Transferring Files between the Deep Impact Spacecrafts and the Ground Data System using the CCSDS File Delivery Protocol (CFDP): a Case Study

Felicia A. Sanders,[*] Grailing Jones, Jr.,[†] and Michael Levesque[‡]
4800 Oak Grove Drive, Pasadena, California, 91109, USA

The CCSDS File Delivery Protocol (CFDP) Standard could reshape ground support architectures by enabling applications to communicate over the space link using reliable-symmetric transport services. JPL utilized the CFDP standard to support the Deep Impact Mission. The architecture was based on layering the CFDP applications on top of the CCSDS Space Link Extension Services for data transport from the mission control centers to the ground stations. On July 4, 2005 at 1:52 A.M. EDT, the Deep Impact impactor successfully collided with comet Tempel 1. During the final 48 hours prior to impact, over 300 files were uplinked to the spacecraft, while over 6 thousand files were downlinked from the spacecraft using the CFDP. This paper uses the Deep Impact Mission as a case study in a discussion of the CFDP architecture, Deep Impact Mission requirements, and design for integrating the CFDP into the JPL deep space support services. Issues and recommendations for future missions using CFDP are also provided.

I. Introduction

THIS document provides an introduction to the mission operations experience with the use of CFDP by the Deep Impact Mission. Deep Impact was the first Jet Propulsion Laboratory mission to fly the CFDP, where the JPL Deep Space Mission System (DSMS) Tracking, Telemetry, and Command system that provided the CFDP ground system services to the Deep Impact Mission.

II. The CCSDS File Delivery Protocol

The CCSCS File Delivery Protocol (CFDP) is a specification describing an open, international standard file delivery protocol. CFDP provides the mechanism to transfer files to and from a remote entity using a local entity. The content of the files is independent of the protocol used to transfer them. Files can be transferred reliably, where it is guaranteed that all data will be delivered if transferred without protocol error, or unreliably, where a best effort transfer is performed. Files can be transmitted with a unidirectional link, a half duplex link, or a full duplex link, with near-Earth and deep space delays. File transfers are controlled through the user application and can be triggered automatically or manually. Figure 1 is a sequence diagram illustrating the file transfer protocol[1].

The CFDP Flight-Ground baseline is a JPL software implementation of the CCSDS specification that when integrated into a complete flight software system or ground software system, provides point-to-point reliable and unreliable file delivery between systems. The CFDP implementation provides a fault tolerant communication mechanism that operates efficiently over the highly constrained physical links encountered in deep space communications. It is designed and implemented to have full error recovery from system resets. The CFDP architecture and implementation allows CFDP to be integrated and executed on a variety of hardware platforms including constrained platforms often required for deep space flight systems.

The CFDP (Flight-Ground baseline) implementation consists of three fundamental assemblies: (1) the Protocol Daemon, (2) the Adaptation Layer, and (3) the Test Harness. The Protocol Daemon assembly is a process that is responsible for implementing the CFDP protocol functional requirements. It processes Protocol Data Units (PDUs) in real-time and

(1) extracts PDU inputs from the remote CFDP entity,

[*] Software Engineer, Flight Software and Data Systems, 4800 Oak Grove Drive M/S 230-305.
[†] Software Engineer, Planning and Execution Systems, 4800 Oak Grove Drive M/S 264-235.
[‡] Software Engineer, Software and Information System Engineering, 4800 Oak Grove Drive M/S 230-305.

(2) formats PDU outputs destined for remote CFDP entity,
(3) implements the CFDP standard rules and other requirements within the specification,
(4) utilizes the Adaptation Layer software for platform independent system resources,
(5) communicates with the local CFDP user software using the Adaptation Layer services,
(6) communicates with the local User Transport (data link) software.

The Adaptation Layer consists of software and its software libraries that enable the functional requirements to be deployed independent of the operational environment. It includes communications services, data services (pipes, queues, lists), files services (file manipulation and control, file storage management), controls (semaphores), etc.

