Apr 30th, 1:00 PM

Paper Session II-B - International Space Station Timeliner System

Bill Dwyer
Mark Rogers
NASA
Richard Berthold
CSDL

Follow this and additional works at: http://commons.erau.edu/space-congress-proceedings

Scholarly Commons Citation
http://commons.erau.edu/space-congress-proceedings/proceedings-1997-34th/april-30-1997/19
The International Space Station Timeliner System
Bill Dwyer, Mark Rogers (NASA), Richard Berthold (CSDL)

Part 1: Introduction and Overview

This paper describes the International Space Station (ISS) User Interface Language (UIL) Timeliner System in three parts. Part 1 provides a high level overview of the Timeliner System, a summary of Timeliner users onboard the ISS, and a short description of the development process adapted due to a number of special problems faced by the development team. Part 2 provides an overview of the Timeliner architecture, characteristics, functionality and capabilities. Part 3 provides an ISS scenario based on a payload utilization of Timeliner.

In the operational ISS environment, the Space Station operations community will be managing the operation of a large number of payloads on a round-the-clock basis, while monitoring the health and safety of the orbiting vehicle. Command and control from the ground systems will be limited by the uplink bandwidth. Additionally, the extended crew stays onboard the ISS coupled with the frequent arrival of new payloads gives rise to special crew training issues. These qualifying factors present a need to change the current Space Shuttle based model for payload and vehicle control that relies heavily on crew manual procedures with the attending extensive ground based training and direct ground management. One solution to these limitations is a flexible, automated procedure execution system that allows the crew and ground engineers to easily monitor and control the procedure execution.

The Timeliner system is composed of three components, a ground based Timeliner Compiler, an onboard Executor and an interface system for the onboard crew and ground engineers. The activity flow for a Timeliner procedure begins with the payload engineer creating the procedure in the English-like User Interface Language (UIL). This procedure is then compiled by the Timeliner Compiler and uplinked to the onboard mass storage. The procedure is then loaded into one of the onboard Timeliner Executors and started by a command from the crew, ground or some onboard procedure control system such as the ISS Onboard Short Term Plan. The procedure execution is then monitored and controlled by the crew or ground engineers. This control includes starting, stopping, single stepping, and skipping to a different procedure line.

There are two Timeliner Executors onboard the U.S Segment of the ISS. One is hosted on the Command and Control Multiplexer-demultiplexer (MDM) and one hosted in the Payload Executive Processor MDM. The Executor in the Command and Control MDM executes procedures for the ISS avionics control systems; the Executor in the Payload Executive Processor executes procedures for the ISS payloads. A Timeliner Compiler at Johnson Space Center (JSC) provides the compiled procedures for the avionics system and a Timeliner Compiler at Marshall Space flight Center (MSFC) provides the procedures for the payload systems. Both the avionics and payload control Timeliner Executors share the ISS Portable System (PCS) for the crew interface displays.

The two Timeliner Executors in the U.S. Segment are identical. The core of the two Timeliner Compilers are also identical, with each ground facilities providing different database interfaces to the Timeliner Compilers, driven by their separate unique requirements and operational practices.

There is one additional Timeliner System onboard the ISS. There is an Executor hosted in the Japanese Experiment Module Control Processor (JCP). Because the JCP is a different processor then the U.S. MDMs, this Executor is slightly different from the U.S. Segment Executors. The Japanese Timeliner Compiler will be at Tsukuba Space Center, Tsukuba, Japan. However, the UIL language and Timeliner capabilities are the same, thus providing a degree of interoperability in a key operational area.

As can be noted from the above paragraphs, several ISS development and operational organizations had a vested interest in the development of the Timeliner System. For the U.S. Segment, this included the ISS Boeing Prime Contractor, Boeing-Huntsville, Alabama, (Product Group-3), the JSC Mission Operations Directorate (MOD), systems operations engineers for the definition of the avionics Timeliner
procedures, the MSFC Mission Operations Laboratory (MOL), payload operations engineers for the definition of the payload procedures for Timeliner, the JSC Engineering Directorate for the development of the PCS, Draper Laboratory, the Timeliner developer, the NASA ISS Program Office, as well as NEC and NASDA for the Japanese version of Timeliner. Each of these organizations and functions levied requirements on the capabilities and functionality of Timeliner.

