Commanding via the CCSDS Forward CLTU Service

Deane E. Sibol
The Johns Hopkins University Applied Physics Laboratory
Laurel, MD USA 20723-6099
(443) 778-8429
deane.sibol@jhuapl.edu

Valerie A. Mallder
The Johns Hopkins University Applied Physics Laboratory
Laurel, MD USA 20723-6099
(443) 778-7846
valerie.mallder@jhuapl.edu

Abstract. The Forward CLTU Service, a component of CCSDS's Space Link Extension, represents a substantial cost savings, mitigates risk, and minimizes ground system development and integration time. This exciting breakthrough enables a consistent interface, independent of the ground hardware and antenna performing the actual uplink. Thus, this standard protocol has the potential to expand the number of heterogeneous uplink assets available to a mission. In addition, mission operators can become familiar with one protocol, decoupled from any mission-specific commanding requirements.

The COmet Nucleus TOUR (CONTOUR) will be the first externally operated NASA mission to incorporate the SLE Forward CLTU Service. Launching in July 2002, CONTOUR will be operated from its Mission Operations Center (MOC) at the Johns Hopkins University Applied Physics Laboratory in Laurel, MD, USA, using the Deep Space Mission System (DSMS). CONTOUR will be the first mission to utilize the SLE Forward CLTU Service for launch and early operations.

The SLE Forward CLTU Service is 100% reusable on future missions. Subsequent missions managed by JHU/APL, including MErcury Surface, Space ENvironment, GEochemistry and Ranging (MESSENGER), Solar-TErrestrial RElations Observatory (STEREO), and New Horizons, will utilize this service, unchanged.

The rapid integration of this new technology into an existing ground system will be described.

Space Link Extension

In the early 1980's, the Consultative Committee for Space Data Systems (CCSDS) was formed by the major space agencies of the world to address the common problems in the development and operations of space data systems. Their goal was to reduce the cost of future space missions, enable cross support, and improve the understanding and ensure the preservation of space related data. Since established, CCSDS has developed recommendations for data and information standards. Focusing initially on ground to space element communications, their Space Link Recommendations have enjoyed widespread adoption by flight and ground system designers.

Building upon the CCSDS Space Link Recommendations, the Space Link Extension (SLE) Recommendations standardize the means by which mission operators configure and transfer data to/from the ground system.[1] While the Space Link standards focus on the transfer between the spacecraft and ground system, the Space Link Extension standards extend to/from the ground system and the user – typically the Mission Data Operation System (MDOS). Thus, the ground system that actually performs the uplink and downlink to/from the spacecraft becomes a "blackbox" to the user. However, the standard for each service includes enough information to allow operators at the MDOS to access the real-time status of the service being provided.

Adoption of these standards by space organizations and agencies would allow for efficient cross support operations using heterogeneous ground assets. To utilize another agency's ground system today without this standard, custom software interfaces and perhaps hardware would be necessary. This incurs additional mission development and maintenance cost, schedule and risk. Using the standard, these factors are eliminated. One consistent interface is used to all ground systems, independent of the actual ground hardware and antenna. Operators that become familiar with this standard interface can easily communicate between multiple agencies and even across multiple missions.

Moreover, as more space agencies adopt these standards, the number of additional ground system resources accessible to a mission increases, adding greater flexibility in scheduling mission coverage.

Abstract Layered Approach

The SLE Application Program Interface (API) uses an abstract layered approach that allows for consistent interfaces, which are independent of the implementation. The SLE API consists of the Service Element, Proxy, Operations, and Utilities and includes interfaces to the Application. Figure 1 depicts how these components interact at an abstract interface level.[2]

A powerful example of the advantage of this approach is the proxy layer. The proxy layer is responsible for the actual format and transfer of data between the two end points, referred to as the user and provider. This layer, in particular, may need to be different between ground systems, as the communication technology in use may differ. There are several alternatives to deal with this challenge.[3] To mitigate risk, JHU/APL chose to use an agreed upon technology (i.e. TCP) and, in fact, uses the identical implementation (from the Jet Propulsion Laboratory (JPL)).

In other cases, if a gateway is not available to bridge the technologies, only one layer - the proxy layer - of the API is replaced and all others, including the application remain unchanged. That is, the new proxy layer can be linked into the existing application without any code modifications.

