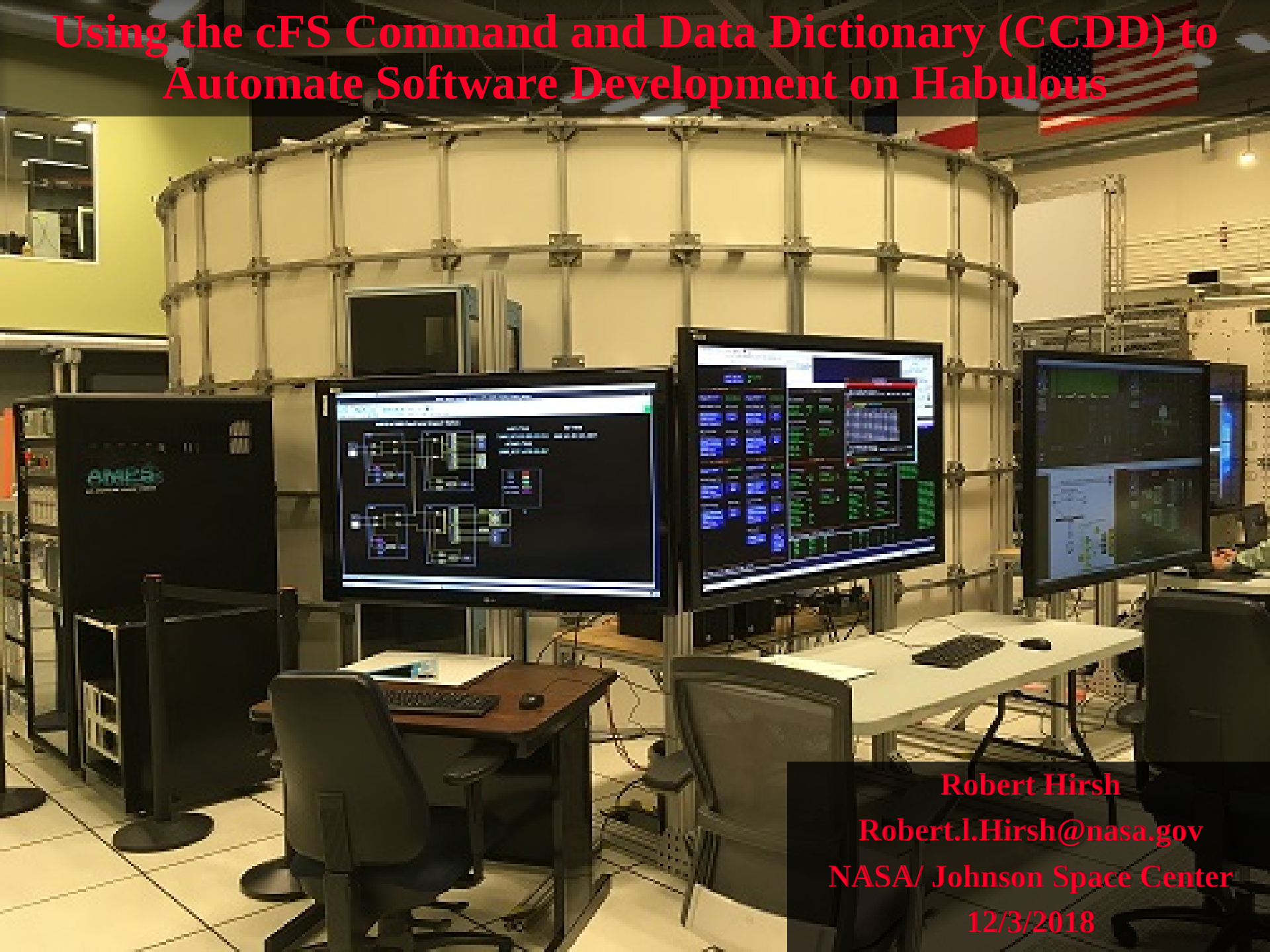
# Using the cFS Command and Data Dictionary (CCDD) to Automate Software Development on Habulous



