

Adapting a Large-Scale Multi-Mission Ground System for Low-Cost CubeSats

William L. Quach*, Lloyd R. DeForrest†, Andrew T. Klesh‡, Joshua B. Schoolcraft§
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

The majority of today’s CubeSat fleet consists of Earth-orbiting missions that mostly use existing ground systems developed by universities because of availability, simplicity, and low-cost. The Interplanetary NanoSpacecraft Pathfinder In Relevant Environment (INSPIRE) mission is a revolutionary CubeSat mission that will launch a pair of CubeSats into deep space to study the feasibility of CubeSats beyond low-Earth orbit. This uncovers a new set of systems and software engineering challenges to the development of a robust and reliable ground system in a low-cost environment. In this paper, we discuss the approach to these challenges by using the Jet Propulsion Laboratory’s (JPL) Advanced Multimission Operation System (AMMOS) Ground Data System (GDS) as well as the methodologies used to engineer the flight system to work with an existing ground system developed for large-scale missions. Specifically we will focus on the command and telemetry subsystem of AMMOS, the Multimission Data Processing and Control System. We conclude with a retrospective on the challenges encountered and a brief discussion on our efforts to provide AMMOS to support future deep space CubeSat missions.

I. Introduction

THE *Interplanetary NanoSpacecraft Pathfinder In Relevant Environment* (INSPIRE) mission’s planned trek into deep space provides unique challenges that only multi-million dollar, government-backed missions have overcome. The nature of this mission eliminates most, if not all, of the existing CubeSat ground systems used for command and telemetry because of the need to communicate with NASA’s Deep Space Network (DSN). The use of the DSN requires the spacecraft to communicate with the ground command and telemetry software using Consultative Committee for Space Data Systems (CCSDS) communication protocols. INSPIRE has chosen to use the AMMOS Multimission Data Processing and Control System (AMPCS) to command and monitor their spacecraft. AMPCS is a reusable, multimission ground data processing, archival, visualization, and command system used for spacecraft testing and mission operations. AMPCS is being developed by the Multimission Ground Systems and Services (MGSS) organization of JPL and is being used by several NASA missions, including the flagship Mars Science Laboratory (MSL), the Soil Moisture Active Passive (SMAP) Earth orbiter, and the Interior Exploration using Seismic Investigations, Geodesy and Heat Transport (InSight) Mars lander. As the first mission to send CubeSats beyond low-Earth orbit, INSPIRE is seeking to leverage the millions of dollars already spent on the research, development, and testing of AMPCS in order to bring value to the mission while lowering risks and costs. JPL’s MGSS is spearheading the effort to provide the INSPIRE mission with an existing, proven, and reliable Ground Data System (GDS) in a low-cost manner.

II. Background

CubeSats projects, flown by students, universities, institutions, and governments around the world have used a myriad of ground system options for communication and control. Many projects create a home-grown solution, based loosely upon amateur radio packet service protocols, allowing for many stations to receive data, which then is transmitted to the host institution.¹ Satellites approved for frequencies also used by the amateur radio community

* MGSS Ground Software Engineer, Mission Control Systems, 4800 Oak Grove Drive, M/S: 301-480.

† Technical Group Supervisor, Mission Control Systems, 4800 Oak Grove Drive, M/S: 301-480.

‡ INSPIRE Principal Investigator, Planetary Mission Formulation, 4800 Oak Grove Drive, M/S: 301-165.

§ INSPIRE Flight Software Engineer, Instrument Flight Software and GSE, 4800 Oak Grove Drive, M/S: 238-420.

have found significant benefit from this kind of partnership – the project receives data, while the amateur community has both assets to communicate with, and additional radio frequency interference measurements (a side product of recording Received Signal Strength Indicator data).² To better utilize a shared network, several common GDSs have been proposed, and at least partially implemented, such as the GENSO, Mercury, and the RAX networks. Commercial small spacecraft have also developed multi-station solutions, such as SkyBox, LLC. Several government spacecraft now make use of the MC3 network³, perhaps one of the few other multi-mission systems to be adapted from large spacecraft to small. While typical university missions fold one version of the ground data system from one CubeSat to the next, student turnover, limited documentation, and limited verification (and testing) have made single institution systems often locked to that institution’s flight software.

INSPIRE is operating far beyond the distance to the Moon and this significant increase in telecommunications range makes the existing CubeSat communications infrastructure unable to meet the mission’s needs. Instead of the usual UHF stations, INSPIRE will use NASA’s 34m DSN stations with deep space X-band frequencies. The interfaces and capabilities of the DSN drive the mission to use different GDS software from traditional CubeSat mission. Other CubeSat missions operating in deep space, or requiring high data rates in LEO, would also benefit from using NASA tracking assets and stations. AMPCS provides an interface to these NASA stations and a complete suite of tools for telemetry processing, monitoring, analysis, archival, and spacecraft commanding.

III. AMPCS: JPL’s Command and Telemetry System

A. AMPCS System Overview

1. History

A new generation of spacecraft telemetry processing and command management components has been developed for JPL’s Mars Science Laboratory (MSL) using modern software engineering techniques. Before MSL, JPL missions stored spacecraft telemetry in its raw form (frames and packets). Whenever users needed access to spacecraft telemetry products, the raw telemetry must be processed into the discrete values. Additionally, most GDS components before MSL were developed in the C programming language. Starting with MSL, the GDS now stores both raw and processed telemetry in a Life Of Mission (LOM) database and all components of AMPCS are developed using the Java programming language. These new Java based command and telemetry components provide telemetry consumers with readily available spacecraft information accessible through modern software interfaces. This architecture enables new classes of telemetry consumer applications to be built without affecting the AMPCS architecture⁴.