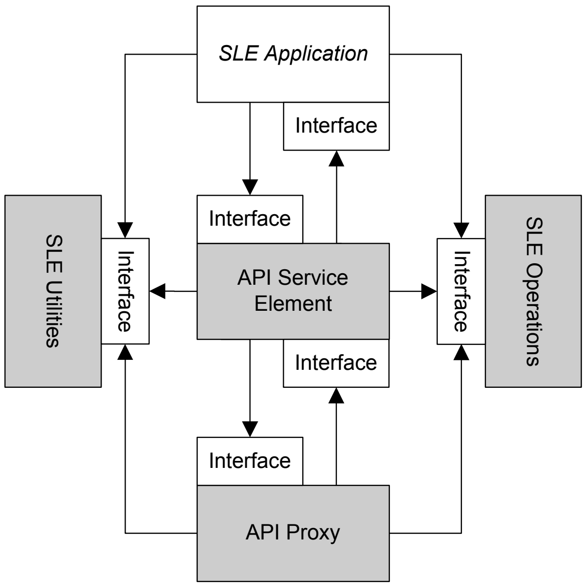

Figure 1: SLE API Components

SLE Forward CLTU Service

The SLE Forward CLTU Service is one of the forward link services included in the SLE specifications. It allows for the transfer of commands from the MDOS to the ground system to be uplinked to the spacecraft. The SLE Forward CLTU Service comprises operations to establish and maintain a communication link from the user to the provider, transfer Command Link Transfer Units (CLTUs) and send/receive status information. Table 1 and Table 2 list a brief description of the operations included in the service.

Table 1: Requests
(operations from user to provider)

Operation	Purpose
CLTU-BIND	To establish an association with the provider
CLTU-UNBIND	To release an association previously established by a CLTU-BIND operation
CLTU-START	To request that the SLE service provider prepare to accept CLTU-TRANSFER-DATA operations
CLTU-STOP	To notify the provider that CLTUs will no longer be sent and to request that any CLTUs in the queue be discarded
CLTU-TRANSFER-DATA	To transfer a CLTU to the service provider
CLTU-SCHEDULE-STATUS-REPORT	To request that the provider send a status report immediately or periodically, or stop reporting
CLTU-GET-PARAMETER	To ascertain the value of an SLE service parameter
CLTU-THROW-EVENT	To forward an event that requires Complex Management to take the actions defined for the event
CLTU-PEER-ABORT	To notify the peer system that the local system detected an error that requires the association to be terminated

Table 2: Responses
(operations from provider to user)

Operation	Purpose
CLTU-ASYNC-NOTIFY	To notify the user of an event affecting production or provision of the Forward CLTU service
CLTU-STATUS-REPORT	To send a status report to the user
CLTU-PEER-ABORT	To notify the peer system that the local system detected an error that requires the association to be terminated

CONTOUR

CONTOUR, a NASA Discovery mission, will visit and study at least two comets. Except during periods of planned hibernation, the spacecraft will be monitored and controlled using NASA/JPL's Deep Space Mission System (DSMS) from the MOC at JHU/APL in Laurel, Maryland, USA. The CONTOUR project accepted the challenge of integrating the SLE Forward CLTU Service into the Mission Operations Center's Ground System. At the time, JPL and the European Space Agency (ESA) were actively designing and testing independent implementations of the SLE specifications in support of the International Gamma-Ray Astrophysics Laboratory (INTEGRAL) mission.[4] While some existing NASA missions are transitioning to this new service provided by the DSMS, CONTOUR will be the first to use it for launch and early operations. CONTOUR is scheduled for launch on July 1, 2002.