Robert Hirsh
Robert.l.Hirsh@nasa.gov
NASA/ Johnson Space Center
12/3/2018

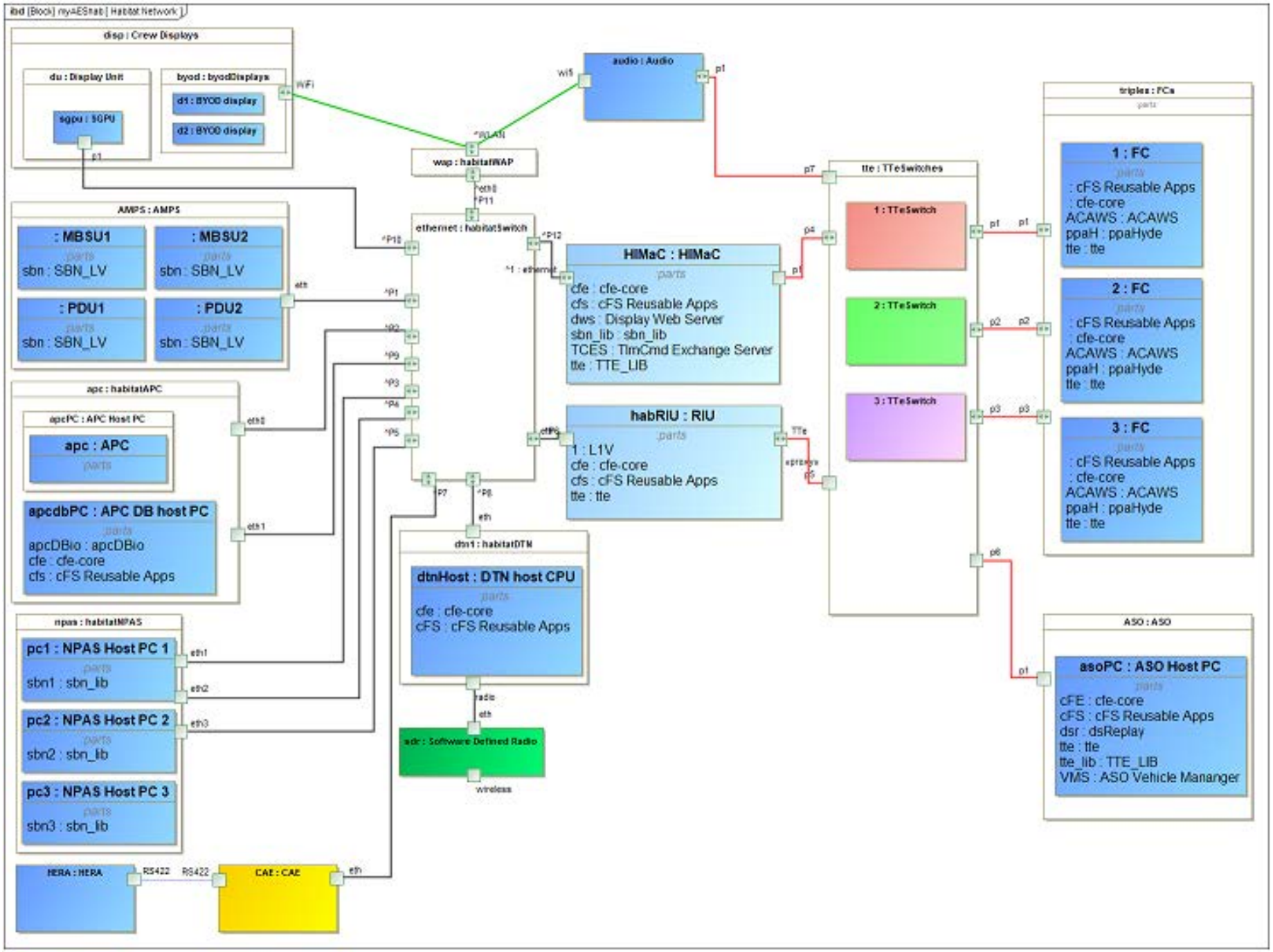
# Agenda



- Habulous Background
- CCDD Overview
- CCDD Products used on Habulous
  - C header files that define all software bus commands/telemetry messages
  - Generating file defining the Message ID's used (cfs_msgids.h )
  - XML Telemetry and Command Exchange (XTCE) files (displays)
  - "Protobetter" code (to manage different endian-ness/architectures)
- Development on Habulous
  - CCSDS_v2 extended headers
  - Extending/customizing SBN to pass messages among computers on multiple networks
  - Using SBN_lib to allow non-cFS node to communicate with cFS nodes
- Next Steps
  - Developing TTE network and schedule tables for all the various CPUs to use