2. Nomenclature

The nomenclature used in this section is defined below:

- **Channels, Channel Data, Channelized Telemetry:** Refers to the sensor values sent down by the FSW. Each flight component’s sensor is allocated a “channel” with a unique ID, usually in the form of a [stem]-[number]. The [stem] could be a letter representing a flight module or an abbreviated flight module name, such as T-0001 or THRM-0001, respectively. Channel data are collected by the FSW and packaged as packet payload for transport.
- **Event Records (EVR):** Event records are messages from the FSW indicating the occurrence of a certain event. Events are identified by a unique (to the mission) event ID, which is defined in an EVR dictionary. Event records are associated with a criticality level (e.g. ACTIVITY_HI, WARNING_LO, FATAL, etc.) and may contain additional metadata to further describe the event. Event records are collected by the FSW and a packaged as packet payload for transport.
- **Frame:** A frame is a fixed-length transport mechanism used in the CCSDS standard. Frames usually contain packets generated by the FSW.
- **Packet:** A packet is a variable-length transport mechanism used in the CCSDS standard. Packets usually contain channelized data, EVRs, or data products generated by the FSW.
- **Data Product:** Any collection of data generated onboard the spacecraft. Data products are “files” in the FSW and may contain anything from channelized data to science data from specific instruments. They are packaged as packet payload for transport and may be processed by AMPCS or a tool downstream from AMPCS, depending on the packet contents.

3. *Driving Principles of the AMPCS*

AMPCS was developed with the following guiding principles in mind:

- **Modularity/Scalability** – AMPCS can be executed in a single laptop computer or in numerous Virtual Machine (VM) instances, depending on system performance requirements.
- **AMPCS as the mission’s “Gold-Standard” telemetry provider** – AMPCS is the telemetry provider for GDS users. Users make requests to the AMPCS repository through common, well documented interfaces for access to LOM storage of spacecraft telemetry. All spacecraft telemetry products, including Channelized Data (also called spacecraft engineering data), EVRs, science and engineering products, command logs, etc., are available through standard AMPCS interfaces.
- **Support for Multiple Missions** –AMPCS is designed to be used on all current and future JPL and any other CCSDS compliant missions. XML is used extensively in the configuration of AMPCS and to control AMPCS behavior, thus greatly reducing the cost to of AMPCS adaptation to new mission requirements.
- **Event Driven Operations** – AMPCS makes heavy use of a message bus to provide notification of spacecraft events to AMPCS users, enabling event driven operations for increased levels of automation and lower operations costs.
- **Test Session Encapsulation** – During spacecraft testing, AMPCS supports the concept of a test “Session”. In AMPCS, the concept of test sessions is used to capture and store all data and related information associated with a test. This includes the input test datasets, telemetry and command dictionaries and other XML configuration files necessary to reproduce that test at any future time. In this way, with a single session ID, any user can re-run that test under identical conditions to the original test.
- **Test As You Fly** - The phrase “Test As You Fly” is used by JPL, NASA and industry to describe a rigorous application of the long standing test principle whereby mission related functions are tested in the most flight like environment and configuration possible.⁵ AMPCS provides specific features that support all phases of a mission’s lifecycle, from mission concept to mission operations, through common interfaces for access to telemetry products. This ensures a mission’s GDS has a strong TAYF story from a telemetry processing and command perspective. The same software, user displays, reports and plots are available in each lifecycle venue, as depicted in Figure 1. This methodology provides an efficient way to develop reliable, proven processes for operations.

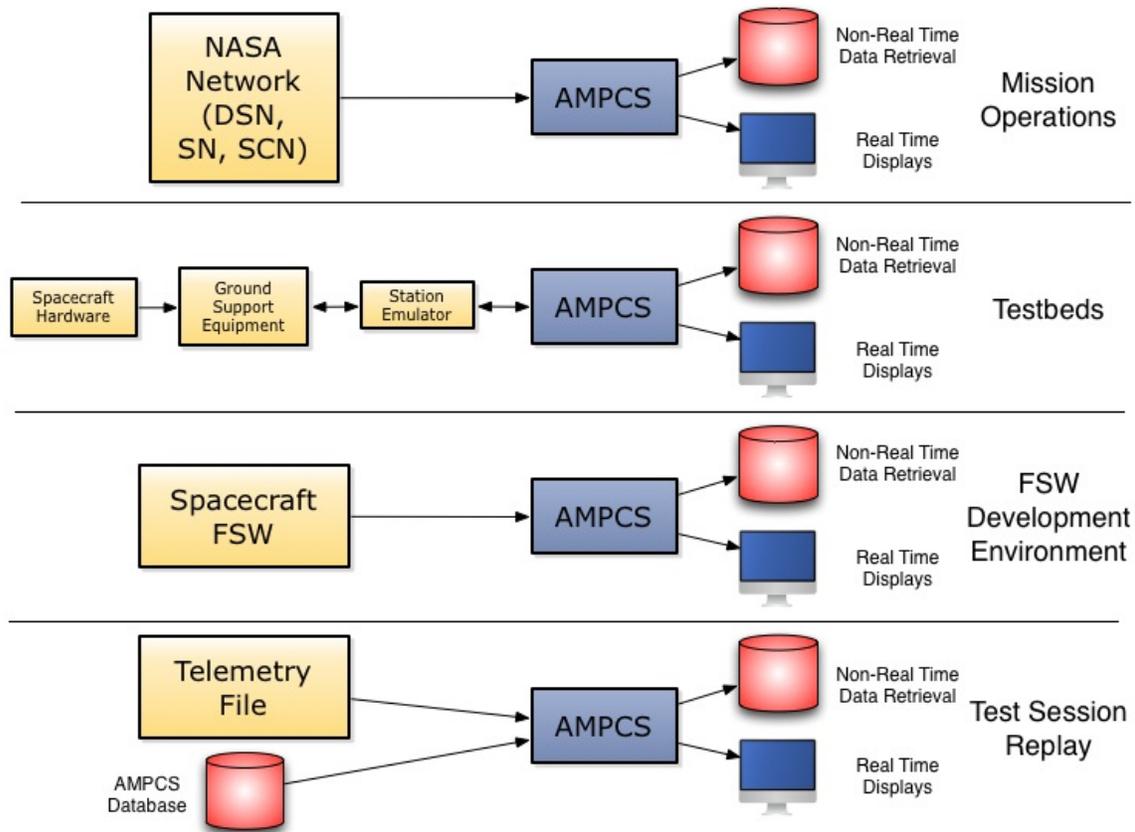


Figure 1. The AMPCS "Test as You Fly" Approach.

