ID	16	UINT	0x1882	0x1882	0x1882	""	
		APPEND_PARAMETER	SEQUENCE	16	UINT	0xC000	MAX_UINT16	0xC000	""	
		FORMAT_STRING	"0x%04X"							
		APPEND_PARAMETER	PKT_LEN	16	UINT	0x0001	0x0001	0x0001	""	
		FORMAT_STRING	"0x%04X"							
		APPEND_PARAMETER	CMD_ID	8	UINT	0	0	0	""	
		APPEND_PARAMETER	CHECKSUM	8	UINT	MIN_UINT8	MAX_UINT8	MIN_UINT8	""	

SAMPLE\_APP\_CMD\_MID  
from sample\_app\_msgids.h

SAMPLE\_APP\_NOOP\_CC  
from sample\_app\_msg.h



# Other Command Parameters



- **Command parameters can have types INT, UINT, FLOAT, DERIVED, STRING, BLOCK**
- **Parameter ranges are specified with Minimum, Maximum, and Default values**
- **For numbers, FORMAT\_STRING specifies the input format of the number**
  - Ex. "0x%04X" specifies input in hexadecimal
- **Parameters can also be selected from a drop down list using the STATE tag**

```
APPEND_PARAMETER ENABLE 32 UINT 0 1 0 "Enable setting"  
STATE FALSE 0  
STATE TRUE 1
```



# Exercise 2 – Part 1



**Objective: Create a command database for sample\_app and send commands to cFS (2 parts total)**

## **Part 1 – Add sample\_app to COSMOS**

1. Navigate to the config/targets/SAMPLE directory in COSMOS
2. Inside “target.txt”, add the following line: `COMMANDS sample_cmds.txt`
  - Note: This file tells COSMOS the name of the file containing the command database
3. Inside the “SAMPLE” directory”, create a “cmd\_tlm” directory
4. Inside the “cmd\_tlm” directory, create a file “sample\_cmds.txt”
5. Open the file `sample_cmds.txt`
6. Create a command definition for each command in `sample_app_msg.h`
  - You should have a total of 3 commands
  - They will be similar to the No-Op command example
  - `sample_app_msg.h` is located in the cFS tree at `apps/sample_app/fsw/src/sample_app_msg.h`



## Exercise 2 – Part 2



### Part 2 – Send commands to sample\_app

1. In a different terminal window, start the cFS
  1. Leave this running, but put the window aside and return to the terminal window with COSMOS
2. Enter the main Cosmos directory and launch COSMOS with “ruby Launcher”
  1. You may need to click “Update Project CRCs” when COSMOS starts up
3. Click on “Command and Telemetry Server” and click “OK” on the dialog that pops up
4. Click on the “Cmd Packets” tab and scroll down until you see the target SAMPLE on the left
5. Click on the “View in Command Sender” button beside “SAMPLE\_NOOP”
6. Click “Send” on the Command Sender window
  1. A no-op event message should show up in the cFS terminal window





# Exercise 2 - Recap



## The sample\_cmds.txt file:

```
COMMAND SAMPLE SAMPLE_NOOP BIG_ENDIAN "Sample_app NOOP Command"
  APPEND_ID_PARAMETER STREAM_ID 16 UINT 0x1882 0x1882 0x1882 ""
  APPEND_PARAMETER SEQUENCE 16 UINT 0xC000 MAX_UINT16 0xC000 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER PKT_LEN 16 UINT 0x0001 0x0001 0x0001 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER CMD_ID 8 UINT 0 0 0 ""
  APPEND_PARAMETER CHECKSUM 8 UINT MIN_UINT8 MAX_UINT8 MIN_UINT8 ""

COMMAND SAMPLE SAMPLE_RESET BIG_ENDIAN "Sample_app Reset Counters Command"
  APPEND_ID_PARAMETER STREAM_ID 16 UINT 0x1882 0x1882 0x1882 ""
  APPEND_PARAMETER SEQUENCE 16 UINT 0xC000 MAX_UINT16 0xC000 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER PKT_LEN 16 UINT 0x0001 0x0001 0x0001 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER CMD_ID 8 UINT 1 1 1 ""
  APPEND_PARAMETER CHECKSUM 8 UINT MIN_UINT8 MAX_UINT8 MIN_UINT8 ""

COMMAND SAMPLE SAMPLE_PROCESS BIG_ENDIAN "Sample_app Process Command"
  APPEND_ID_PARAMETER STREAM_ID 16 UINT 0x1882 0x1882 0x1882 ""
  APPEND_PARAMETER SEQUENCE 16 UINT 0xC000 MAX_UINT16 0xC000 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER PKT_LEN 16 UINT 0x0001 0x0001 0x0001 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER CMD_ID 8 UINT 2 2 2 ""
  APPEND_PARAMETER CHECKSUM 8 UINT MIN_UINT8 MAX_UINT8 MIN_UINT8 ""
```



# Exercise 2 - Recap



The image shows a multi-windowed application interface for the COSMOS Command and Telemetry Server. The main window, titled "COSMOS Command and Telemetry Server - Demo Configuration", has a menu bar with "File", "Edit", and "Help". Below the menu bar are tabs for "Interfaces", "Targets", "Cmd Packets", "Tlm Packets", "Routers", "Logging", and "Status". The "Interfaces" tab is active, displaying a table with the following data:

Interface	Connect/Disconnect	Connected?	Clients	Tx Q Size	Rx Q Size	Bytes Tx	Bytes Rx	Cmd Pkts
LOCAL	Disconnect	true	0	0	0	0	0	0

Below the table, there is a log of timestamps: 2020/07/24 13:54:50.256, 2020/07/24 13:54:50.256, 2020/07/24 13:54:50.256, and 2020/07/24 13:54:50.257.

A second window, also titled "COSMOS Command and Telemetry Server - Demo Configuration", is overlaid on the main window. It has the same menu bar and tabs. The "Targets" tab is active, displaying a table with the following data:

Target	Cmd Packets	Tlm Packets	View Raw	View in Command Sender
INST2 SLRPNLDEPLOY	0	0	View Raw	View in Command Sender
INST2 SLRPNLRESET	0	0	View Raw	View in Command Sender
SAMPLE SAMPLE_NOOP	0	0	View Raw	View in Command Sender
SAMPLE SAMPLE_PROCESS	0	0	View Raw	View in Command Sender
SAMPLE SAMPLE_RESET	0	0	View Raw	View in Command Sender

A red box highlights the row for "SAMPLE SAMPLE\_NOOP", and a red arrow points to it from the right.