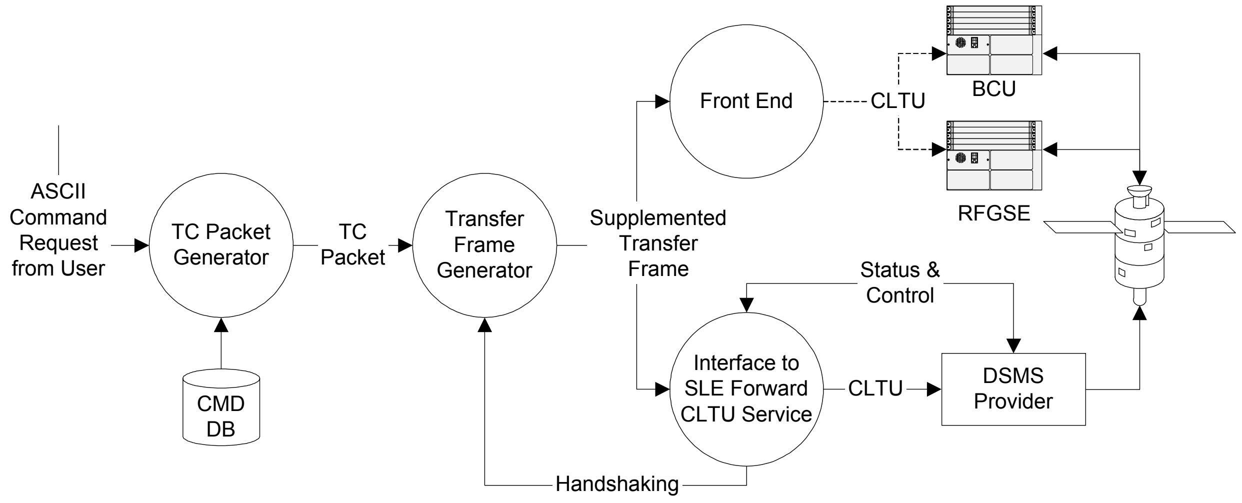

Figure 2: CONTOUR Ground System Command Components

Existing Ground System

The commanding components of CONTOUR's Ground System are depicted in Figure 2. Spacecraft commands entered by the user are constructed via database into TeleCommand (TC) packets. These packets are forwarded to another process that encapsulates them into TC frames. During spacecraft integration and testing, these frames are then sent to a front-end processor, which forwards them to other ground support equipment for uplink via an umbilical or via Radio Frequency (RF) communications to the spacecraft. The front-end processor includes software to convert the TC frame to a Command Link Transfer Unit (CLTU).

Given the existing architecture, it was simple to see that routing the TC frames to another process, instead of the front-end would limit the modifications to the other components and reduce the risk involved in integrating this new functionality. This new application would need to convert the TC frame to a CLTU and act as the user side of a SLE Forward CLTU Service. Except for the addition of a small handshaking protocol to the TC Frame Generator, this isolated the new functionality into one process, allowing essentially independent development and unit testing prior to system integration.

Multi-Mission Requirement

Very early in its design, this new component of the ground system was levied the requirement that it be reusable, with little modification, for future missions. So, not only was the component responsible for integrating the new SLE standard, which would potentially allow it to be used to interface to other agencies for command uplink, it had to be flexible enough to accommodate future missions.

Personnel from both previous and planned missions managed by JHU/APL provided input on which features of the new SLE Forward CLTU Service would be necessary.

Physical Layer Operation Procedure (PLOP)

The PLOP defines the manner in which the actual physical telecommand channel to the spacecraft is activated and deactivated. Two (2) procedures are currently defined: PLOP-1 and PLOP-2. In PLOP-1, a period of unmodulated carrier (no subcarrier) goes out between CLTUs and each CLTU is preceded by the acquisition sequence. In PLOP-2, the carrier is always modulated (subcarrier, with optional idle) between CLTUs. The acquisition sequence is only sent after the first CLTU following unmodulated carrier.

Normally, the selection of PLOP for a mission would effect the configuration of the provider only.[5] However, JPL's implementation for DSMS does not include support for PLOP-1 directly. Their implementation supports PLOP-None, which simply transmits exactly what is received. No acquisition sequence is ever prepended. Thus, to simulate PLOP-1 using PLOP-None, the user-side application would need to prepend the acquisition sequence to each CLTU. The existing front-end code that converted TC frames to CLTUs unconditionally prepended an acquisition sequence to each CLTU and even appended a trailing idle sequence, in addition to the required tail sequence. CONTOUR initially indicated that it would use PLOP-1. To support both current and future missions, both emulated PLOP-1 (via PLOP-None) and PLOP-2 had to be supported.

Constructing a CLTU

A CLTU is composed of a start sequence, encoded telecommand data, and a tail sequence. The telecommand data is encoded into a series of codeblocks. To support future missions, both the size of a codeblock and whether to randomize the data before it is encoded is configurable.