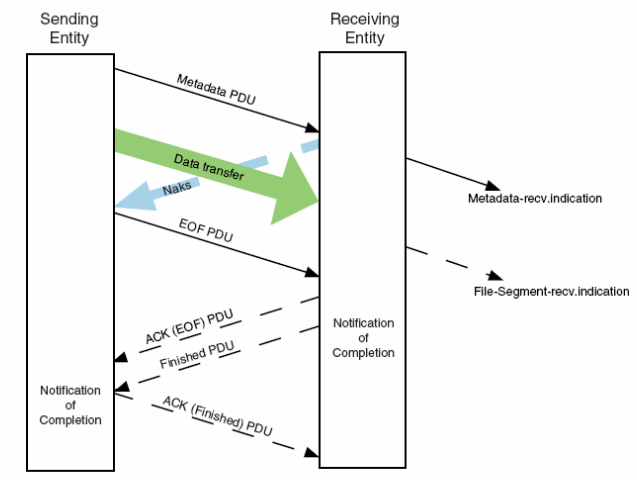

Figure 1. Protocol Sequence Diagram.

III. Uplink Scenario

The Deep Impact Mission used traditional command services for non-CFDP uplinks with a unique command virtual channel for Hardware CRC commands and another command virtual channel for software immediate commands. Deep Impact uplink scenario did not call for the interleaving of commands with CFDP file uplinks. In addition, Deep Impact did not suspend and resume CFDP uplinks for the purpose of interleaving commands. All uplinks occurred within a single station uplink activity, so the mission had no need to split an uplink across multiple station tracking sessions. File reception status was reported in telemetry, but at a minimum, files not received correctly through CFDP never reached their final destination in the on-board spacecraft filestore. Files uplinked were stored in the Data Management System on the ground. The ground user application provided a method for the user to select and queue approved or scheduled files for uplink using CFDP. For commands, radiation success or failure status was provided back to the user through the user application.

A. Flyby Spacecraft Uplink Process

Spacecraft commanding on Deep Impact divided into two categories, real-time and stored. Uplink of real-time commands was accomplished using traditional command services. This traditional method featured unique command virtual channels for Hardware and Software immediate commands. File uploads were accomplished via CFDP over their own virtual channel.

The stored command files for uplink varied in content depending on their source. Three principle groups provided files for uplink, the Spacecraft Team (SCT), the Flight Software Group (FSW) and the Autonomous Navigation Group (AutoNav). Figure 2 illustrates the mapping of the file types to their respective processes as part of preparation for radiation.