- The Habulous project is an Earth-based testbed (HW/SW)
  - Prototyping future space habitat unit and technologies
  - Representation from various NASA centers and aerospace organizations
    - » ARC/JSC/GRC/Goddard/Stennis
  - Distributed nature of the team makes data interfaces especially critical
    - » Massively heterogeneous computer architectures and operating systems
      - *– 32/64-bit, Big/Little Endian, Linux/VxWorks/Windows, x86/PPC/RaspberryPi*
  - Multiple CPUs use the SBN application to communicate
    - » Most CPUs run cFS (use SBN app and Protobetter)
    - » Non-cFS CPU (use SBN_lib with Prototbetter)

# Habulous Block Diagram



# CCDD Background



- CCDD stands for cFS Command and Data Dictionary
- Goddard's Core Flight System (cFS) has been, is, and is intended to be used by many projects
  - Examples: Lunar Reconnaissance Orbiter (LRO), Morpheus, Exploration EMU (xEMU) spacesuit, Orion Backup Flight Software (BFS)
  - Success of the cFS concept is shown by the number cFS projects at FSW-2018
- A command and data dictionary (CDD) defines telemetry/command messages
- Each cFS project must select a way to manage their CDD
  - Frequently involves using a spreadsheet, with custom SW to convert into useful files
- cFS Command and Data Dictionary utility (CCDD) was designed as a generic utility to eliminate duplication of effort in order to make CDD management easier