Earliest and Latest Radiation Times

Data for uplink to the spacecraft is transferred to the provider using the CLTU-TRANSFER-DATA operation. Within the request, the invoker may specify a window in which the CLTU should be radiated. The CLTU would be queued at the provider until the specified earliest radiation time and would not be radiated after the specified latest radiation time had expired.

JHU/APL mission operators rarely issue commands to the spacecraft in real-time. Onboard command macros, time-tagged commands and sequenced command loads are utilized extensively. When real-time commanding is necessary or when a software load or routine command load is uplinked, immediate radiation of CLTUs was desired. Thus, neither an earliest nor a latest radiation time would be specified in the CLTU-TRANSFER-DATA.

Inter-CLTU Delay

Additionally, the request to the provider may include an inter-CLTU delay, which specifies the minimum amount of time to wait after radiating the current CLTU, before the next CLTU begins radiation. Initially, there was a vague perception that the capability to specify a minimum amount of time between CLTUs would be useful.

To simulate PLOP-1 using PLOP-None, the provider is configured to generate unmodulated carrier between CLTUs. However, if multiple CLTUs are queued and no earliest radiation time is specified, they will attempt to be radiated contiguously. The capability to specify an inter-CLTU delay proved to be mandatory to emulate PLOP-1, since the CONTOUR command detector needed enough unmodulated carrier (50 bits) between CLTUs to allow it to reset.

Multiple Station Support

During launch, handovers, and critical events, it is common to have multiple ground stations supporting operations. Having the capability to BIND to multiple stations simultaneously is mandatory. Figure 3 illustrates how one process instance interfaces to the existing command system, allows insight into four (4) simultaneously bound stations, and restricts CLTU output to only one of these stations. Using this design, instead of an instance of the process for each station, minimizes the changes to the existing command processes and reduces the complexity by keeping the interface to the Transfer Frame Generator to one and only one connection.

For example, during a handover scenario between two (2) stations and while still bound to the outgoing station, the user can bind to the incoming station prior to Acquisition of Signal (AOS) to assess the status of the provider. A CLTU-START could be invoked to enable commanding. At the time of the handover, the user simply issues a SET STATION directive to the application to route subsequent spacecraft commands to the new station.