B. AMPCS Design Overview

Figure 2 depicts the top level AMPCS design. For Downlink processing, raw telemetry from the spacecraft is received by AMPCS in the form of CCSDS formatted frames or packets into the Downlink Processing block. The Downlink Processing block transforms the raw telemetry into science and engineering data products, channelized data, EVRs, channels in alarm, and log/status messages. These telemetry products are sent to the Message Bus for real time notification of telemetry product availability and to the AMPCS database, along with the original frames and packets for LOM storage. The downlink processor is able to ingest frames or packets from any NASA station, data streams or files, or directly from the AMPCS database.

Real time clients receive notification of telemetry availability from the AMPCS Message Bus. These clients can be real time channel visualization clients, simple message bus listeners, Web clients or any user developed application that needs to ingest real time telemetry products. AMPCS has a built-in, Java based monitor providing extensive telemetry visualization capabilities. Visualization of frame, packet and product quality is also provided.

Non-real time users access the AMPCS database through a standardized set of APIs. AMPCS provides built-in queries and reports to provide users with a base level of capabilities. Spacecraft teams typically develop their own sets of tools that access the AMPCS database through the provided APIs.

For Uplink control, AMPCS provides an interface to the NASA station during mission operations and to the missions testbed during the mission's development phases. During mission operations, uplink flow management is provided, as well as the ability to send single commands, command sets and binary files (new FSW images) to the spacecraft.

AMPCS also provides two levels of automation. During spacecraft testing, session based automation is provided through AMPCS enabled Python scripts. For example, a script can send a command to the spacecraft and wait for a response in the telemetry from the spacecraft before sending the next command. In this way, repeatable test scripts can be developed and used throughout the mission's lifecycle. The other level of automation provided is during

mission operations, when the need for cross-session automation is necessary. This capability enables “Lights Out” automation for a mission.

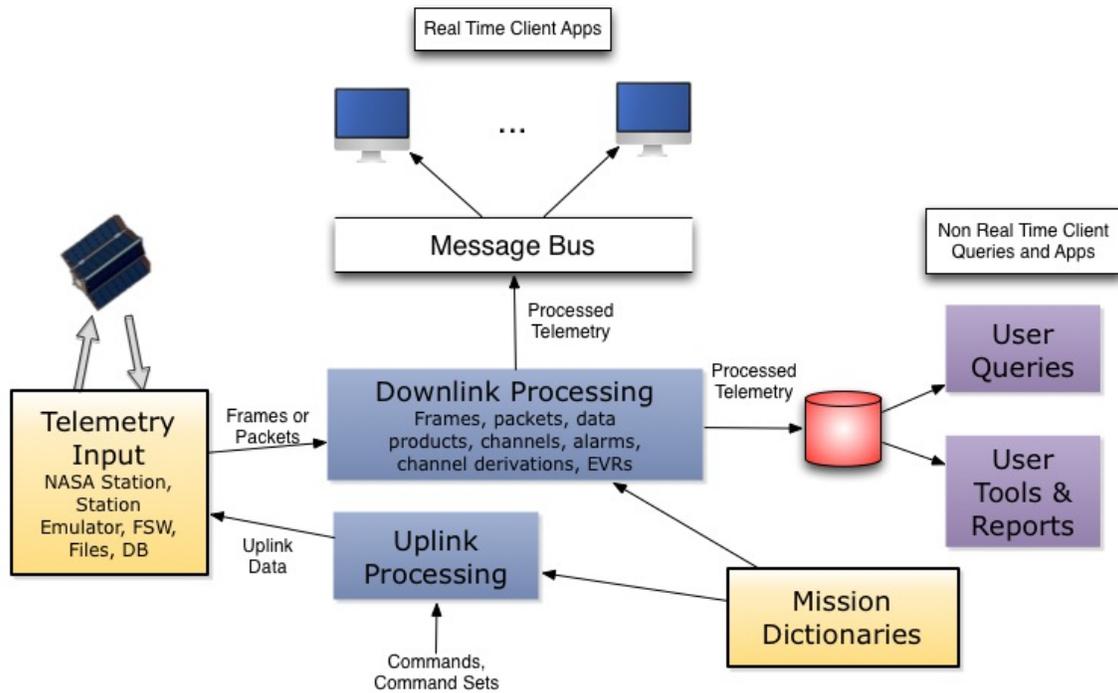


Figure 2. The AMPCS Block Diagram.

C. Key Software Concepts

1. Session

AMPCS is built around a concept of a session. Each run of AMPCS is allocated a new session, with a unique session ID, where the mission’s raw data, processed EVR, channelized data, command uplinks, and system logs are captured. This allows a seamless segregation of test data in the system, providing easy retrieval for post-test analyses. In the operations venue, the concept of sessions is transparent to the user as AMPCS provides methods to retrieve and analyze data across session boundaries.

2. Perspective

AMPCS graphical user interfaces (GUI) are composed in a user-defined perspective. A perspective allows arrangement of various display elements, such as plots and data tables. Missions using AMPCS can predefine various perspectives that are tailored for various roles, providing them with consistent location, size, color, font size, and content of the graphical elements. This consistency accelerates user training, lowers the probability of human errors, and facilitates efficient user operation.

3. Templating

AMPCS command-line interface (CLI) tools provide the ability for the users to define a specific output format using a template. This not only allows customized formatting of the output from the CLI tools, but also insulates any user scripts ingesting AMPCS data from changes to the CLI tools’ default output formats.