A third window, titled "Command Sender", is overlaid on the bottom left. It has a menu bar with "File", "Mode", and "Help". It features a search bar, a "Target:" dropdown menu set to "SAMPLE", a "Command:" dropdown menu set to "SAMPLE\_NOOP", and a "Send" button. A red arrow points to the "Send" button from the right. Below the dropdowns are fields for "Description:" and "Parameters:". The "Parameters:" section contains a table with the following data:

Name	Value or State	Units	Description
STREAM_ID:	6274		
SEQUENCE:	0xC000		
PKT_LEN:	0x0001		
CMD_ID:	0		



## Exercise 2 - Recap



```
ejtimmon@gs580s-582cfs: ~/cFS/training/cFS/build/exe/cpu1
File Edit View Search Terminal Help
1980-012-14:03:20.25397 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25410 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25427 CI_LAB listening on UDP port: 1234
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized. Version 2.3.5.0
1980-012-14:03:20.25450 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25461 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.25515 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
1980-012-14:03:20.25525 ES Startup: SCH_LAB_APP loaded and created
SCH Lab Initialized. Version 2.3.7.0
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on TO_LAB_TLM_PIPE pipe,app
TO_LAB_APP
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on TO_LAB_TLM_PIPE pipe,app
TO_LAB_APP
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on TO_LAB_TLM_PIPE pipe,app
TO_LAB_APP
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on TO_LAB_TLM_PIPE pipe,app
TO_LAB_APP
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.7.0 Awaiting enable comm
and.
1980-012-14:03:20.30535 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30537 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE TIME 21: Stop FLYWHEEL
EVS Port1 42/1/SAMPLE_APP 3: SAMPLE: NOOP command Version 1.1.11.0
```



National Aeronautics and Space Administration



# Defining Telemetry



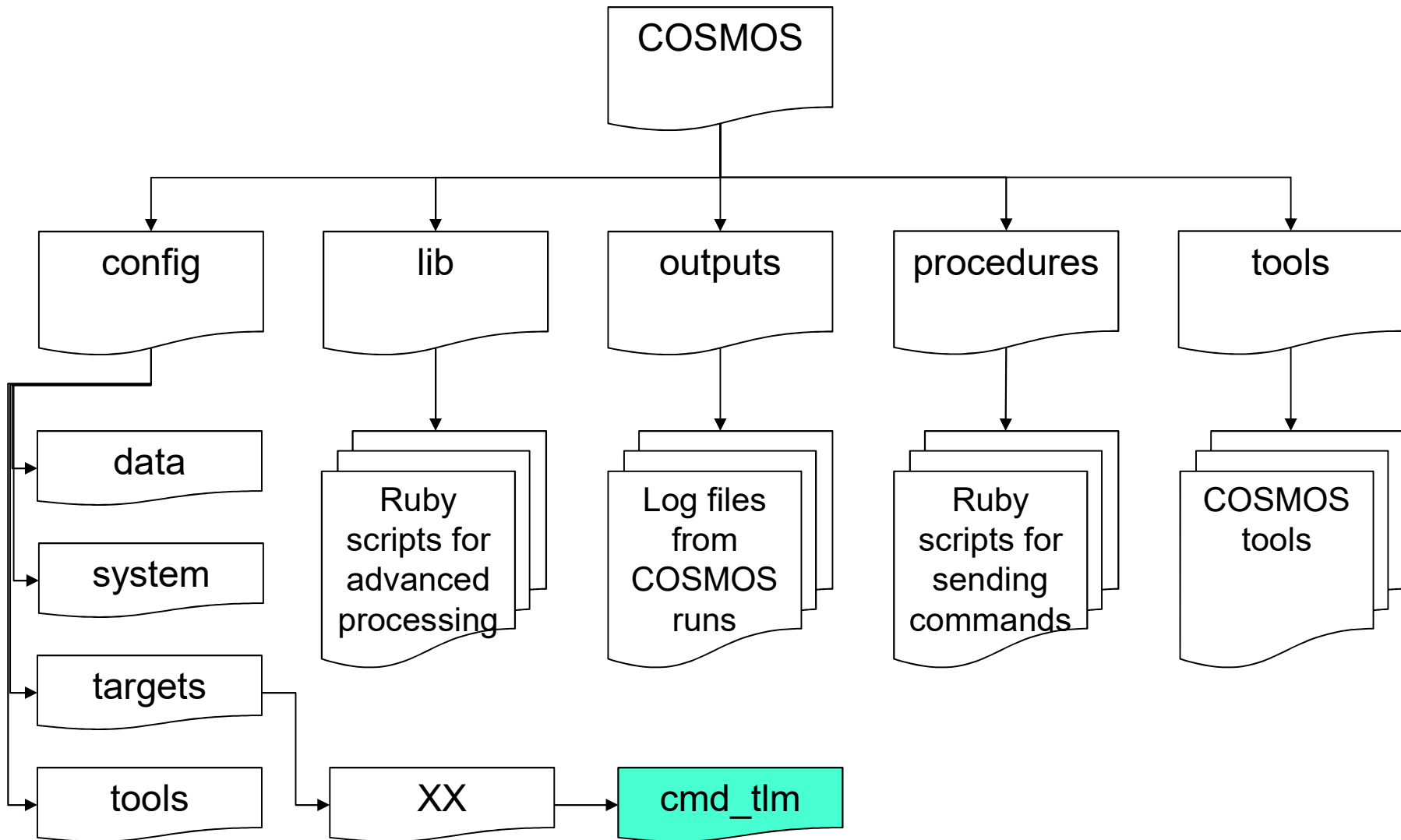
# Telemetry Databases



- **A telemetry database defines the telemetry that can be received from the flight software**
- **COSMOS uses text-based telemetry databases**
- **These databases must specify every field in a telemetry packet**
- **The database must also tell COSMOS how to identify the packet**



# Location of Telemetry Database





# Relationship with cFS



- **The COSMOS telemetry database generally relies on the following files in a cFS app:**
  - `XX_msg.h`
  - `XX_msgids.h`
- **Each telemetry message structure defined in `XX_msg.h` should be defined in the telemetry database**
- **The `XX_msgids.h` file is used to find the message ID**



# Defining a Telemetry Database



- **The telemetry database file resides in the cmd\_tlm folder under the target**
- **In the file, each command starts with a TELEMETRY tag**

TELEMETRY <Target> <Packet Name> <Endianness> <Description>

- **Under the TELEMETRY tag, each telemetry item is appended to the packet**
  - Using APPEND\_ID\_ITEM or APPEND\_ITEM