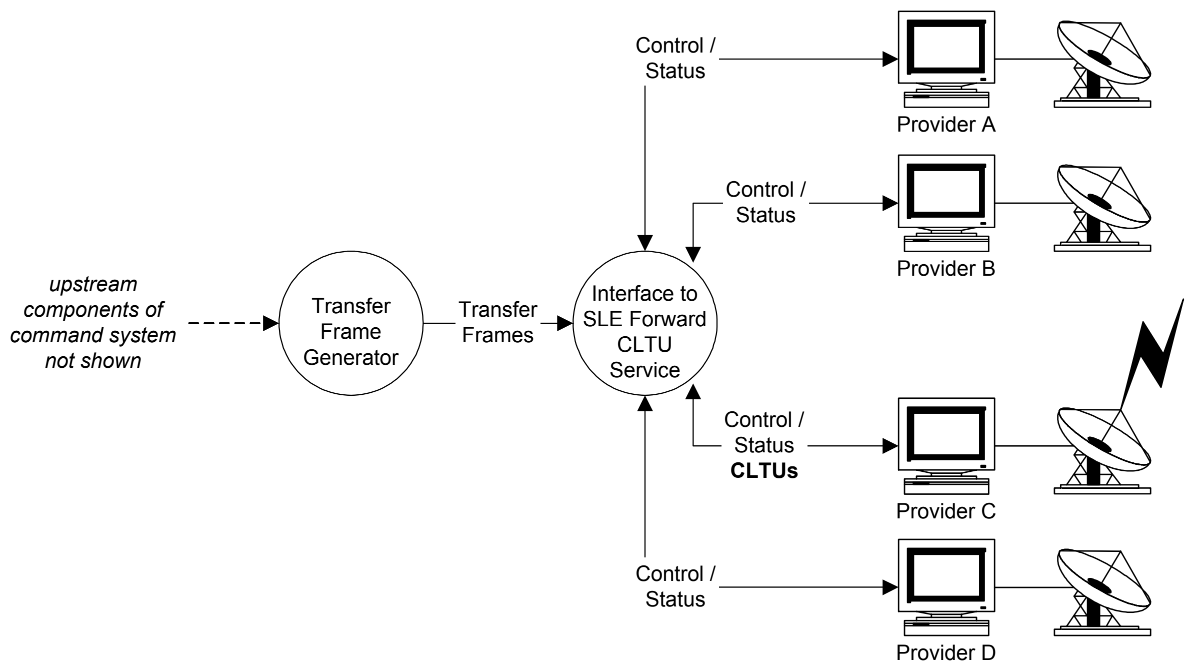

Figure 3: One User Process with Multiple Providers

Issues Concerning End-Users

To ease the transition to this new command protocol, the goal was to provide as much status information as possible in a concise manner, while hiding the details of protocol. End users were involved in the following design choices.

Automated CLTU-START and CLTU-STOP

The CLTU-START operation is used to enable command (CLTU) queuing and transmission. The CLTU-STOP operation both clears the provider's CLTU queue and disables further receipt of CLTUs. The initial design had the application automatically invoking the CLTU-START operation when the first transfer frame was received following a BIND. The CLTU-STOP operation was propagated to a user directive, as the operator needed the ability to clear the CLTU queue at any time. Although there are certain instances when the application knew that a CLTU-STOP was necessary, it was never automatically issued by the application. Unfortunately, this decision implied that the users would need to become familiar with those instances for which the CLTU-STOP operation was mandatory.

Through lengthy discussion, the decision to automatically invoke the CLTU-START operation was ultimately reversed. This too meant that users would need to become more familiar with the protocol. They soon learned that the BIND and START directive were tied and that the SLE service instance state indicator gave them insight into the provider's current ability to accept CLTUs. The main advantage to propagating the CLTU-START to the user was that it allowed a crude locking mechanism. The CLTU-START enabled commanding via the station and the CLTU-STOP disabled this ability.

Flow Control

The SLE Forward CLTU Service does an excellent job of providing up-to-date statistics about the status of service. One key element of this status is the number of bytes available in the queue. However, since CLTUs are of variable length, this cannot correlate to the number of CLTUs that can be sent, but does provide some idea of the current capacity of the queue.

When sending CLTUs to be queued at the station, the application could (1) send and queue all CLTUs that were available up to the maximum capacity of the station's queue; (2) send a CLTU only after the previous CLTU was radiated; or (3) send and queue some CLTUs and then send more as each queued CLTU finishes radiating.

One key goal to choosing an approach to flow control was to eliminate any delay in radiating CLTUs due to network propagation delays. This immediately eliminated the approach of a one-to-one design in which the next CLTU was sent only after the previous one had radiated. This incurred the undesirable effect of two propagation delays – the notification of radiation of the previous CLTU and the transfer of the next CLTU to be queued/radiated.

The disadvantage to queuing as much as possible is the case when a CLTU fails to radiate. When this occurs, the queue must be cleared via a CLTU-STOP and the entire queue reloaded.

The design chosen allowed the number of CLTUs to be queued to be dynamically configurable. The current system sets this parameter to ten (10) CLTUs. Thus at any time, a maximum of ten (10) CLTUs could be queued at the station. As a CLTU is radiated, it is replaced in the queue by a new CLTU, if available. Note that the unit of this parameter is the number of CLTUs, not the number of bytes. If the value of this parameter is not dynamically modified for high data rates and/or when CLTUs are very small, the queue *may* empty quickly; even faster than it can be filled. Nevertheless, this design allows for nearly continuous radiation, as CLTUs are almost always available in the queue for radiation.

This flexible flow control design also allows the other two approaches to be emulated.

Hiding PLOP

If both PLOP-1 and PLOP-2 were directly available in the DSMS implementation, the provider would be configured for either and the MDOS application and user would need no special configuration or insight.

However, PLOP-None adds requirements to what is sent to the provider when it is used to emulate PLOP-1. Specifically, each CLTU must be prepended with the acquisition sequence and an adequate inter-CLTU delay must be included with the CLTU.

Thus to support both PLOP-1 and PLOP-2, the application's configuration must include an optional acquisition sequence and inter-CLTU delay. The design chosen opted not to key off the station's configured plop-in-effect. Instead, the application is configured to prepend an acquisition pattern of a specified length. When set to zero (0), an acquisition sequence is never prepended. So, if a mission chose to use PLOP-2, the application would be configured to use a zero-length acquisition sequence.