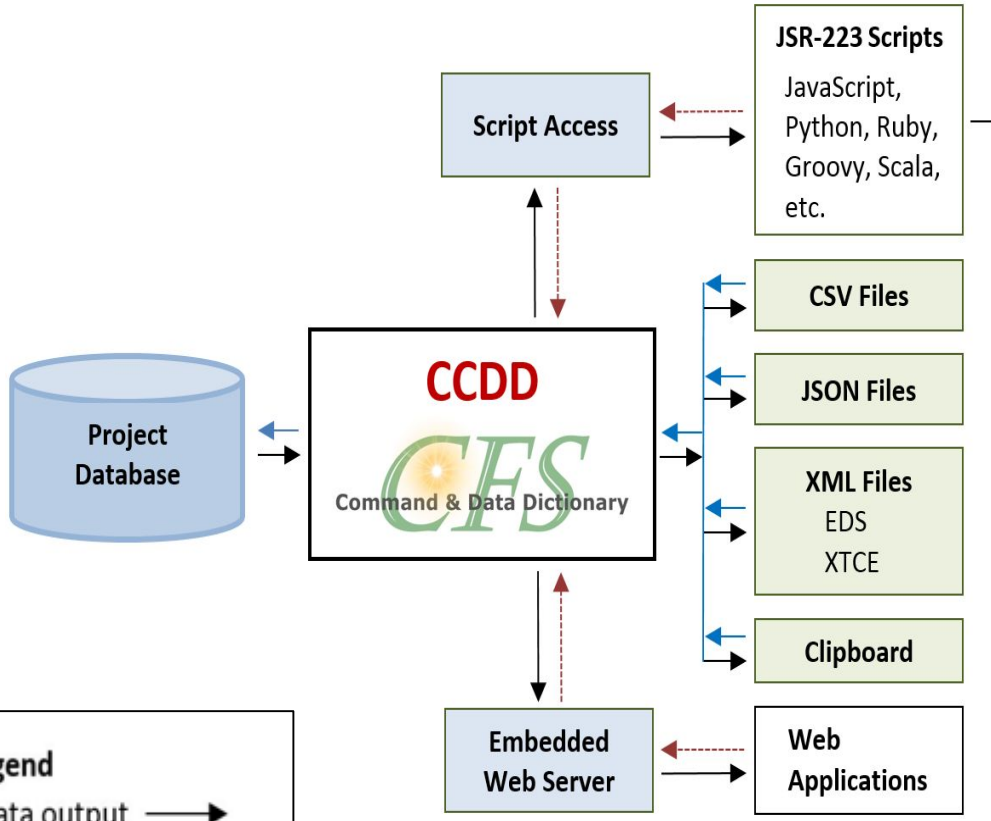
# CCDD Goals



- Create a configurable CDD utility that runs on multiple operating systems
  - Written in Java for maximum portability
- Easy creation/modification of CDD information
  - Graphical user interface (GUI) to interact with the database
- Store all CDD information into a standard database (postgreSQL)
- Bidirectional transfer of information to/from the CCDD
  - Cut-n-paste to Excel, import/export via XTCE/CSV/JSON
- Easy access to CDD information (via scripting languages and web applications )
  - Allows user to code in various languages (ruby/python/js) and access CDD information
    - »Create vehicle and ground software products, data summary, etc
    - »Generate complicated CFS products: Schedule or network tables, copy table, etc

Data is accessible to scripting languages (JavaScript, Python, etc.)

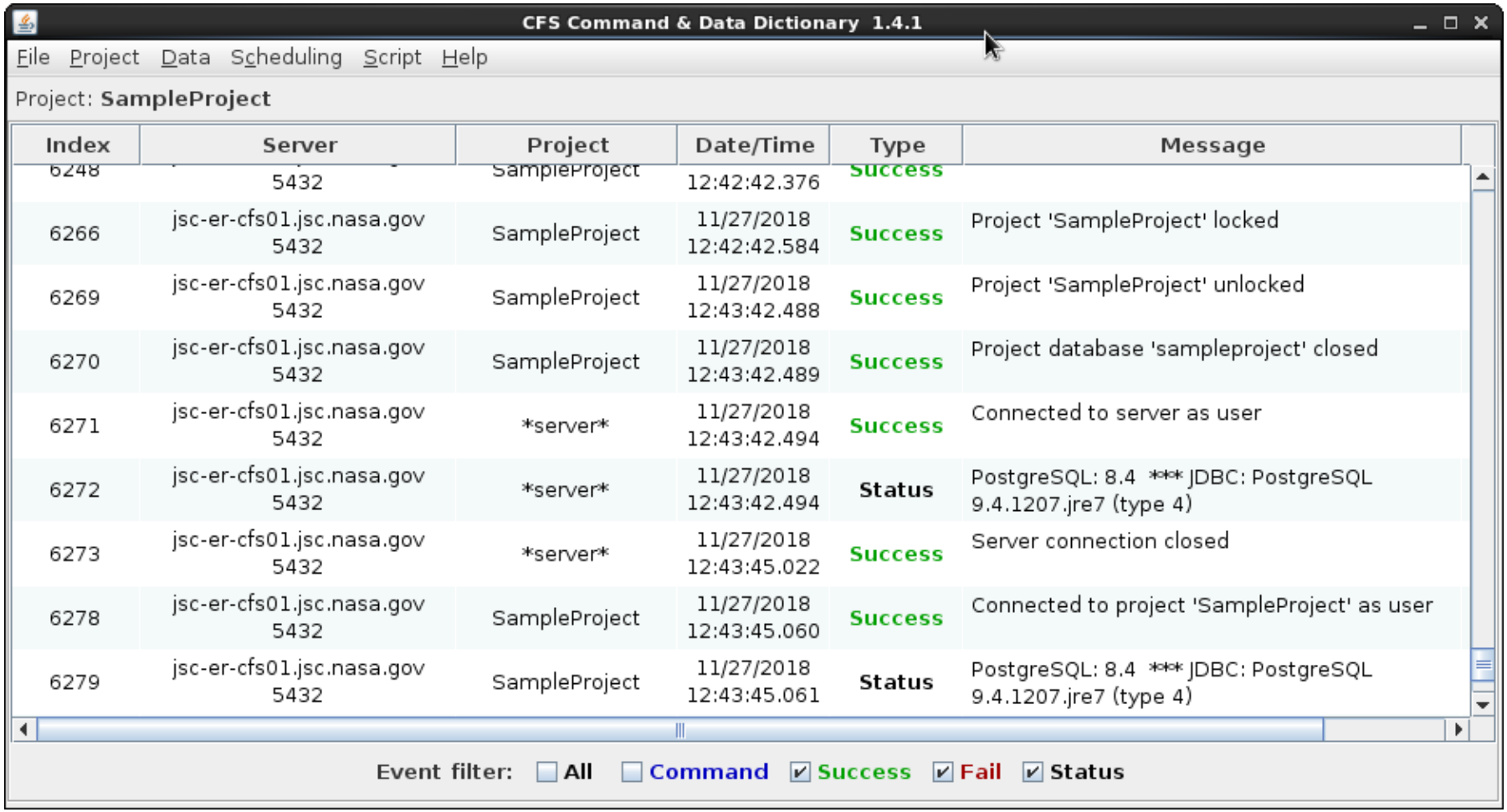
- Example scripts provided for common products