Reference: <https://cosmosrb.com/docs/telemetry/>





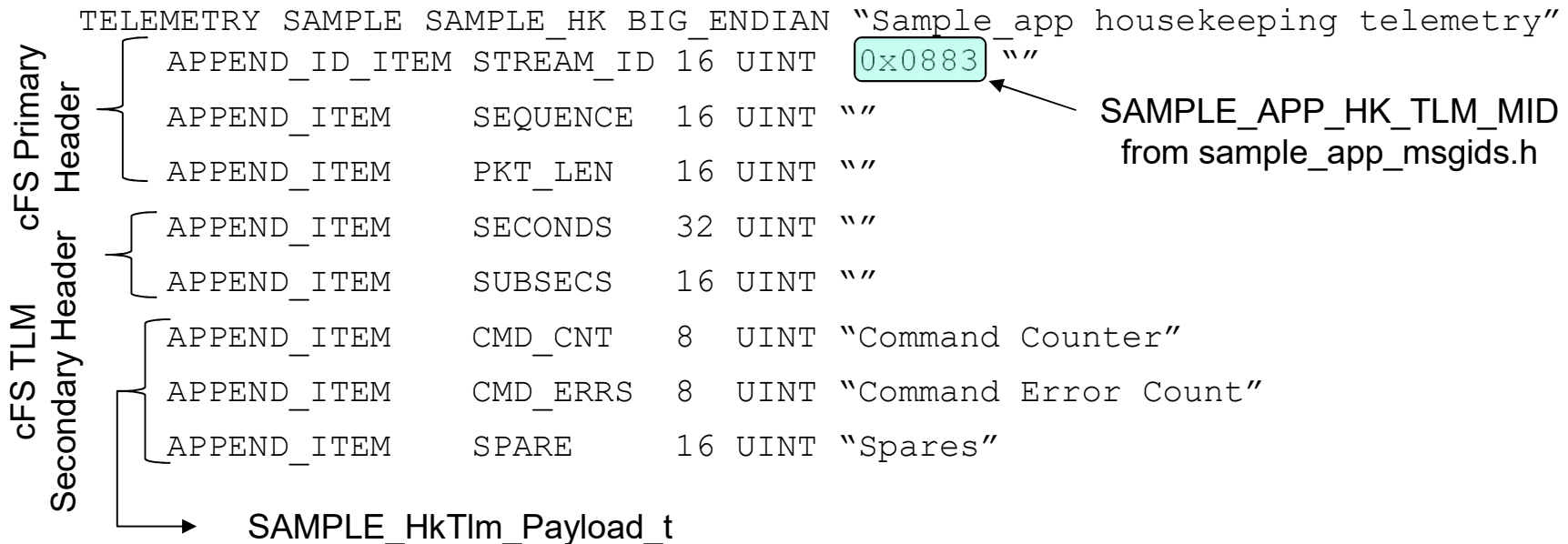
# HK Example



- **Sample\_app housekeeping telemetry in cFS:**

```
typedef struct {
    uint8_t TlmHeader[CFE_SB_TLM_HDR_SIZE];
    SAMPLE_HkTlm_Payload_t Payload;
} OS_PACK SAMPLE_HkTlm_t;
```

- **Sample\_app housekeeping packet in COSMOS:**





## Exercise 3



- **Objective: Create a telemetry database for sample\_app and get telemetry from cFS (4 parts)**

### **Part 1 – Add the sample\_app HK packet**

1. Navigate to the config/targets/sample directory in COSMOS
2. Open the target.txt file and add the line “TELEMETRY sample\_tlm.txt”
3. Inside the “cmd\_tlm” directory, create a file “sample\_tlm.txt”
4. Add the definition for the sample\_app housekeeping packet



## Exercise 3 – Part 2



### Part 2 – Add a target for TO\_Lab

*This is necessary because we need to enable telemetry in cFS before we will see it in COSMOS.*

1. Navigate to the config/targets directory in COSMOS
2. Create a directory called “TO\_LAB”
3. Enter the “TO\_LAB” directory
4. Create a file called “target.txt”
5. Inside “target.txt”, add the following line: `COMMANDS to_lab_cmds.txt`
6. Inside the “TO\_LAB” directory”, create a “cmd\_tlm” directory
7. Inside the “cmd\_tlm” directory, create a file “to\_lab\_cmds.txt”
8. Navigate to the directory “cosmosdemo/config/tools/cmd\_tlm\_server”
9. Open the “cmd\_tlm\_server.txt” file
10. Under the LOCAL interface, add the line “TARGET TO\_LAB”
11. Navigate to the directory “cosmosdemo/config/system”
12. Open the “system.txt” file and add “DECLARE\_TARGET TO\_LAB”

Same process used to add the “SAMPLE” target in Exercise 1



## Exercise 3 – Part 3



### Part 3 – Add a command database for to\_lab

1. Navigate back to the “config/targets/to\_lab/cmd\_tlm” directory in COSMOS
2. Open the file “to\_lab\_cmds.txt”
3. Create a command definition for the TO\_LAB\_EnableOutput\_t command
  - This definition is located in to\_lab\_msg.h (located in the cFS directory under apps/to\_lab/fsw/src/)



## Exercise 3 – Part 4



### Part 4 – Send commands/receive telemetry from cFS

1. Enter the main Cosmos directory and launch COSMOS with “ruby Launcher”
  1. You may need to click “Update Project CRCs” when COSMOS starts up
2. On the Launcher window, click on “Command and Telemetry Server” and click “OK” on the dialog that pops up
3. On the Launcher window, click on “Command Sender”
4. In the drop-down list beside target, select TO\_LAB. The command field should automatically update to “TO\_LAB\_ENABLE”. Click Send.
  1. An event message should appear in the cFS window
  2. On the “Tim Packets” tab of the “Command and Telemetry Server” window, the count of “SAMPLE\_HK” packets should be incrementing.
5. Click on the “View in Packet Viewer” button beside “SAMPLE\_HK”
  1. At this point, the “CMD\_CNT” field should be “1” if you are still running the same cFS instance as in Exercise 1, or 0 if you restarted cFS
6. Send a SAMPLE\_APP NOOP command as in Exercise 1
  1. The “CMD\_CNT” field in the packet viewer should increment by 1



## Exercise 3 - Recap



### The sample\_tlm.txt file:

```
TELEMETRY SAMPLE SAMPLE_HK BIG_ENDIAN "Sample_app housekeeping telemetry"  
  APPEND_ID_ITEM STREAM_ID 16 UINT 0x0883 ""  
    FORMAT_STRING "0x%04X"  
  APPEND_ITEM SEQUENCE 16 UINT ""  
    FORMAT_STRING "0x%04X"  
  APPEND_ITEM PKT_LEN 16 UINT ""  
  APPEND_ITEM SECONDS 32 UINT ""  
  APPEND_ITEM SUBSECS 16 UINT ""  
  APPEND_ITEM CMD_ERRS 8 UINT "Command Counter"  
  APPEND_ITEM CMD_CNT 8 UINT "Command Error Count"  
  APPEND_ITEM SPARE 16 UINT "Spares"
```