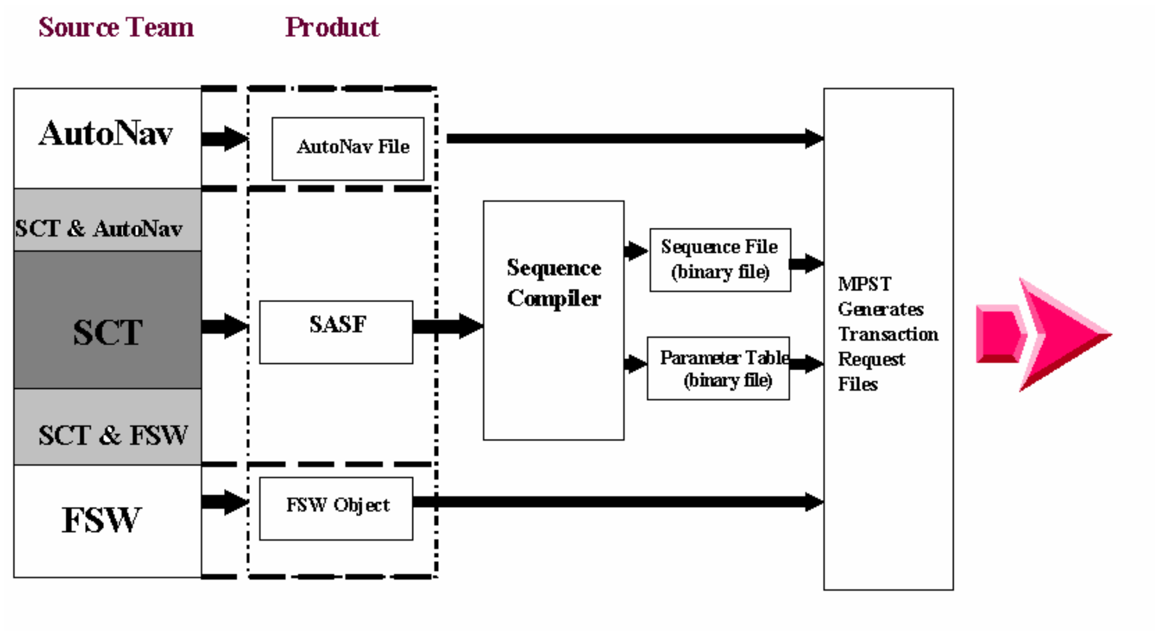
Source Team
Product
AutoNav
SCT & AutoNav
SCT
SCT & FSW
FSW
AutoNav File
SASF
FSW Object
Sequence Compiler
Sequence File (binary file)
Parameter Table (binary file)
MPST Generates Transaction Request Files

Figure 2. Uplink Product to Process Mapping

Of the stored command files for uplink, Sequence Files and Parameter Tables were the most common. These uplink products were coded by members of the Spacecraft Team as Spacecraft Activity Sequence Files (SASF). SASFs are ASCII text files of absolute or relative timed lists of commands. SASFs were submitted to the Sequence Team for further verification and compilation into Sequence Files or Parameter Tables. These compiled binary files were then placed in the project file repository for further processing. Upon successful completion of uplink, the Sequence Files and Parameter tables were directly sourced by onboard flight software. Sequence Files provided time ordered list of directives, while Parameter Tables contained parameter values of used by flight software.

In addition to Sequence Files, The Flight Software Team (FSW) also submitted precompiled Flight Software Object files for uplink. These objects were uploaded to directories on-board the spacecraft for subsequent sourcing by Flight Software by commanded reboot. The Flight Software Team also submitted SASFs to compile into Parameter Tables for uplink.

The Autonomous Navigation group provided yet another file type for uplink. The AutoNav file was an ASCII file of text containing parameters used by the Autonomous Navigation software. This file did not require compilation for use.

B. Uplink Transaction Request

To be properly sourced by onboard software, all files needed to be routed to their designated directories onboard and stored with designated file names. Complicating matters for Deep Impact, the project uplink file repository schema demands files follow a naming convention that usually resulted in the product to be uplinked having a name different from what was needed in spacecraft memory. The renaming of the ground-sourced file and its ultimate routing and naming onboard was done using Transaction Request Files (TRFs).

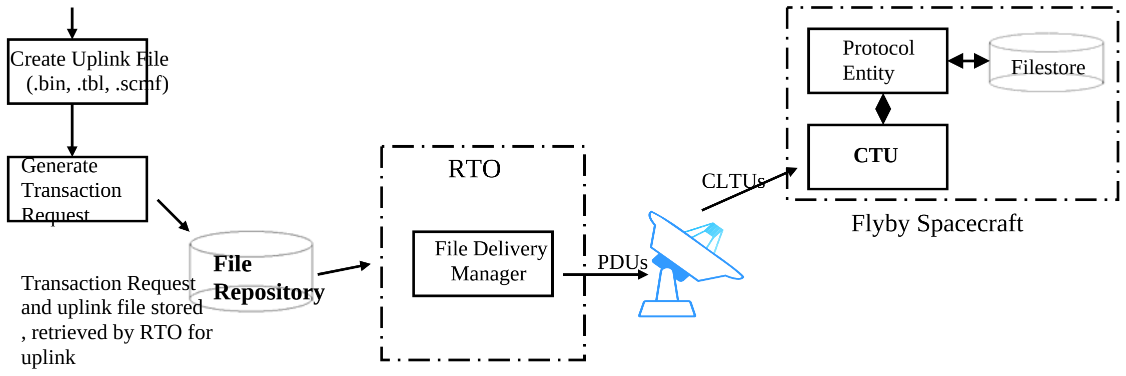
Create Uplink File (.bin, .tbl, .scmf)
Generate Transaction Request
File Repository
Transaction Request and uplink file stored , retrieved by RTO for uplink
RTO
File Delivery Manager
PDUs
CLTUs
Protocol Entity
Filestore
CTU
Flyby Spacecraft