The inter-CLTU delay was propagated to the user and made dynamically configurable. In PLOP-2, it is rarely used and remains at zero (0) to allow for continuous radiation of bits. To emulate PLOP-1, it must be set, based on the uplink data rate, to allow the command detector to reset to an inactive state.

Therefore, the details of PLOP are really implemented using configurable parameters whose default values are read from a configuration file when the application starts. While the user can see these values in ground status telemetry, they should be chosen by a mission and frozen prior to system testing.

Command History

When using the front-end interface in the current system, very little status information about the CLTU was available once it was written to the interface for it to be forwarded to the spacecraft.

The new application set out to provide step-by-step status of each CLTU from the time it was received by the application to the time it was radiated.

To accomplish this, the CLTU-SCHEDULE-STATUS-REPORT requests periodic status, usually at the provider's minimum allowable value. The CLTU-ASYNC-NOTIFY, requested in every CLTU-TRANSFER-DATA, provides additional up-to-date information when the CLTU radiates or if it fails to radiate.

Table 3: CLTU Status State Transitions

Start State	End State	Cause of State Change
None	Sent	CLTU-TRANSFER-DATA invoked without error
Send	Ack	CLTU-TRANSFER-DATA returned with positive result
Ack	Cleared	CLTU-STOP invoked and returned with positive result
Ack	Radiate	CLTU-STATUS-REPORT contains 'radiation started' cltu-status for (this) cltu-last-processed
Ack or Radiate	Xmit	CLTU-ASYNC-NOTIFY received with 'cltu radiated' notification-type for (this) cltu-last-OK
Ack	Failed	CLTU-TRANSFER-DATA returned with negative result or CLTU-ASYNC-NOTIFY received with 'sldu expired', 'unable to radiate', or 'production halted' notification type
Radiate	Failed	CLTU-ASYNC-NOTIFY received with 'unable to radiate' notification type

One requirement that simplified this command history capability was that one transfer frame be converted to one CLTU. This allowed a one-to-one mapping of the transfer frame sequence number to the CLTU identifier used and reported by the provider.

CLTUs are tracked through the system using the following states: NONE, SENT, ACK, RADIATE, XMIT, FAILED, and CLEARED. Figure 4 shows the state diagram and Table 3 describes the transitions.

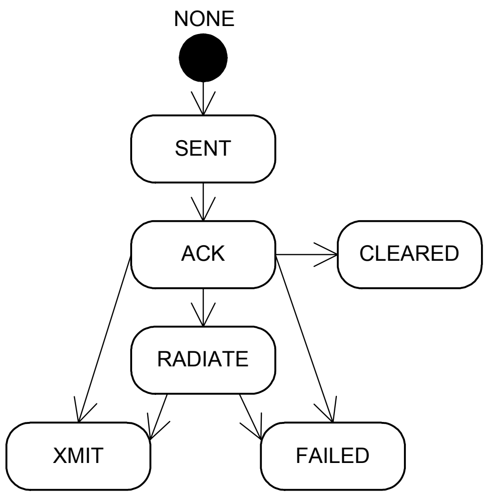

Figure 4: CLTU Status Indicator State Diagram

When available, the actual radiation start time (first bit) and radiation stop time (last bit) of the CLTU are reported. If the CLTU length is large enough and the uplink data rate low enough, the Radiation Start Time is available during the transition to the RADIATE state, as the CLTU-STATUS-REPORT contains that information. The Radiation Stop Time is only available when the CLTU successfully completes radiation. It is nominally obtained from the CLTU-ASYNC-NOTIFY that is received following radiation.

At high data rates and/or with small CLTUs, the transition to RADIATE may not occur. This also depends on the frequency of the CLTU-STATUS-REPORT. In this case, both the Radiation Start and Stop Time are reported on a transition to XMIT, nominally on receipt of the CLTU-ASYNC-NOTIFY following successful radiation.

The CLTU length and an estimate of the radiation time, based on the current uplink data rate, are provided on a transition to the SENT state for each CLTU. Figure 5 illustrates a typical command history.

Collecting and formatting the CLTU status in this manner allows the operator an up-to-date snapshot of the commanding queue – both a picture of the current contents of the provider's queue and what was or will be in that queue.