4. Dictionary

The interface between AMPCS and the flight software (FSW) is encompassed in the dictionaries developed by the FSW development team. AMPCS provide standardize XML schemas that define how a FSW dictionary should be formatted for ingestion by AMPCS. Depending on the number of AMPCS features being used by the mission, each dictionary may consist of definitions for packets, channelized data, EVRs, commands, and more. Dictionaries

allow AMPCS to capture the specifications laid forth in the flight interface specifications and process telemetry according to those specifications.

5. Mission Adaptation and Configuration

AMPCS was designed with an architecture to support multiple missions using a single code base. The majority of AMPCS capabilities are contained within a core module, and mission adaptations are built around this core module. Any mission specific capabilities may be added without disturbing the core module by integrating the capabilities onto predefined adaptation points. Additionally, almost every feature provided in AMPCS is configurable to fit a particular mission's needs. AMPCS utilizes a 3-level configuration design that allows easy reconfiguration for each mission adaptation. Each level in the configuration hierarchy overlays on the previous level, allowing AMPCS to deliver a default set of "system" configuration and allowing different projects and users to overlay their specific "project" and/or "user" configurations on top of the delivered system configuration.

Using this methodology, AMPCS can be effortlessly adapted for a new mission by identifying the features desired by the mission and configuring those features to fit the parameters of the mission. Items such as frame size, frame encoding, packet header, and spacecraft clock precision are adaptable and configurable for a mission, just to name a few. As a result, AMPCS can be adapted for use with a new mission with relatively little effort.

6. Graphical, command-line, and scripting interfaces.

AMPCS provides graphical and command-line user interfaces to many of its capabilities as well as a scripting interface for automation. Most AMPCS features provide multiple interfaces to support different modes of operations, from interactive to non-interactive, script-based operations.

7. External Interfaces

AMPCS provides standard external interfaces to provide hooks for mission-developed software. As depicted by Figure 4, this mechanism provides a powerful hook into AMPCS that enables integration with mission-developed tools to process data downstream of the NASA station and AMPCS.

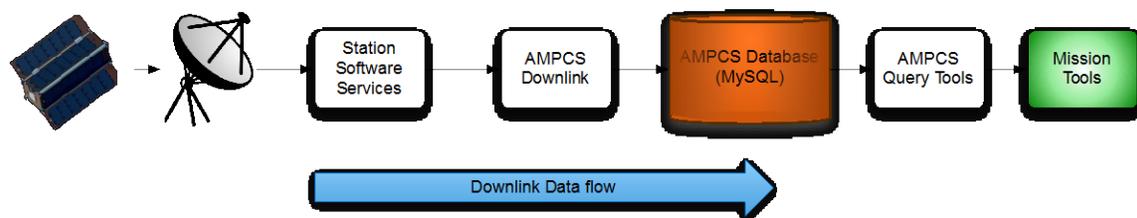


Figure 5. AMPCS Downlink Data Flow. Mission tools can easily tap into the data flow downstream of AMPCS to perform additional processing on raw or AMPCS-processed data.

D. Telemetry Processing

The AMPCS downlink processor handles the telemetry input acquisition and processing. AMPCS provides a variety of methods for ingesting raw spacecraft telemetry, from reading files on a local disk to accepting telemetry from NASA station services. This allows AMPCS to adapt to the different telemetry input sources throughout the lifecycle of the mission. Once telemetry input is acquired, AMPCS will process the frames to extract the packets, and process the packets to extract the channelized telemetry, EVRs, or data products. After the relevant data are extracted, they are sent to the real time stream for monitoring tools and the archival stream for storage.

AMPCS is compatible with CCSDS formatted telemetry frames and packets. The FSW is expected to package channelized data, EVR, and products in variable length CCSDS packets and load the packets into fixed-length CCSDS frames for transport. Once the frames are received and processed by AMPCS, the packets contained within the frames are extracted and its payload processed according to the packet type definitions in the packet dictionary provided by the mission.

E. Telemetry Monitoring

AMPCS provides a graphical monitoring tool that subscribes to the messages published by the downlink processor to the message bus. The message bus is based on the Java Message Service (JMS). JMS is a Java API that provides a loosely coupled method of communication between applications. In the case of AMPCS, the JMS

provides the uplink and downlink applications a method of broadcasting data and events to client applications such as real time monitoring applications and archival applications.

The real time telemetry monitoring applications contains a suite of displays and tools that provide a multitude of methods to monitor and analyze incoming spacecraft telemetry. Data may be displayed on variable-column tables, variable-axes plots, and user-defined fixed displays. In addition to the graphical monitoring tool, AMPCS provides a number of command-line tools to monitor the JMS for specific data types, providing a way for mission scripts and applications to effortlessly tap into AMPCS' real time broadcast.

