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
• 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)
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 (postgresSQL)
• 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

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")

Web-based dataserver (JSON)
## Project: SampleProject

<table>
<thead>
<tr>
<th>Index</th>
<th>Server</th>
<th>Project</th>
<th>Date/Time</th>
<th>Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>6248</td>
<td>5432</td>
<td>SampleProject</td>
<td>12:42:42.376</td>
<td>Success</td>
<td>Project 'SampleProject' locked</td>
</tr>
<tr>
<td>6266</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:42:42.584</td>
<td>Success</td>
<td>Project 'SampleProject' unlocked</td>
</tr>
<tr>
<td>6269</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:42:42.488</td>
<td>Success</td>
<td>Project database 'sampleproject' closed</td>
</tr>
<tr>
<td>6270</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:43:42.489</td>
<td>Success</td>
<td>Connected to server as user</td>
</tr>
<tr>
<td>6271</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td><em>server</em></td>
<td>11/27/2018 12:43:42.494</td>
<td>Success</td>
<td>Server connection closed</td>
</tr>
<tr>
<td>6272</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td><em>server</em></td>
<td>11/27/2018 12:43:42.494</td>
<td>Status</td>
<td>PostgreSQL: 8.4 *** JDBC: PostgreSQL 9.4.1207.jre7 (type 4)</td>
</tr>
<tr>
<td>6273</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td><em>server</em></td>
<td>11/27/2018 12:43:45.022</td>
<td>Success</td>
<td>Connected to project 'SampleProject' as user</td>
</tr>
<tr>
<td>6278</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:43:45.060</td>
<td>Success</td>
<td>Postgres: 8.4 *** JDBC: PostgreSQL 9.4.1207.jre7 (type 4)</td>
</tr>
<tr>
<td>6279</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:43:45.061</td>
<td>Status</td>
<td>Postgres: 8.4 *** JDBC: PostgreSQL 9.4.1207.jre7 (type 4)</td>
</tr>
</tbody>
</table>
• **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

• 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
Habulous MID definition

Telemetry

2018 used by AES

Command

keep 32-bit alignment

Type ID Sec Hdr

MSB LSB

bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Version Type ID Sec Hdr

Segmentation Flags

EDS Version (2^5)

Endian Playback

APID Qualifier (System ID)

Milliseconds

APID Qualifier (Subsystem ID 2^5)

Length

APID Qualifier (System ID)

Function Code

Seconds LSW

Checksum

Application Id (APID 2^{15})

Seconds MSW
• 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

![Diagram](image)