# Exercise 3 - Recap



## The to\_lab\_cmds.txt file:

```
COMMAND TO_LAB TO_LAB_ENABLE BIG_ENDIAN "TO_Lab enable telemetry"
  APPEND_ID_PARAMETER STREAM_ID 16 UINT 0x1880 0x1880 0x1880 ""
  APPEND_PARAMETER SEQUENCE 16 UINT 0xC000 MAX_UINT16 0xC000 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER PKT_LEN 16 UINT 0x0001 0x0001 0x0001 ""
    FORMAT_STRING "0x%04X"
  APPEND_PARAMETER CMD_ID 8 UINT 6 6 6 ""
  APPEND_PARAMETER CHECKSUM 8 UINT MIN_UINT8 MAX_UINT8 MIN_UINT8 ""
  APPEND_PARAMETER DEST_IP 128 STRING "127.0.0.1" "Destination IP"
```



# Exercise 3 - Recap



```
ejtimmon@gs580s-582cfs: ~/cosmosdemo/config
File Edit View Search Terminal Help
1 # Declare Targets that make up the system
2 # DECLARE_TARGET target_name [substitute_name]
3
4 # AUTO_DECLARE_TARGETS
5 DECLARE_TARGET INST
6 DECLARE_TARGET INST INST2
7 DECLARE_TARGET EXAMPLE
8 DECLARE_TARGET TEMPLATED
9 DECLARE_TARGET DART
10 DECLARE_TARGET SYSTEM
11 DECLARE_TARGET SAMPLE
12 DECLARE_TARGET TO_LAB
13
```

system.txt

cmd\_tlm\_server.txt

```
ejtimmon@gs580s-582cfs: ~/cosmosdemo/config
File Edit View Search Terminal Help
1 TITLE 'COSMOS Command and Telemetry Server - Demo Configuration'
2
3 # PACKET_LOG_WRITER Parameter Notes
4 # nil:use default log names
5 # true: logging enabled
6 # nil: Don't cycle logs based on time
7 # 2000000000: Create new log after 2 Billion bytes
8 # nil: Use the default log directory
9 # false: Log synchronously - more efficient
10 PACKET_LOG_WRITER DEFAULT packet_log_writer.rb nil true nil 2000000000 nil false
11 # PACKET_LOG_WRITER SYSTEMLOG packet_log_writer.rb system
12
13 INTERFACE LOCAL udp_interface.rb 127.0.0.1 1234 1235 nil nil 128 nil nil
14 TARGET SAMPLE
15 TARGET TO_LAB
16
```







# Exercise 3 - Recap



```
ejtimmon@gs580s-582cfs: ~/cFS/training/cFS/build/exe/cpu1
File Edit View Search Terminal Help
SAMPLE Lib Initialized. Version 1.1.4.0
1980-012-14:11:03.25205 ES Startup: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:11:03.25216 ES Startup: SAMPLE_APP loaded and created
1980-012-14:11:03.25248 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:11:03.25256 ES Startup: CI_LAB_APP loaded and created
1980-012-14:11:03.25288 ES Startup: Loading file: /cf/to_lab.so, APP: T0_LAB_APP
1980-012-14:11:03.25296 ES Startup: T0_LAB_APP loaded and created
1980-012-14:11:03.25370 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
1980-012-14:11:03.25440 ES Startup: SCH_LAB_APP loaded and created
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.11.0
1980-012-14:11:03.25571 CI_LAB listening on UDP port: 1234
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized. Version 2.3.5.0
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on T0_LAB_TLM_PIPE pipe,app T0_LAB_APP
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on T0_LAB_TLM_PIPE pipe,app T0_LAB_APP
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on T0_LAB_TLM_PIPE pipe,app T0_LAB_APP
EVS Port1 42/1/CFE_SB 7: Duplicate Subscription,MsgId 0x0 on T0_LAB_TLM_PIPE pipe,app T0_LAB_APP
EVS Port1 42/1/T0_LAB_APP 1: T0 Lab Initialized. Version 2.3.7.0 Awaiting enable command.
SCH Lab Initialized. Version 2.3.7.0
1980-012-14:11:03.30502 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:11:03.30504 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE TIME 21: Stop FLYWHEEL
EVS Port1 42/1/T0_LAB_APP 3: T0 telemetry output enabled for IP
EVS Port1 42/1/SAMPLE_APP 3: SAMPLE: NOOP command Version 1.1.11.0
```