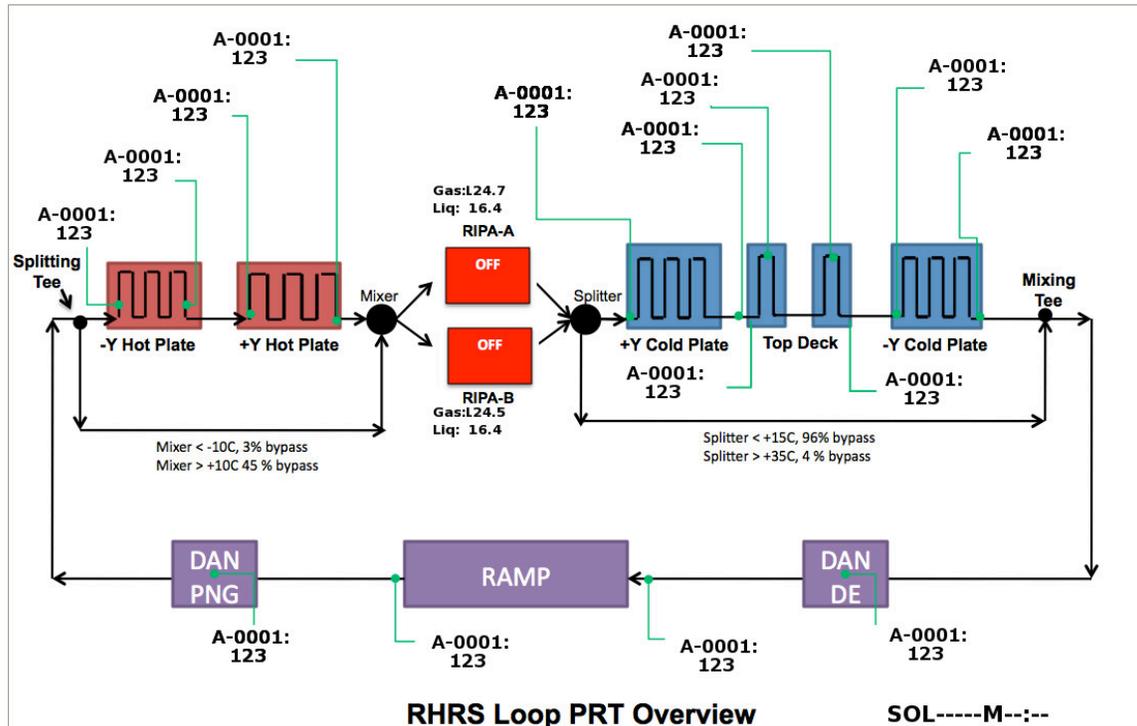


Figure 6. AMPCS Fixed-View Displays. AMPCS provide tools to generate highly customizable, user-defined telemetry displays. Each fixed-view display can contain a number of telemetry display items, fixed to a specific location on the page, and optionally overlaid on a user-provided image. Actual channel IDs data has been removed from this image and substituted with false channel IDs and data values.

Missions can define alarms for specific channels or a combination of channels. Alarm definitions are a part of the mission's dictionary, and ingested by AMPCS. The alarm definitions define certain conditions of telemetry values that must be met before an alarm is triggered. There are many different flavors of alarms, from the basic high/low value alarms and change alarms, to more complicated combination and hysteresis alarms. For each alarm definition, there are two levels of alarms, yellow and red, each with its own threshold. When an alarm is triggered by incoming telemetry, AMPCS will mark the appropriate channels as "in-alarm" at the appropriate alarm level. This alarm state is reflected in displays containing the channel and the archived channel value is marked as being "in alarm" at the particular level. This way, alarms can be monitored in real time as well as reviewed at a later time to see exactly which channel values went into alarm and which channel values went out of alarm. On top of the standard alarm features, AMPCS offers alarm notification, which allows missions to configure AMPCS to send emails and/or text messages to designated recipients in the event of an alarm on a particular channel or a set of channels.

F. Spacecraft Commanding

The AMPCS Uplink module provides capabilities to transmit commands to the spacecraft's FSW. Depending on the venue, the command data flow path may vary. Like with other AMPCS applications, AMPCS uplink supports command scenarios in test venues as well as in operations venues. For example, during early FSW development,

commands may be sent directly from AMPCS to the FSW through a socket, while during operations, the command would go from AMPCS to the station uplink service, then to the station, and eventually radiated to the spacecraft.

The AMPCS uplink application translates the command mnemonic and command arguments to binary format and packages them according to CCSDS specifications. For each command sent, AMPCS saves the exact command and its argument as a file on the local disk, so that it can be resent if necessary. This is a useful feature for preparing and validating commands in the spacecraft testbed before radiation to the actual spacecraft.

Command mnemonics and argument values are defined in the mission's command dictionary, which is ingested by AMPCS and used in the translation. AMPCS provide several uplink capabilities to support the wide range of use-cases at different phases of the mission. Each venue may be configured to enable a subset of the available features to ensure that users are only using the features appropriate for the venue. The subsections below describes the major capabilities offered by AMPCS uplink.

1. Immediate Commanding

The immediate commanding feature is the quickest way to send a command. A free-text field is provided to enter a single command mnemonic and any command arguments. AMPCS uses the mission's command dictionary to format the command appropriately and transmit it to the uplink target.

2. Command Builder

The command builder loads the mission's dictionary and provides a GUI to search for command mnemonics. After the desired command is found, a form is generated with a validated text field for each command argument. After the required arguments are provided and validated, AMPCS formats the command appropriately and transmit it to the uplink target.

3. Send File Load

This is analogous to an FTP transfer to the spacecraft's file system. It can be used to uplink binary files to the spacecraft for a variety of purposes, such as updating FSW.

4. Send Command Sets

AMPCS provides a feature that allows users to send a file containing a set of commands for the FSW. As stated above, this file is usually prepared and validated beforehand, by either using AMPCS or another tool. Upon selecting the command set file, AMPCS parses the file header and calculates the checksum to ensure that the file has not been tampered with since its creation and validation.

5. Fault Injector

The fault injector is a test feature that provides a way to inject controlled faults into the uplink stream. It is useful for testing FSW during development. The user is provided with a step-by-step wizard that builds a command and allows the user to inject a fault (corruption, bit-flips, missing frames, etc.) into the command, its arguments, or any of the command's transport units.

G. Automation

AMPCS includes extensive python-based frameworks to allow both test and operations automation. A python script can be written using the AMPCS python interface functions, enabling most user-initiated actions to be programmed by a script. For example, a script can be written to test a FSW command count channel by invoking a function to uplink a command and subsequently invoking another function to wait for the incremented command counter channel value before proceeding to other activities. This example is a notable methodology, called closed-loop commanding as shown in Figure 6. It can be used in the operations venue for "lights-out" commanding, whereby a script is initiated during a spacecraft pass to uplink commands based on the traversal of a decision tree using downlinked spacecraft telemetry.⁶ The functions provided by the python frameworks include retrieval and waiting for channelized data values and EVRs, command uplinks, station command service control, and a number of different convenience functions designed to minimize the effort of automation.