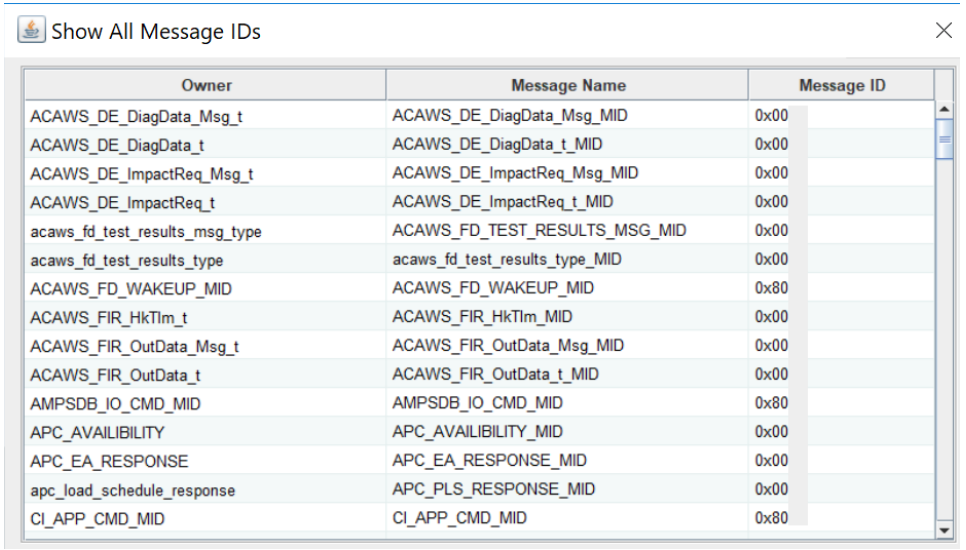

Web-based dataserver (JSON)

Imported/exported via:

- CSV (comma-separated values)
- JSON (JavaScript Object Notation)
- XML (Extensible Markup Language)
  - » EDS (Electronic Data Sheet)
  - » XTCE (XML Telemetric and Command Exchange)
- OS clipboard (“cut & paste”)