National Aeronautics and Space Administration



# Creating Telemetry Displays



# Telemetry Displays

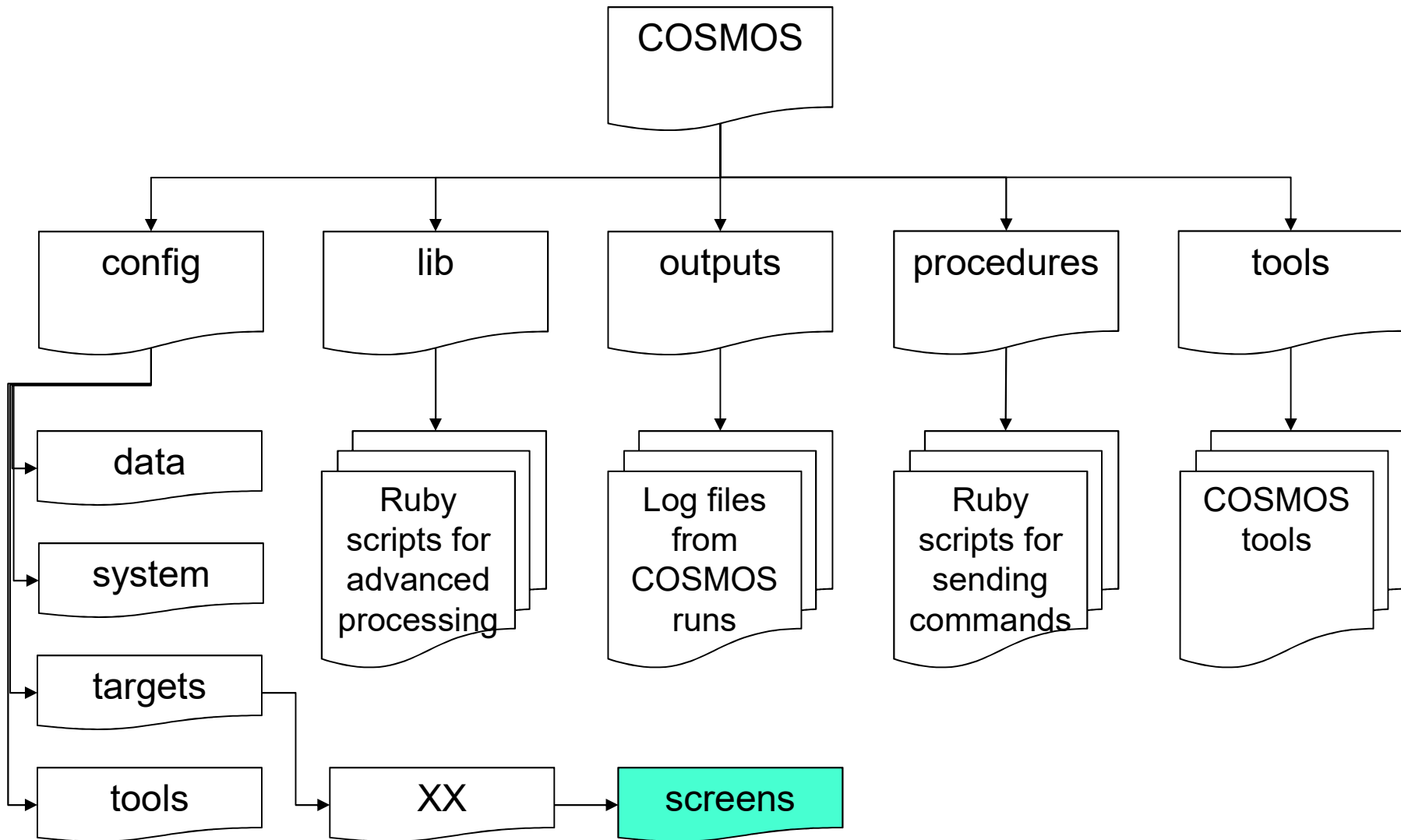


- **Provides a way to create custom telemetry displays that can be used in place of the packet viewer**
- **Can display values in “human readable” format**
- **Can display all data types**
- **By default, will automatically show staleness by graying out fields**
- **Provides a number of different layout tags for formatting pages**
- **Built-in widgets allow custom limit highlighting, graphing, and trending**
- **Interactive widgets can be tied to ruby scripts to initiate actions**

Reference: <https://cosmosrb.com/docs/screens/>



# Location of Telemetry Displays





## Exercise 4



### **Objective: Create a telemetry display for sample\_app**

1. Navigate to the config/targets/SAMPLE directory in COSMOS
2. Create a directory called "screens"
3. Inside the "screens" directory, create a file "sample\_screen.txt"
4. Develop a screen that displays the sample\_app housekeeping packet
  - Try to experiment with different layouts
  - Try changing the background color of the screen
5. Enter the main Cosmos directory and launch COSMOS with "ruby Launcher"
  1. You may need to click "Update Project CRCs" when COSMOS starts up
6. On the Launcher window, click on "Command and Telemetry Server" and click "OK" on the dialog that pops up
7. On the Launcher window, click on "Telemetry Viewer"
8. Click on "Show Screen" beside "SAMPLE\_SCREEN"



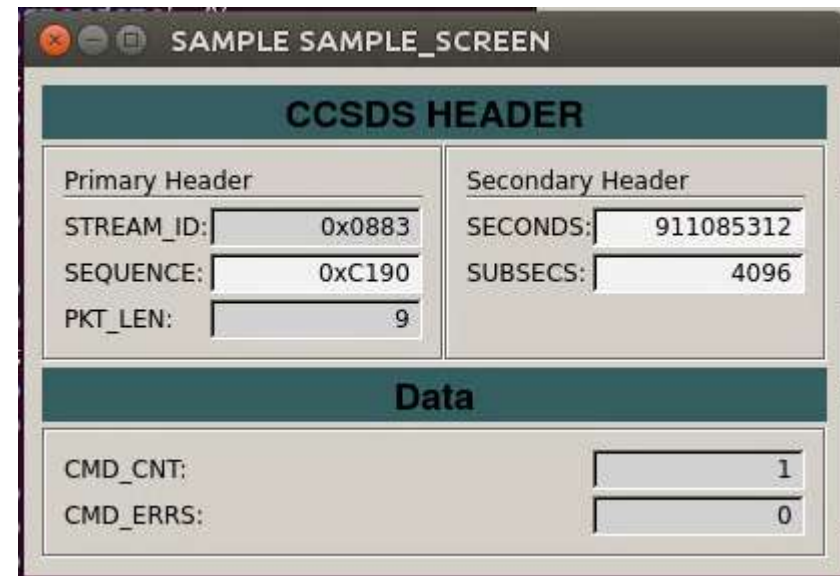
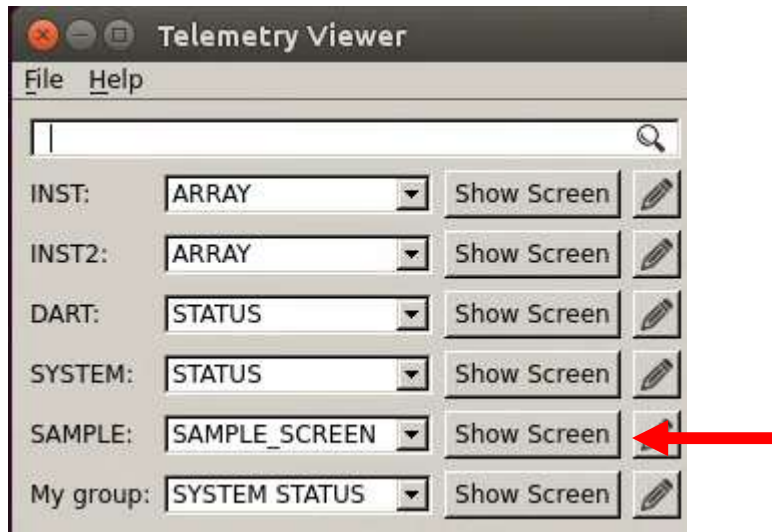
# Exercise 4 - Recap



```
ejtimmon@gs580s-582cfs: ~/cosmosdemo
File Edit View Search Terminal Help
1 SCREEN AUTO AUTO 0.5
2
3 VERTICAL
4
5     TITLE "CCSDS HEADER"
6     SETTING BACKCOLOR 54 95 98
7
8     HORIZONTAL
9
10        VERTICALBOX
11            SECTIONHEADER "Primary Header"
12
13                LABELFORMATVALUE SAMPLE SAMPLE_HK STREAM_ID "0x%04X" FORMATTED
14                LABELFORMATVALUE SAMPLE SAMPLE_HK SEQUENCE "0x%04X" FORMATTED
15                LABELFORMATVALUE SAMPLE SAMPLE_HK PKT_LEN "%d"
16            END
17        VERTICALBOX
18            SECTIONHEADER "Secondary Header"
19
20                LABELFORMATVALUE SAMPLE SAMPLE_HK SECONDS "%d"
21                LABELFORMATVALUE SAMPLE SAMPLE_HK SUBSECS "%d"
22            END
23    END
24
25    TITLE "Data"
26    SETTING BACKCOLOR 54 95 98
27
28    VERTICALBOX
29        LABELFORMATVALUE SAMPLE SAMPLE_HK CMD_CNT "%d"
30        LABELFORMATVALUE SAMPLE SAMPLE_HK CMD_ERRS "%d"
31    END
32 END
33
34 END
35
```