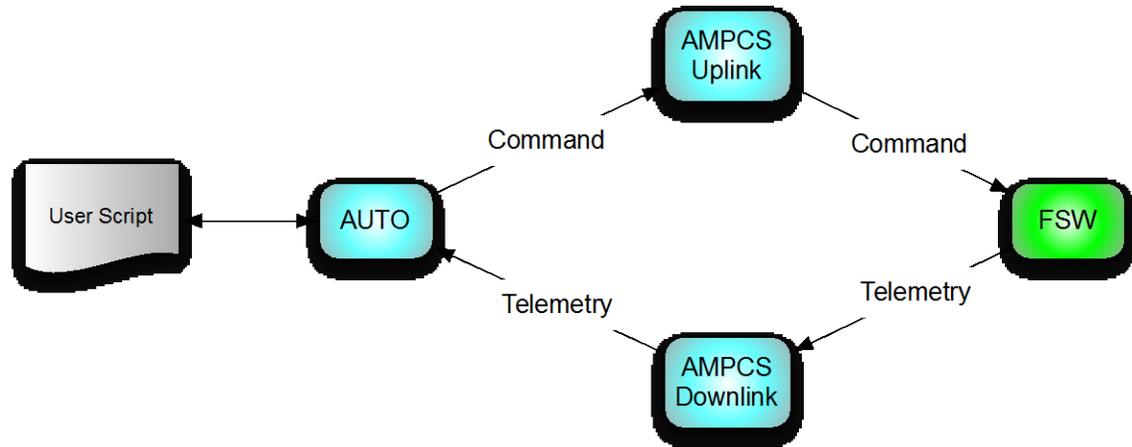


Figure 7. Closed-loop commanding using AMPCS. *The AMPCS Utility Toolkit for Operations (AUTO) framework provides an API that enables scripted uplink, which allows the mission to make uplink decisions based on downlinked telemetry. For simplicity, intermediary systems between AMPCS uplink/downlink and FSW are excluded from the diagram.*

H. Data Archival and Analysis

All AMPCS data, both raw and processed, are stored in a MySQL database for later retrieval. Any data being viewed on monitoring applications, such as incoming frames, system logs, or channelized spacecraft telemetry, can be later reviewed by using the AMPCS query tools to retrieve the desired data sets from the database. AMPCS provides built-in queries and reports to provide users with a base level of capabilities. Spacecraft teams typically develop their own sets of tools that access the AMPCS database using the built-in query tools.

Processed data can be queried to perform post-test analysis, plotting, or ingestion into other tools. Raw data (frames or packets) can also be queried by the supplied tools into a file and replayed through the downlink processor, or directly fed from the database into a new downlink session.

IV. INSPIRE Overview

INSPIRE, which will be deep-space-flight-ready in summer 2014, will become the world's first dedicated interplanetary CubeSat mission, opening deep-space heliophysics and planetary science to the CubeSat revolution. The INSPIRE mission will accomplish a tiered set of technology-demonstration and education objectives: it will demonstrate that NASA CubeSats can (1) operate, communicate, and be navigated outside of Earth's orbit; (2) host and support CubeSat components whose hardware has not yet had the opportunity to fly in deep space; and (3) deliver useful science that in turn opens future mission opportunities for investigators. These objectives intimately embody many NASA strategic objectives, and enable a new generation of low-cost solar system explorers.



Figure 8. INSPIRE Spacecraft Mechanical Fit Check. *While Reed-Solomon and Convolutional encoding are applied (and removed) within the Iris radio subsystem (bottom four boards), the JPL-built flight software, running on a small microprocessor on the Command and Data Handling Unit, is responsible for all CCSDS frame creation, decoding, and appropriate processing for uplink, downlink, and relay communications. The spacecraft is approximately 30cm tall.*

The INSPIRE flight system comprises two identical, three-axis-stabilized 3U spacecraft that combine existing subsystems for the Command and Data Handling (C&DH) watchdog, attitude determination, and power functions with next-step modifications of subsystems for cold-gas attitude control and deep-space navigation & communication. Demonstrating the integrated system performance of these core spacecraft components will establish a proven foundation for diverse future NASA missions to host special-purpose payloads in deep space.

The spacecraft will also host a science payload: a half-U JPL-developed compact vector-helium magnetometer to measure the fine structure of the solar wind, and prove the science utility of such a small platform.

The project significantly leverages experience from prior CubeSat launch opportunities, including hardware, software, and personnel from the RAX2, M-Cubed/COVE, GRIFEX, CP-series, BEVO-2, and ArmadilloSat CubeSat projects.

INSPIRE's demonstration of CubeSat functionality and utility in the deep-space environment is crucial as a stepping-stone for interplanetary CubeSats. By leveraging JPL's 50 plus years of deep space experience, INSPIRE will establish a flight heritage for future interplanetary CubeSat missions, as well as creating a cadre of partners experienced with the challenges of interplanetary missions.

V. AMPCS for INSPIRE

As described in section III.A.3, TAYF is a widely-accepted best-practice for spacecraft mission development. Successfully implementing TAYF depends on the mission's policies, procedures, software tools, and computing environment. As the majority of ground operation procedures involve software tools, it is imperative that a consistent computing environment and platform is provided.

A generic adaptation of AMPCS was delivered to INSPIRE as a part of the AMMOS, which is a collection of software tools, applications, environment, and platforms. The deployment of AMMOS relies on a concept of a kickstart, which is the process of installing AMMOS software and environment on a system.

Typically, the goals of FSW development on most JPL missions revolve primarily around the needs of the flight system, and those needs differ from mission to mission, mainly due to the different science payloads. Effort would have to be allocated for every deviation from past missions, and most missions have deviations. The INSPIRE mission took a different approach in developing the FSW. The majority of the INSPIRE FSW is accommodating the existing GDS features and supported standards, enabling INSPIRE to use AMPCS as an off-the-self product. With only configuration inputs, INSPIRE was able to take the generic AMPCS adaptation and use it for their mission. This resulted in an extremely low-cost effort to provide INSPIRE with a command and telemetry system.