Figure 3. Uplink Product to Spacecraft

Transaction Request Files contain the parameters governing particular CFDP transactions. In the case of uplink, only the CFDP "Put" operation was used. Deep Impact used a tailored suite of CFDP options for carrying out the "Put" operation in uplink. Ease of operations, and concern with protocol robustness drove this decision.

As mentioned above, the Uplink Files were the command files ready for radiation and stored in the ground repository. To store in the ground repository, all file names were given an 11 character preamble. The convention of this preamble and the root file name afforded enough discriminators to determine the proper parameter values for the CFDP transaction. The default values designated for other Transaction Request parameters completed the set.

Spacecraft ID was the first parameter of the request. This parameter is the numeric designation of the spacecraft where the destination protocol entity of the proxy "put" operation resides. This served to differentiate between Flyby and Impactor

CFDP Destination ID is the identifier for the protocol entity destination for the file. Protocol entities are allocated to individual filestores, so in the case of Deep Impact the main and redundant spacecraft computers (which did not share memory) were given their own. In addition to the two Flyby protocol entities, the Impactor also had one for its single computer.

The Source Filename is the name of the uplink product as stored in the ground file repository. The File Delivery Manager used this name to access the file to be radiated from the repository on the ground. The destination filename is the absolute path and name of this radiated file in the destination memory. The convention adapted on Deep Impact was to drop the 11 character preamble of the ground product and retain the name root in determining the destination filename. Rules for interpreting the preamble gave the spacecraft ID, as well as the absolute path and name for file destination.

The remaining Transaction Request parameters were handled by designating default values. CFDP Segmentation Control controls whether files will be delivered as an array of constant or variable length records. Since this feature was not used, the default value for this parameter was set to "FALSE". CFDP also allowed for either Reliable or Unreliable forms of proxy put. In Reliable mode, the receiving node can send metadata to request missing packets and give the sender status of the transaction. For uplink this value was defaulted to "TRUE".

The last parameter was the Transaction Request File name for the transaction. Each file for uplink was given its own individual Transaction Request File containing the information described above. Transaction Request Files were created as part of the processing of their client uplink files by the Sequencing Team, and stored in the same file repository. The Transaction Request Files were given the name of their client uplink files, with the extension ".trf".

When preparing to send files, the real-time operations personnel operating the File Delivery Manager would use an uplink manifest listing the individual files for uplink and their attendant TRFs. The operator draws the TRFs listed into the File Delivery Manager, and each individual TRF point to its client uplink file in the file repository. The parameters contained in the individual TRFs govern the CFDP transaction as described above.

C. Impactor Spacecraft Uplink Process

The impactor telecommand link was initiated from the ground to the flyby via X-Band, then from the flyby to the impactor via a separate RF cross-link. The impactor cross-link receiver fed the “dirty” bitstream to the Command/Telemetry Board (CTB) on the impactor. Since the CTB expected to see CLTUs as its input, the flyby cross-link transmitter was required to output CLTUs. In order to simplify functionality on the flyby, the CLTUs were created by the ground system only and uplinked to the flyby as the contents of an impactor command file. The file was identified as destined for the impactor. Special software on the flyby facilitated the transfer of CLTU data contained in the impactor command file to the cross-link transmitter on the flyby. If a file contained more than one CLTU, the ground system was responsible for inserting the idle pattern between CLTUs.