Propagating Parameters to the User

As described earlier, most SLE Forward CLTU Service operations were propagated to the user. Some parameters for these operations were also propagated.[6]

TF Seq #	CLTU Id	State	Radiation Start Time	Radiation Stop Time	CLTU Len (bytes)	Est. Rad Time (seconds)
159	2123	XMIT	13:26:12.123	13:26:13.123	16	1
160	2124	XMIT	13:26:13.124	13:26:14.124	16	1
161	2125	XMIT	13:26:15.125	13:26:16.125	16	1
162	2126	RADIATE	13:26:16:126		33	2
163	2127	ACK			56	4
164	2128	ACK			514	33
165	2129	ACK			48	3
166	2130	ACK			323	21
167	2131	ACK			87	6
168	2132	SENT			259	17

Figure 5: Example of Command History

Through a series of configuration files, the user issues the BIND directive with only a station-key. The station-key is used to look up the invoker-credentials, initiator-identifier, responder-port-identifier and service-instance-identifier for the CLTU-BIND invocation.

The CLTU-UNBIND always uses the 'suspend' unbind-reason, to allow for a subsequent reBIND without loss of configuration and statistics.

The CLTU-START never specifies the first-cltu-identification. Instead, the CLTU-START is immediately followed by a CLTU-GET-PARAMETER to request the next CLTU identifier to use. This allows for a more consistent numbering scheme since the provider sets and maintains the identifier.

As mentioned above, the cltu-identification used in the CLTU-TRANSFER-DATA is initially retrieved from the provider following a CLTU-START. The identifier advances by flywheeling from the initial value. If a CLTU ever fails to be queued, the user must send a CLTU-STOP, which implies a CLTU-START must later be sent to continue commanding. This will resynchronize the CLTU identifier. As mentioned earlier, the earliest-radiation-time and latest-radiation-time are never specified and the delay-time parameter is propagated to the user to specify. The provider is always requested to produce a report upon completion of CLTU radiation. In fact, this CLTU-ASYNC-NOTIFY is one of the keys to providing flow control.

The user may specify a reporting-cycle for the CLTU-SCHEDULE-STATUS-REPORT. If zero (0) is specified, the 'stop' report-request-type is used.

The CLTU-GET-PARAMETER is not directly propagated to the user. However, the user may issue a REFRESH directive to request the most up-to-date information from the provider and the application. A REFRESH directive triggers both a request for a CLTU-STATUS-REPORT and a set of CLTU-GET-PARAMETERs. These CLTU-GET-PARAMETERs are also automatically invoked after a successful CLTU-BIND. The Forward CLTU parameters currently included are: expected-event-invocation-identification, maximum-cltu-length, modulation-frequency, modulation-index, plop-in-effect, reporting-cycle, and subcarrier-to-bit-rate-ratio.

All event-identifications for the CLTU-THROW-EVENT supported by DSMS's implementation are propagated to the user.

The CLTU-PEER-ABORT is propagated to the user, but is not used in normal operations.

Number of CLTUs Queued

Operators requested that they have a timely indicator of the number of CLTUs currently queued at the station.

The remaining number of bytes available for storage in the queue (cltu-buffer-available) is available in both the return from the CLTU-TRANSFER-DATA and in the CLTU-STATUS-REPORT. If the application kept a record of the length of each CLTU sent to the provider, this statistic might be used to ascertain the number of CLTUs currently in the queue.

Additionally, the CLTU-STATUS-REPORT includes the number of CLTUs received, processed, and radiated. The number of CLTUs that are currently queued could be calculated from these statistics. However, the periodic nature of the CLTU-STATUS-REPORT may not allow this to be updated in a timely fashion.

To simplify the calculation of this indicator and to provide updates to it in a timely fashion, the application book keeps the number of CLTUs queued at the station. This is integrally tied to the command history, which is tracking the CLTUs through the system. The application keeps a list of CLTUs that have been acknowledged by a positive result within the CLTU-TRANSFER-DATA response. CLTUs are then removed from this list when they are successfully radiated, fail to radiate, or are cleared (via a CLTU-STOP). The use of the CLTU-ASYNC-NOTIFY allows for timely update to the length of this list and thus to the indicator of the provider's queue depth.

Lessons Learned

Several valuable lessons were learned in the design, implementation, and testing phases associated with integration the SLE Forward CLTU Service via a single application into CONTOUR's existing ground system.