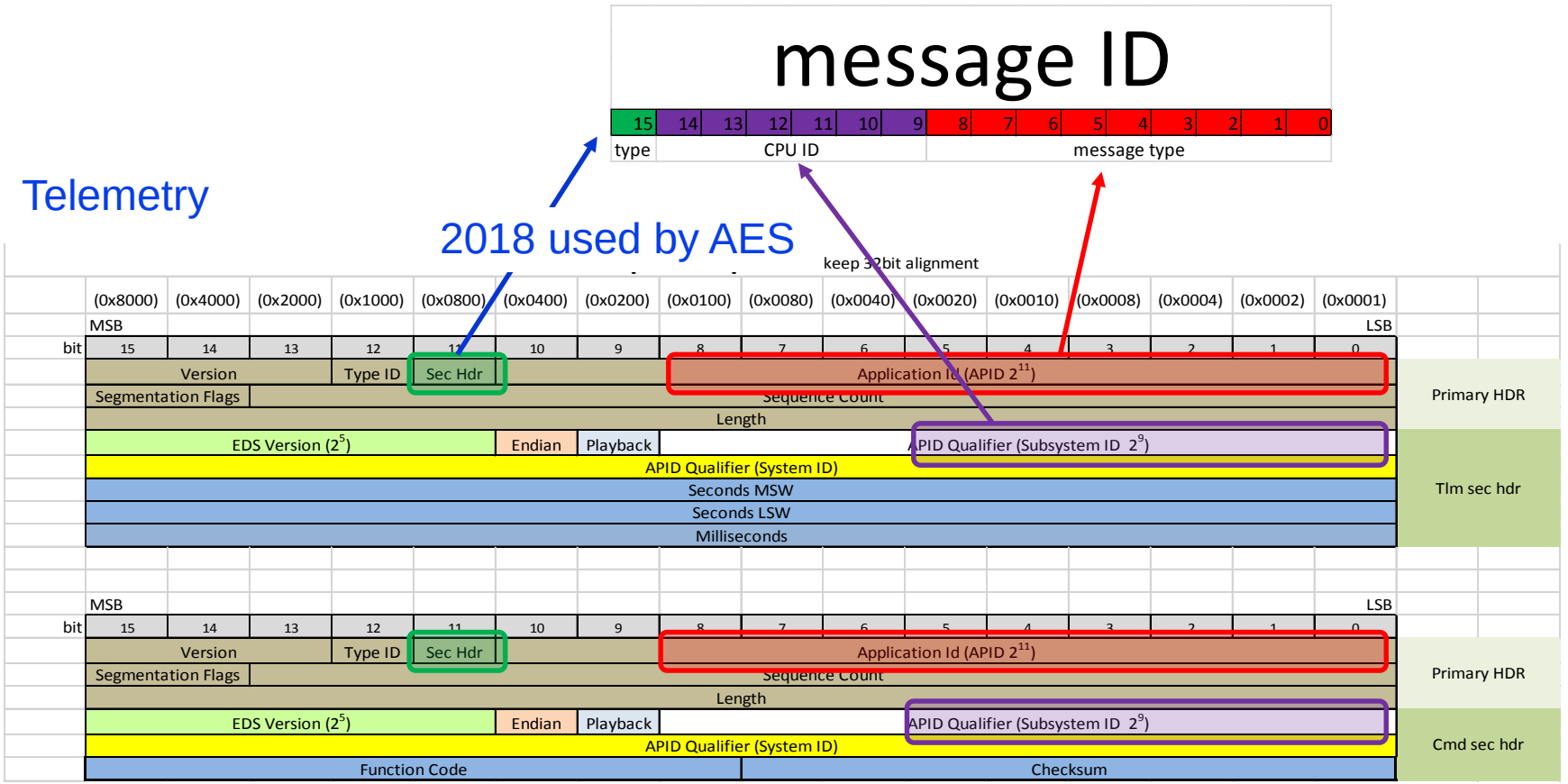

CFS Command & Data Dictionary 1.4.1
File Project Data Scheduling Script Help
Project: SampleProject
Index | Server | Project | Date/Time | Type | Message
6248 | 5432 | SampleProject | 12:42:42.376 | Success |
6266 | jsc-er-cfs01.jsc.nasa.gov 5432 | SampleProject | 11/27/2018 12:42:42.584 | Success | Project 'SampleProject' locked
6269 | jsc-er-cfs01.jsc.nasa.gov 5432 | SampleProject | 11/27/2018 12:43:42.488 | Success | Project 'SampleProject' unlocked
6270 | jsc-er-cfs01.jsc.nasa.gov 5432 | SampleProject | 11/27/2018 12:43:42.489 | Success | Project database 'sampleproject' closed
6271 | jsc-er-cfs01.jsc.nasa.gov 5432 | *server* | 11/27/2018 12:43:42.494 | Success | Connected to server as user
6272 | jsc-er-cfs01.jsc.nasa.gov 5432 | *server* | 11/27/2018 12:43:42.494 | Status | PostgreSQL: 8.4 *** JDBC: PostgreSQL 9.4.1207.jre7 (type 4)
6273 | jsc-er-cfs01.jsc.nasa.gov 5432 | *server* | 11/27/2018 12:43:45.022 | Success | Server connection closed
6278 | jsc-er-cfs01.jsc.nasa.gov 5432 | SampleProject | 11/27/2018 12:43:45.060 | Success | Connected to project 'SampleProject' as user
6279 | jsc-er-cfs01.jsc.nasa.gov 5432 | SampleProject | 11/27/2018 12:43:45.061 | Status | PostgreSQL: 8.4 *** JDBC: PostgreSQL 9.4.1207.jre7 (type 4)
Event filter: All Command Success Fail Status