# Exercise 4 - Recap







National Aeronautics and Space Administration



# Basic Scripting



# COSMOS Scripts

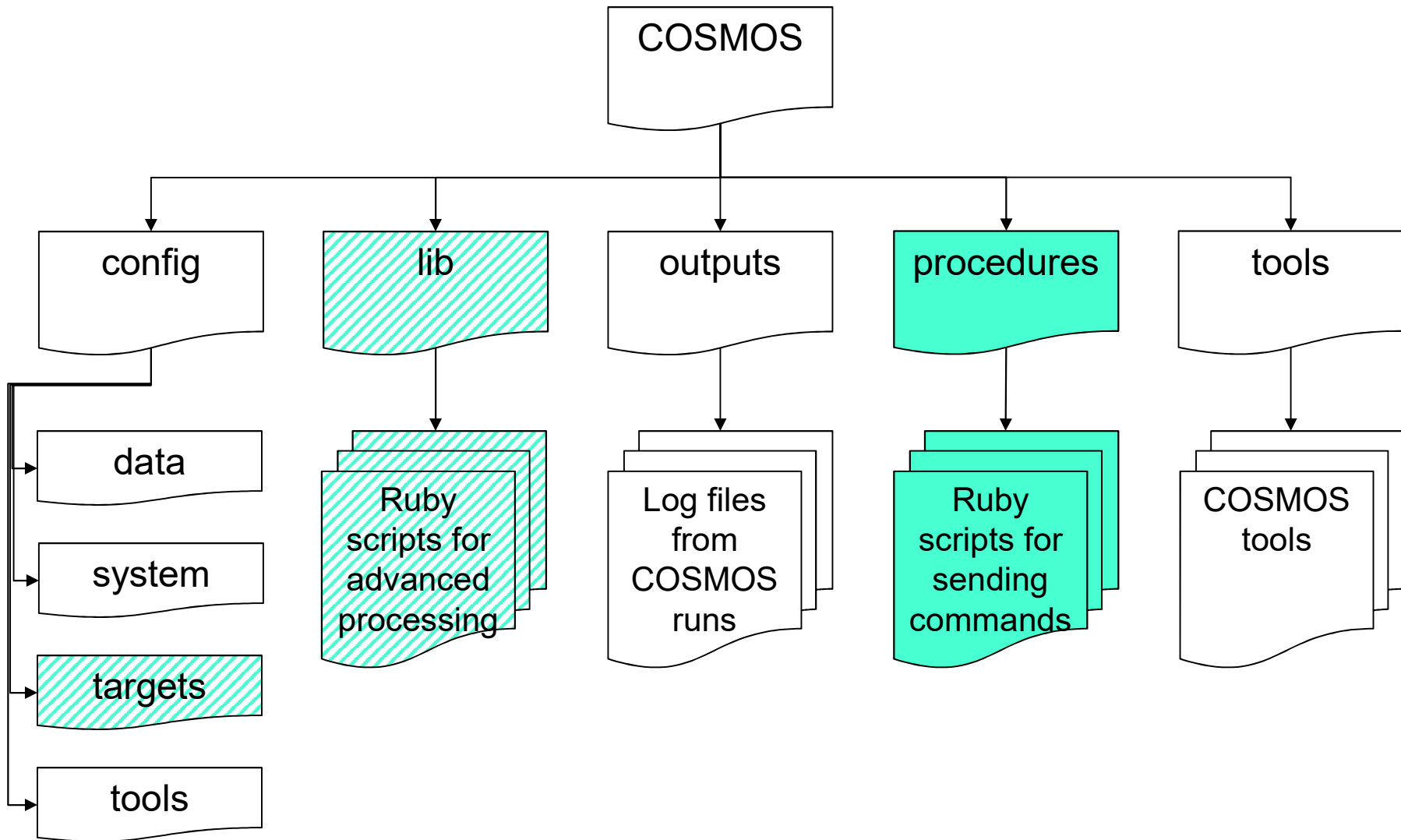


- **COSMOS provides the ability to develop Ruby scripts**
- **Ruby scripts can reference any defined commands and telemetry**
- **This is useful for testing and repeated onboard operations**