A. Kickstarts

An AMMOS kickstart is the encapsulation of software, deployment procedures, dependency management, and configurations. Kickstarts are versioned, managed, and deployed by a kickstart server. INSPIRE was provided with a kickstart server, which they used to instantiate workstations in their testbed and eventually to instantiate workstations in their Mission Support Area (MSA) for operations.

In order to accommodate rapid configuration and deployment, the INSPIRE project leveraged virtualized systems for both development and mission ground support infrastructure. Kickstart images worked well in this environment, requiring minimal interaction after initiation and completing quickly. Initially some amount of post-configuration was required which was later wrapped into the image. The virtualization environment allowed for rapid duplication and deployment of a fully configured image which included all the necessary software tools: scripting, numerical analysis, embedded development environment, networking, and serial link capability.

B. Configurations & Dictionary

As the INSPIRE FSW was developed with a CCSDS complaint interface for uplink and downlink, minimal modifications of AMPCS configuration was required. Most of the configuration items were used to “personalize” AMPCS for the INSPIRE mission. These configurations additions include:

- Transfer frame size, encoding, and sync marker
- Spacecraft clock precision
- Spacecraft ID (SCID)
- Spacecraft name and mnemonic

Besides the configuration additions, INSPIRE developed a complete dictionary based on multi-mission standards that include:

- Channel dictionary
- Channel derivation algorithms
- Packet Application Process Identifier (APID) definitions*
- Packet decommutation (decom) maps†
- Command dictionary

C. FSW Development

The nature and form of INSPIRE's flight software was ultimately driven by the constraints of the spacecraft's processing platform: a low-power microcontroller with 8kB of RAM. In this space, the software rapidly coalesced from a tight and robust core toward higher functionalities that were required at various phases of testing. Establishing a link with an existing "flight-like" ground system early in the software process was critical to the development of higher functionality, making the telecommunications protocols implemented by the ground software among the first elements developed and tested on top of the software core. Ultimately, spacecraft-internal data flow revolved around standard telecommand and telemetry formats implemented and documented by the ground system.

Simple and fast reconfigurability was critical. Software command handlers in the flight software were auto-generated from the ground system command dictionary at compile-time. This assured that the flight software command handlers would be exactly matched to the ground software dictionary. It also effectively removed the possibility of mistakes from hand-coding a large number of commands in the flight software. A similar approach was taken with telemetry and spacecraft-specific identification code segments.

Using basic network sockets, data streams could be trivially piped across any standard network link, regardless of geographic differences between ground and flight hardware. Further, data flow to and from INSPIRE hardware was low speed (for terrestrial networks) and byte-aligned, making the ideal low-level hardware interface a simple and universally understood serial link. Combined, these factors allowed great flexibility in development, easily accommodating a rolling cycle of continually linking and testing ground and flight software.

D. Current Status

In lab, the AMPCS systems live in easily-duplicated virtualized servers, allowing for the segregated testing of multiple spacecraft units simultaneously. Flight and prototype hardware can easily be moved about the lab (even changing network links) without the need to change anything beyond the basic settings that launch an AMPCS

* APID is an attribute of a CCSDS packet that defines the onboard application process that generated the packet. This tells the ground system what type of data to expect in the packet and how it should be processed

† Decom maps define the location of the bytes in the packet payload where channelized values are stored.

session. All tests are automatically logged at both the data stream level and in AMPCS databases. Spacecraft commands are developed and added to the mission dictionary on an as-needed basis as testing progresses. Similarly, telemetry channels are created as the need arises during both development and testing activities. All dictionaries are kept under revision control and mapped directly to each software release, preventing confusion with dictionary-flight software version mismatches.

The portability of the data flow interface allowed for rapid re-configuration and incorporation of additional elements in the telemetry chain, including link simulators, and radios. This portability proved valuable during the radio frequency (RF) compatibility and end-to-end data flow test with the DSN's Development Test Facility 21 (DTF-21). Due to the fact that DTF-21 was geographically distant from the INSPIRE laboratory, a single AMPCS image was installed on a compact computer system along with the embedded development environment. This created an easy-to-transport mobile command and telemetry interface that doubled as a hardware troubleshooting system for the entire spacecraft during testing.

VI. Current Challenges

The AMPCS for INSPIRE initiative has proven to be successful so far, however there were challenges that were encountered throughout the experience. Most notably, AMPCS is a very flexible product designed to satisfy a wide range of mission requirements. As such, it requires a substantial level of effort to correctly configure it for each specific mission. Additionally, AMPCS provides a large catalog of features designed for reliable mission operations, which is accompanied by a steep learning curve for users not familiar with AMPCS or certain mission testing and operation procedures. Lastly, AMPCS only supports missions that communicate using CCSDS protocols, which is not widely used in the CubeSat community.

Work is being done to address these challenges so that future low-cost missions will not encounter these issues. We are working to come up with standard configurations for low-cost missions so that it can use AMPCS out of the box and to provide a more concise set of user guides. The next section discusses in further detail the ongoing effort by MGSS to support low-cost missions.

VII. Future Work in Support of Low Cost Missions

Adapting to the unique features of new missions, including differences between subsequent generations of C&DH, instrument payloads, spacecraft buses, etc., tend to drive up the costs of each new mission's GDS. Because of these unique features, a typical new mission will inherit the previous mission's GDS, adapt that GDS for its own needs and during the adaptation process, improve upon it based on the unique experiences of the GDS engineers. JPL's MGSS organization is funded to provide a new mission's basic GDS components from a multimission catalog of capabilities. Due to the unique features of each new mission, MGSS is also funded to make changes to the Multimission GDS's code baseline, which can be costly as each change must be designed as a reusable component serving multiple missions.

Low cost missions pose a special challenge to MGSS. Low cost missions have very few individuals allocated to developing, adapting, deploying, and maintaining the GDS, along with a meager budget to pay for these services. On the other hand, low cost missions do not impose the same level of requirements and requests for new features on the GDS as flagship missions. As a result, MGSS is investigating a new approach to provide the telemetry and command components of a mission's GDS to low cost missions at a minimal cost. This new approach is called the Interplanetary Networks Directorate Customer Assistance Package (ICAP).