- cfs_msgids.h file generation
  - Same file compiled by all CPUs
  - Defines all the MIDs for each cFS message sent/received on any of the various CPUs
  - Using CCSDSv2, so each MID is a combination of APID/SystemID/SubSystemID




| Owner | Message Name | Message ID |
|---|---|---|
| ACAWS_DE_DiagData_Msg_t | ACAWS_DE_DiagData_Msg_MID | 0x00 |
| ACAWS_DE_DiagData_t | ACAWS_DE_DiagData_t_MID | 0x00 |
| ACAWS_DE_ImpactReq_Msg_t | ACAWS_DE_ImpactReq_Msg_MID | 0x00 |
| ACAWS_DE_ImpactReq_t | ACAWS_DE_ImpactReq_t_MID | 0x00 |
| acaws_fd_test_results_msg_type | ACAWS_FD_TEST_RESULTS_MSG_MID | 0x00 |
| acaws_fd_test_results_type | acaws_fd_test_results_type_MID | 0x00 |
| ACAWS_FD_WAKEUP_MID | ACAWS_FD_WAKEUP_MID | 0x80 |
| ACAWS_FIR_HkTlm_t | ACAWS_FIR_HkTlm_MID | 0x00 |
| ACAWS_FIR_OutData_Msg_t | ACAWS_FIR_OutData_Msg_MID | 0x00 |
| ACAWS_FIR_OutData_t | ACAWS_FIR_OutData_t_MID | 0x00 |
| AMPSDB_IO_CMD_MID | AMPSDB_IO_CMD_MID | 0x80 |
| APC_AVAILIBILITY | APC_AVAILIBILITY_MID | 0x00 |
| APC_EA_RESPONSE | APC_EA_RESPONSE_MID | 0x00 |
| apc_load_schedule_response | APC_PLS_RESPONSE_MID | 0x00 |
| CI_APP_CMD_MID | CI_APP_CMD_MID | 0x80 |

- Using the CCDD information to automatically generate the C-header files
  - Define the structure for all software bus (SB) commands/telemetry messages
- Generate XML Telemetry and Command Exchange (XTCE) files
  - Used by display team to make displays for any CPU
- Generating “Protobetter” code for communication with other CPUs
  - Manages packing and different endian-ness/architectures


message ID
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
type
CPU ID
message type
Telemetry
2018 used by AES
keep 32bit alignment
(0x8000) (0x4000) (0x2000) (0x1000) (0x0800) (0x0400) (0x0200) (0x0100) (0x0080) (0x0040) (0x0020) (0x0010) (0x0008) (0x0004) (0x0002) (0x0001)
MSB
LSB
bit
Version
Type ID
Sec Hdr
Application Id (APID 2^11)
Segmentation Flags
Sequence Count
Primary HDR
Length
EDS Version (2^5)
Endian
Playback
APID Qualifier (Subsystem ID 2^9)
APID Qualifier (System ID)
Seconds MSW
Seconds LSW
Milliseconds
Tlm sec hdr
Function Code
Checksum
Cmd sec hdr
Command

- Updating to CCSDS_v2 (and using CPU# as subsystem ID)
  - Running out of room for unique MIDs on all CPUs for the 11-bits of version 1
  - See next slide
- Exporting XTCE files to allow drag-n-drop display development for all CPUs
- Extending/customizing SBN to pass messages to computers
  - Computers with multiple interfaces act as a “bridge” to CPUs that can’t talk directly
  - “Protobetter” developed to manage packing/endian differences
- Using SBN_lib to allow non-cFS node to communicate with cFS nodes
  - Allows non-cFS nodes to “impersonate” a cFS node and talk to SBN on other CPUs
- Worked to develop the CDD before the SW development was complete
  - Not treat CDD as an “as built” post-development documentation effort
  - Required iterations on data structures and MIDs, but minimized interface issues

# Future Work



- The CCDD tool has successfully been used to automate/autocode a large amount of software used on Habulous
- Working to allow the CCDD to define even more products including
  - Time-triggered Ethernet (TTE) network tables/maps
    - »Coordinate message passing between various synchronized machines
  - cFS schedule table (for each CPU)
  - Automated CCDD to SysML export