Reference: <https://cosmosrb.com/docs/scripting/>



# Location of Scripts





## Exercise 5

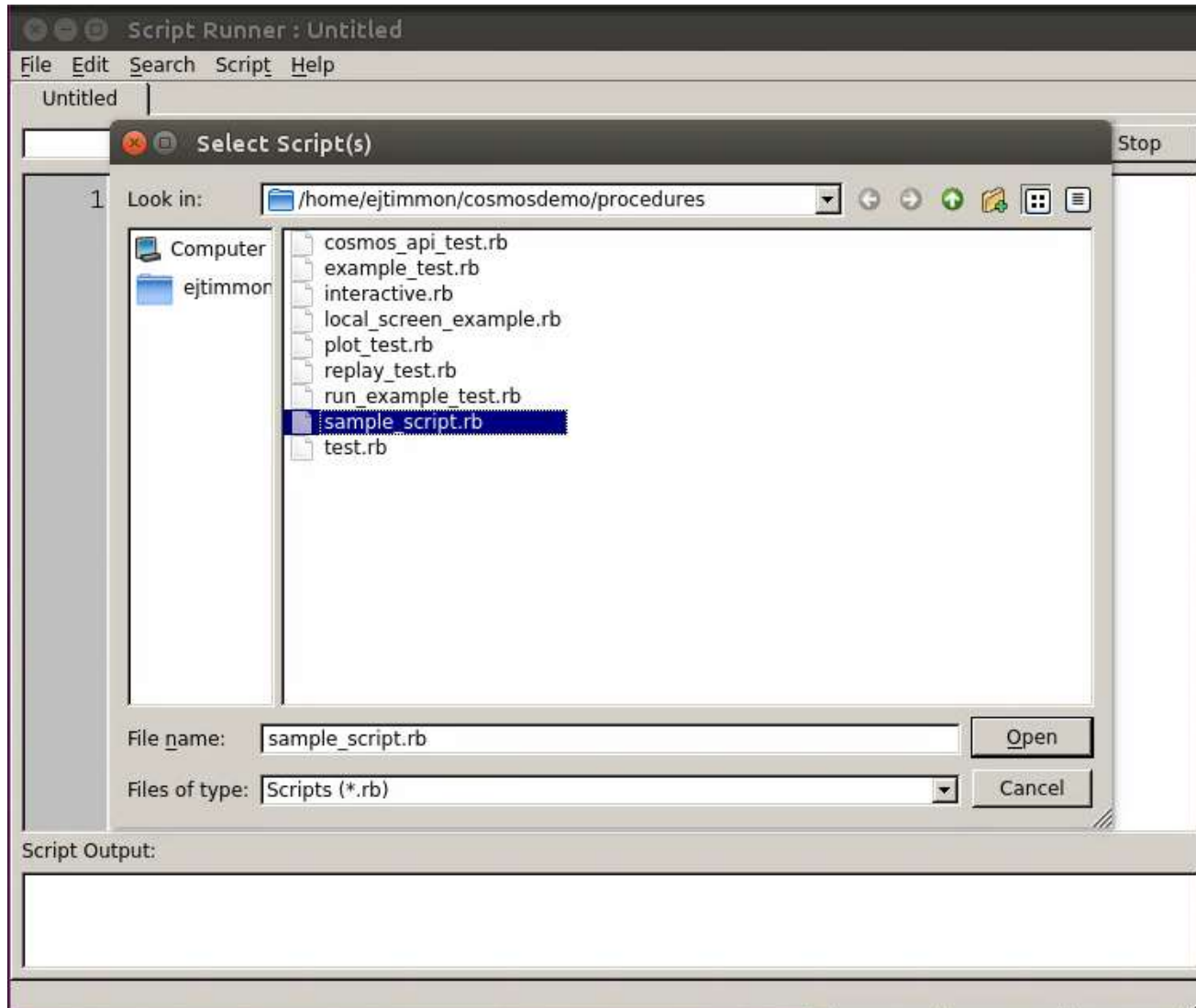


### **Objective: Write and execute a simple script for the sample\_app**

1. Navigate to the cosmos/procedures directory in COSMOS
2. Create a file called sample\_script.rb
3. Write a script that sends a sample\_app no-op command and receives the housekeeping telemetry
4. Enter the main Cosmos directory and launch COSMOS with “ruby Launcher”
5. On the Launcher window, click on “Command and Telemetry Server” and click “OK” on the dialog that pops up
6. On the Launcher window, click on “Script Runner”
7. Click “File” → “Open”, navigate to the procedures directory and then select “sample\_script.rb”
8. When the script loads, click “Start” on the script runner window



# Exercise 5 - Recap





# Exercise 5 - Recap



Script Runner : /home/ejtimmon/cosmosdemo/procedures/sample\_script.rb

File Edit Search Script Help

sample\_script.rb

Stopped

```
1 prompt("Sending enable telemetry")
2 cmd("TO_LAB TO_LAB_ENABLE with DEST_IP 127.0.0.1")
3
4 value = combo_box("Ready to send a No-Op?", 'Yes', 'No')
5 case value
6 when 'Yes'
7     cmd("SAMPLE", "SAMPLE_NOOP")
8     tval = tlm("SAMPLE SAMPLE_HK CMD_CNT")
9     wait (2)
10    prompt("# commands " + tval.to_s)
11 when 'No'
12     alert("Exiting")
13 end
14
```

Script Output:



National Aeronautics and Space Administration



# Test Runner



# Test Runner

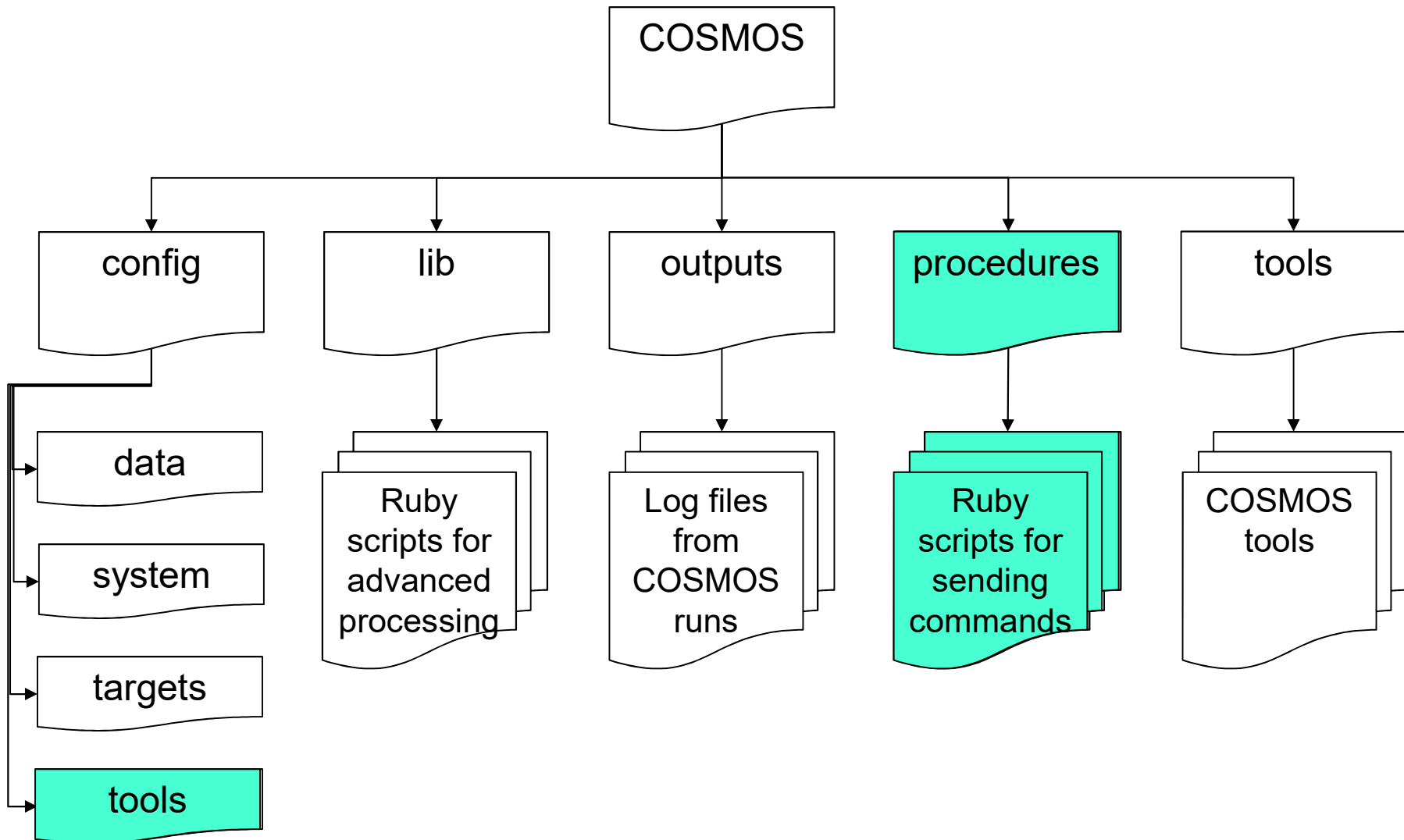


- **The COSMOS test runner builds on the scripting capability to create organized, repeatable test suites**
  - Useful for functional, system, and regression testing
- **Provides a pass/fail test summary and detailed test logs**
- **Tests can be organized and run by “test case”, “test group”, and “test suite”**
  - A **Test Case** is a single test.
  - A **Test Group** is a collection of related tests cases.
  - A **Test Suite** is a collection of test cases and/or test groups.