The ICAP is intended to be a support package designed to provide mission GDS engineers with a set of telemetry and command capabilities on the first day of the mission, with little or no involvement from MGSS. In this way, the cost to the low cost mission is minimized, as long as no MGSS intervention is needed. The ICAP is composed of two main pieces: The deployment package and the supporting documentation and procedures.

The deployment package consists of a downloadable package of software. Typically a set of Red Hat Linux RPM packages and an environment dependency description. The MGSS components (populated with AMPCS initially) contained in the ICAP deployment package are pre-adapted for a generic low cost mission. The package also includes sample test data, sample telemetry and command dictionaries, sample telemetry displays, and reports. From the MGSS perspective, the generic low cost mission is treated like any other MGSS supported mission. New features that are added to the MGSS controlled baseline from other missions are included in future releases of the generic low cost mission adaptation.

The supporting documentation and procedures component is a set of material that is intended to assist each mission's GDS engineer with creating, adapting, and deploying the deployment package. The supporting documentation and procedures will consist of:

- Procedures for dealing with applicable State Department and NASA Rules and Polices, including procedures for addressing ITAR and Export License issues, IT Security considerations and Caltech/JPL Licensing issues
- Relevant Tests Performed
- Requirements and Constraints of the software contained in the deployment package
- A description of how to interface with the DSN
- Descriptions of GDS software processes including deploying the software, testing the software configuration, and operating the software,
- Other GDS development and operations documentation including User's Guides, Adaptation Guides and Troubleshooting Guides,
- Cost Estimates if MGSS services are required.

A draft ICAP will be available by the end of 2014 with updates to follow in subsequent years.

VIII. Conclusion

The MGSS and INSPIRE partnership resulted in a novel methodology to develop low-cost space missions. Instead of having the development of the flight system drive the development or the adaptation of the ground system, we viewed the FSW and GDS as two loosely-coupled modules in a larger system. By adhering to standards and defining clean, concise interfaces, we were able to completely insulate the modules from one another without compromising their ability to work together. As a result, MGSS was able to provide INSPIRE with a cutting-edge command and telemetry system at a very low cost.

Appendix

Acronym List

AMMOS	Advanced Multimission Operations System
AMPCS	AMMOS Multimission Data Processing and Control System
API	Application Programming Interface
APID	Application Process Identifier
AUTO	AMPCS Utility Toolkit for Operations
CCSDS	Consultative Committee for Space Data Systems
C&DH	Command & Data Handling
CLI	Command Line Interface
CLTU	Command Link Transmission Unit
DSN	Deep Space Network
DTF	Development Test Facility
EVR	Event Record
FSW	Flight Software
GDS	Ground Data System
GUI	Graphical User Interface
ICAP	Interplanetary Networks Directorate Customer Assistance Package
INSIGHT	Interior Exploration using Seismic Investigations, Geodesy and Heat Transport
INSPIRE	Interplanetary NanoSpacecraft Pathfinder In Relevant Environment
JMS	Java Message Service
JPL	Jet Propulsion Laboratory
MGSS	Multimission Ground Systems and Services
MSA	Mission Support Area
MSL	Mars Science Laboratory
NASA	National Aeronautics and Space Administration
RAM	Random Access Memory
RF	Radio Frequency
SCID	Spacecraft Identifier
SCMF	Spacecraft Command Message File
SMAP	Soil Moisture Active Passive
TAYF	Test as You Fly
UHF	Ultra High Frequency
XML	Extensible Markup Language

Acknowledgments

The authors would like to thank the following teams for their dedication to this initiative:

- The AMPCS Team for their hard work and commitment to making AMPCS a superb multimission product and their support to INSPIRE and the CubeSat initiative: Mark Indictor, Lori Nakamura, Kyran Owen-Mankovich, and Nicole Witek.
- The extended INSPIRE and NASA team supporting this effort, especially Lauren Halatek, Allen Kummer, George Rinker, Steve Waldherr, and Thomas Werne.
- The MGSS organization for their support and guidance: Eleanor Basilio, Andrew Downen, Scott Markham, Steve Ma, Steve Ng, George Rinker, Gregory Welz.

The work described in this paper was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology (Caltech), under a contract with the National Aeronautics and Space Administration (NASA). The work was funded by the Multimission Ground System and Services Office (MGSS) and the Interplanetary NanoSpacecraft Pathfinder In Relevant Environment (INSPIRE) mission.

References

¹ Funase, R., et al., "Technology demonstration on University of Tokyo's pico-satellite "XI-V" and its effective operation result using ground station network," *Acta Astronautica*, Vol. 61, Issues 7-8, 2007, pp. 707-711

² Springmann, J., Kempke, B., Cutler, J., "Initial Flight Results of the RAX-2 Satellite" *Small Satellite Conference*, Logan, UT, USA, Aug 13-16, 2012

³ Minelli, G., et al., "Mobile CubeSat Command & Control (MC3) Ground Stations" *CubeSat Developers' Workshop*, Logan, UT, USA, Aug 11-12, 2012

⁴Dehghani, N., Tankenson, M., "A Multi-mission Event-Driven Component-Based System for Support of Flight Software Development, ATLO, and Operations first used by the Mars Science Laboratory (MSL) Project," *SpaceOps 2006 Conference*, Rome, Italy, Jun 19-23, 2006.

⁵Arthur Amador, "Test As You Fly, Rev2", Guidance, JPL Rules! DocID: 68672 (internal JPL document), October 02, 2010.

⁶Choi, J. S., Sanders, A.L., "Cost-Effective Telemetry and Command Ground Systems Automation Strategy for the Soil Moisture Active Passive (SMAP) Mission," *SpaceOps 2012 Conference*, Stockholm, Sweden, Jun 11-15, 2012.