Because the flyby was not equipped with the capability to interpret impactor commands or telemetry, CFDP could not be used for reliable file transfer across the flyby-impactor link. However, the impactor was equipped with the CFDP capability, so a method to accomplish a somewhat more reliable transmission of file data to the impactor was devised. This method entailed having the ground system create and send two (2) copies of each PDU in the impactor command file. In this way, the inherent file accounting capability of CFDP was used to increase the likelihood of receiving the file in its entirety, though the transfer method was not guaranteed.
<ADD FIGURE>

IV. Downlink Scenario

The Deep Impact Mission used CFDP for the data transfers from both the flyby and impactor spacecrafts. The CFDP data was merged with traditional real-time CCSDS telemetry frames containing packets for each spacecraft on separate virtual channels. CFDP PDUs also contain CCSDS packets as stored into files on-board the flyby spacecraft. Under nominal mission scenarios, the data access latency built up with transfer packet data as files was not prohibitive to the mission plan or spacecraft health and safety. Engineering telemetry packets in CFDP PDU’s were processed and displayed as they were received. This was very important for analysis and recovery of safe-mode conditions at low bit rates. Finally, file reception status was made available to the flight operations team, so that incomplete files could be scheduled for re-transmission, and partially received files could be analyzed. This is important in the “live-for-the-moment” philosophy of the flyby mission, since there was the possibility that the spacecraft might not survive to allow for the retransmission of science images.

V. Issues and Resolutions

Deep impact showed us unacknowledged file transfers worked for the store and forward relay link. Deep impact was a single impact event and flyby scenario so all return link data transport was sequenced and could be pre-prioritized and transferred in a first in first out manner by file type. Once the spacecraft was past harm there was an extended period for retransmissions over the return link. All file data could be sent multiple times in its entirety based on retransmission commands from the ground. Lists of directories and file names where the meta-data about file types had been mapped worked well under this scenario. CFDP was designed to provide these data transport capabilities and to easily handle these scenarios.

For unacknowledged mode CFDP transactions, as used on DI downlink, some partial data file reconstruction was inevitable. For DI, partial data recovery was not critical once the spacecraft was safely past the dangers of the comet encounter. Had a lengthy retransmission period not existed, partial data in some cases would not have been usable since it was difficult to trace to its original event and even simply data type when the critical CFDP meta-data was lost. In unacknowledged mode the recover of the data was only through complete file retransmission. For this mode of transaction, CFDP itself may need to change to include self-identifying file data PDUs as part of the standard. <TBD: tie into the use of insert packets as opposed to packets containing unique APIDs>

In addition criticality, priority and pedigree will need to be included with the data. Missions will want to be able to transfer commands, both autonomous or sequenced and real-time, to manage data including deleting, change priorities or QOS, and confirmation of receipt, particularly important in forward link file transfers, across the participating assets. Standards will likely need to emerge to make this interoperable.

While CFDP provides the data transport capabilities necessary to move files in acknowledged or unacknowledged modes and has extendable capabilities for transferring meta-data it alone was not designed to provide data management capabilities. Data catalogs will be critical for this since directory listings or embedding information in directory structures or file names will provide limited capability at best.

Thus data management in the space communications pipeline will become more important and critical to the success of future missions in maximizing science return at acceptable operations costs.

VI. Conclusion

<TBS>

Acknowledgments

<TBS>

References

[1]CCSDS 727.0-R-5: CCSDS File Delivery Protocol (CFDP). Blue Book. Issue 3. June 2005.

[2]Smyth, David & Krasner, Sandy, "Proposed Protocol Stack, Packet Formats, And Topologies for Mars Science Laboratory", internal JPL presentation at JPL DM/DT Workshop, September 8, 2005.