Design Review

Early in the development of the new application, the design review provided a means for the end users to provide input. The agreement to adopt the SLE Forward CLTU Service meant that the mission operations personnel would have to depart from the protocol they were familiar with, to one that was new – both for them and the DSMS station operators. Including the end-users in the design of the new application proved to smooth this transition.

JPL's SLE API

Absolutely invaluable to the success of integrating this new service was the SLE API provided by JPL. Much more than the proxy layer, the API included almost all the code necessary to implement all layers of the SLE Forward CLTU Service. Except for a handful of SLE application layer interfaces that had to be added, the package of nine (9) libraries included everything necessary for the user side. This alone allowed JHU/APL to focus on how we would integrate it into our ground system in a way that would be simple and reusable.

JPL's Provider Simulator

Further, JPL provided a SLE Forward CLTU Service Provider simulator. After successfully performing a quick compatibility test via the Internet to a JPL hosted provider, further development and testing confidently proceeded using the simulator. As such, for much of development and integration, the heavily utilized DSMS resources were not necessary. The simulator also provided a training ground for end users.

Network Security

One major obstacle that was uncovered when testing commenced using the DSMS test facility was network security. Being an external organization that accesses DSMS facilities, several firewalls, routers, translators and other devices had to be traversed to make the TCP connection. Having the MDOS initiate the connection was a major issue. JPL's SLE API only supported user initiated BINDs, as a TCP client. Placing an additional security device between JHU/APL and DSMS proved undesirable, as the device did not properly provide a seamless gateway at the protocol layer. In retrospect, having a firm idea of the features of the operational physical link that will make the connection from the user to the provider would have allowed addition resources focused on the details of this interface and uncovered this shortcoming earlier in development.

ASN.1 Compiler

An Abstract Syntax Notation One (ASN.1) compiler was necessary to compile and link pieces of JPL's SLE API. To ease integration of the SLE API, the same compiler used by JPL was initially chosen. Due to cost factors, additional compilers were investigated and one chosen to prototype replacement. A majority of the SLE Protocol Data Units (PDUs) was recompiled with the candidate compiler and the API source code modified to integrate their use. The prototype successfully sent and received PDUs to the JPL provider simulator, which retained the original ASN.1 implementation. However, unlike the original compiler, the candidate compiler API functions did not perform full memory management. Due to both schedule constraints and to reduce the risk associated with replacing the compiler, the original implementation was integrated.

<u>Current and Future Uses</u>

Launching on July 1, 2002, CONTOUR will be the first mission to use the SLE Forward CLTU Service during launch and early orbit operations. It is also the first externally managed mission using the DSMS network to operationally utilize this new service. Testing continues with all three DSMS Complexes in Goldstone, Canberra, and Madrid until launch.

Subsequent missions managed by JHU/APL, including MESSENGER, STEREO, and New Horizons, will utilize this same service. The design of the MOC's real-time ground system for these future missions is largely inherited from CONTOUR. The application which integrates the SLE Forward CLTU Service into these ground systems has been designed to be 100% reusable. That is, it has been designed to be completely configurable and should be able to be used for these future missions – unchanged.

References

1. Cross Support Reference Model – Part 1: Space Link Extension Services, CCSDS 910.4-B-1, Washington, DC, May 1996.

2. SLE C++ Application Program Interface for Transfer Services, SLE-SW-API-0001-TOS-GCI, Issue 1.1, European Space Agency, December 2000.

3. Affaitati, F, Martin Götzelmann, and Wolfgang Hell, "Standardised Software Components – A Cost-Effective Approach to Seamless Interoperability." Proceedings of the 2nd ESA Workshop on Commanding, Telemetry, and Tracking Systems for Space Applications, Noordwijk, Netherlands, October 2001.

4. Maldari, P and W. Hell, "CCSDS Space Link Extension (SLE) Cross Support Services, A Challenging Development for INTEGRAL and Future Missions." Proceedings of the 5th International Symposium on Space Mission Operations and Ground Data Systems, Tokyo, Japan, June 1998.

5. Telecommand – Part 1: Channel Service, CCSDS 201.0-B-3, Washington DC, June 2000.

6. Space Link Extension – Forward CLTU Service Specification, CCSDS 912.1-R-2, Washington DC, February 2000.