# Location of Test Runner





## Exercise 6



### **Objective: Write a simple test for the sample\_app**

*This will adapt the simple Ruby script from Exercise 4 into a COSMOS test suite and test case*

1. Navigate to the cosmos/procedures directory in COSMOS
2. Create a file called sample\_test.rb
3. Write a test that sends a sample\_app no-op command, receives the housekeeping telemetry, and verifies that the sample\_app command counter incremented by 1
  - This script should have a test suite and a test case
  - Look at example\_text.rb for an example of syntax
4. Open the file “config/tools/test\_runner/test\_runner.txt” and add the line “LOAD\_UTILITY ‘sample\_test’”
5. Enter the main Cosmos directory and launch COSMOS with “ruby Launcher”
6. On the Launcher window, click on “Command and Telemetry Server” and click “OK” on the dialog that pops up
7. On the Launcher window, click on “Test Runner”



## Exercise 6 Continued



8. Select “SampleTestSuite” from the drop down list beside “Test Suite”
  - The name might be different depending on your exact test script
  - The “Test Group” and “Test case” fields should auto-populate
9. Click on “Start” next to the Test Suite
10. Optionally update the “OPERATOR\_NAME” field in the dialog that appears and click “Start Test”
  - At this point the test should run, showing the real-time execution of the script in the Test Runner window
  - At the end of the test, a “Results” window will appear with a summary of the tests that passed and failed
  - A detailed output from the script can be found in the “Script Output” panel at the bottom of the Test Runner window



# Exercise 6 - Recap



## test\_runner.txt

```
ejtimmon@gs580s-582cfs: ~/cosmosdemo
File Edit View Search Terminal Help
1 LOAD_UTILITY 'example_test'
2 LOAD_UTILITY 'sample_test'
3 LINE_DELAY 0
4 # RESULTS_WRITER 'results_writer.rb'
5 ALLOW_DEBUG
6 PAUSE_ON_ERROR TRUE
7 CONTINUE_TEST_CASE_AFTER_ERROR TRUE
8 ABORT_TESTING_AFTER_ERROR FALSE
9 MANUAL TRUE
10 LOOP_TESTING FALSE
11 BREAK_LOOP_AFTER_ERROR FALSE
12 CREATE_DATA_PACKAGE
13 AUTO_CYCLE_LOGS
14 COLLECT_METADATA
15 # DISABLE_TEST_SUITE_START
16 # DISABLE_TEST_GROUP_START
17 # IGNORE_TEST_SUITE EmptyTestSuite
18 # IGNORE_TEST EmptyTest
19 # MONITOR_LIMITS
20 PAUSE_ON_RED
~
~
~
"config/tools/test_runner/test_runner.txt" 20L, 463C 20,1 All
```



# Exercise 6 - Recap



## sample\_test.txt

Creates a Test Group to which individual test cases can be added

Creates a single Test Case.  
*Note that test cases must start with "test"*

Creates a Test Suite and adds one Test Group to it

```
ejtimmon@gs580s-582cfs: ~/cosmosdemo
File Edit View Search Terminal Help
1 load 'cosmos/tools/test_runner/test.rb'
2
3 class SampleCmdTest < Cosmos::Test
4
5   def initialize
6     super()
7   end
8
9   def test_sample_noop
10    puts "Running #{Cosmos::Test.current_test_suite}: #{Cosmos::Test.current_test}
11    cmd("TO_LAB TO_LAB_ENABLE with DEST_IP 127.0.0.1")
12
13    puts "Getting initial command count"
14    wait(2)
15    initcnt = tlm("SAMPLE SAMPLE_HK CMD_CNT")
16    wait(1)
17
18    puts "Sending no-op command"
19    cmd("SAMPLE", "SAMPLE_NOOP")
20    wait(1)
21
22    expcnt = initcnt + 1
23    wait_check("SAMPLE SAMPLE_HK CMD_CNT == #{expcnt}", 5)
24
25  end
26 end
27
28
29 class SampleTestSuite < Cosmos::TestSuite
30   def initialize
31     super()
32     add_test('SampleCmdTest')
33   end
34 end
```

34,4 All



# Exercise 6 – Recap



Test Runner

File Edit Script Help

Pause on Error     Manual

Continue Test Case after Error     Loop Testing

Abort Testing after Error     Break Loop after Error

Test Suite: SampleTestSuite    Start    Setup    Teardown

Test Group: SampleCmdTest    Start    Setup    Teardown

Test Case: test sample noop    Start

Executing Test Case:    Pass: 1    Skip: 0    Fail: 0    0%

Stopped    Start    Pause    Stop

```
1 TestRunner.start(SampleTestSuite)
```

Enter Test Metadata

Page 1

COSMOS\_VERSION 4.4.2

USER\_VERSION Unofficial

RUBY\_VERSION 2.5.8p224

OPERATOR\_NAME Unspecified

Start Test    Cancel Test

Script Output:

```
2020/07/27 11:36:50.145 (sample_test.rb:20): WAIT: 1 seconds with actual time of 1.017711291 seconds
2020/07/27 11:36:50.234 (sample_test.rb:23): CHECK: SAMPLE SAMPLE_HK_CMD_CNT == 1 success with value == 1 after waiting 0.00292465 seconds
2020/07/27 11:36:50.257 (SCRIPTRUNNER): Script completed: SampleTestSuite
```



# Exercise 6 - Recap



The screenshot shows the Test Runner application interface. A 'Results' dialog box is open, displaying the following information:

- Report Filename: /home/ejtimmon/cosmosdemo/outputs/logs/2020\_07\_27
- Detailed Test Output Logged to: /home/ejtimmon/cosmosdemo/outputs/lo
- Metadata:
  - COSMOS\_VERSION = 4.4.2
  - USER\_VERSION = Unofficial
  - RUBY\_VERSION = 2.5.8p224
  - OPERATOR\_NAME = Unspecified
- Settings:
  - Pause on Error = true
  - Continue Test Case after Error = true
  - Abort Testing after Error = false
  - Manual = true
  - Loop Testing = false
  - Break Loop after Error = false
- Results:
  - 2020/07/27 11:36:45.598: Executing SampleTestSuite
  - 2020/07/27 11:36:50.246: SampleCmdTest:test\_sample\_noop:PASS
  - 2020/07/27 11:36:50.307: Completed SampleTestSuite
- Test Summary ---
- Run Time : 4.71 seconds
- Total Tests : 1
- Pass : 1
- Skip : 0
- Fail : 0

At the bottom of the dialog box, there is an 'OK' button. A red arrow points to the 'OK' button. Another red arrow points to the 'Abort Testing after Error' checkbox in the Test Runner interface.