In addition to coordinating and managing the inputs from the large number of organizations interested in the development of Timeliner Development, the Timeliner System developers faced unique schedule integration problems. The Timeliner development originally started during the Space Station Freedom Program, and thus the ISS Timeliner development lifecycle was running early in the overall ISS Software development lifecycle. For example, a Timeliner System Critical Design Review was held before the Command and Control Software System Requirements Review. Thus, Timeliner development was in need of detailed interface information before the ISS Command and Control System had defined its functions and requirements. To overcome these problems, Timeliner development project implemented two solutions: instead of the standard “waterfall” software development lifecycle, Timeliner development implemented a “phased evolutionary spiral” development process; secondly, an adhoc “team” was formed to focus sharply on the definition of the Timeliner interfaces for both the onboard Executor and the ground-based Compiler. This “team” included membership from each of the organizations listed above, with Draper Labs representing the Japanese, and became a defacto Integrated Product Team (IPT). The ISS NASA Program Office, the Boeing Prime Contractor, and the development organizations cooperated fully with the new IPT. Draper Labs took the lead role in writing a “white paper Interface Control Document (ICD)”. The team, which took the name “Timeliner Roundtable”, met at JSC, MSFC and Draper Laboratory, on a rotating basis and successfully completed the detailed interface definitions, to the level of complete Ada specifications. Concurrently, the “phased evolutionary spiral” development process permitted three complete requirements-design-implementation processes, each generating a complete product. This allowed the Timeliner functionality to converge to the evolving Command and Control system development.

The interface “white papers” have now been merged into the Timeliner Software Detailed Design Documents and the appropriate ISS flight and ground systems design and requirements specifications. Thus, it can be said that the Timeliner Roundtable was “an IPT that worked”.

Part 2: The Timeliner System

Historically, Timeliner was created to emulate the timelines for onboard crew procedures followed by the crew of the Space Shuttle. It was used as a simulation driver in tests of the Space Shuttle system, mimicking crew actions in monitoring and controlling the spacecraft systems. The Timeliner simulation system was extended for use in other applications, and was tailored to provide real-time procedure execution and support interactive control commands that might be entered by the systems engineer: for example, procedure start and stop.

In 1992, Timeliner was selected by NASA as the User Interface Language (UIL) for Space Station Freedom, and later for International Space Station (ISS). Since that time, Timeliner has evolved as a modular, extensible system that allows procedures to be developed and executed in virtually any systems environment, can be applied to control a variety of target systems and meet a wide range of mission objectives.

In general, the Timeliner system lowers the workload in the performance of mission or process control operations. In a mission environment, Timeliner procedures can be used to automate spacecraft or aircraft operations including autonomous or interactive vehicle control, performance of preflight and post-flight subsystem checkouts, or handling of failure detection and recovery. Timeliner may also be used for mission payload operations, such as stepping through predefined procedures of a scientific experiment. Other applications may include process control for manufacturing, materials processing, robotics, or any automated procedures
that can be clearly defined by a systems specialists (rather than software engineers) and need to be executed reliably, insuring repeatability.

The Timeliner System is characterized by three key aspects, a.) the Timeliner language, which provides the ability to write system control 'scripts' using a system-level English-like language; b.) the Timeliner development and execution environments, which allows these scripts to be developed, compiled and executed to monitor and control a specific target system, without the need for system rebuild and; c.) the monitoring and control environment, which allows interactive operator tracking and control of the execution of these compiled scripts.

The Timeliner language is specifically intended as an operational language for system specialists and operations engineers. The keywords of the Timeliner lexicon are "operational" in their meanings thus allowing the system specialist or operations engineer to easily write a script for the operational procedure desired. The complete Timeliner language for ISS is specified in the NASA document SSP 30529, User Interface Language Specification.

The Timeliner script may contain Timeliner "statements" that, when compiled and executed, cause actions to be taken on the basis of time or on the basis of other real-time events or conditions. Timeliner statements are grouped together to form a sequential set of actions called "sequences." Timeliner sequences, which logically execute in parallel, can be grouped together to form "bundles." Figure 1 illustrates the Timeliner script organization and its control paradigm.

![Figure 1: Timeliner Script Organization](image)

Timeliner sequences contain instructions for monitoring, controlling and reporting on the operations of a target system. The Timeliner language provides statements for organizing and performing the instructions. There are six general types of statements:

- Block declaration statements -- these organize the bundles, sequences, and subsequences of Timeliner (e.g. BUNDLE, SEQUENCE).
- Timing control statements -- these affect timing or flow of execution (e.g. EVERY, WAIT, CALL)
- Conditional control statements and their modifier clauses -- these allow for specific conditions that control execution based on general system values (e.g. WHEN; WHENEVER with modifiers BEFORE, WITHIN, and OTHERWISE; and IF-THEN-ELSE).
• Actions statements -- these are used to carry out actions affecting the target system and supports interaction with the mission operator (e.g. COMMAND, SET, MESSAGE, MESSAGE/PAUSE).

• Bundle/Sequence Control statements -- these are equivalent in function to those indicated in Figure 2 to allow one sequence to interact and control other sequences (e.g. INSTALL, REMOVE, START, STOP, RESUME, STEP).

• Non-executable statements -- these are used for definitions of symbols and reserving of local storage (e.g. DEFINE, DECLARE).

The Timeliner Compiler and Executor allow the scripts to be defined, written, compiled into executable sequences, verified, and executed in a rapid, 'roll-in' fashion, independent of other software that might be executing in the same Ada main program or processor. The uplinking and loading of a Timeliner procedure into the Executor is also non-disruptive to any other software executing in the same Ada main program or processor. This independence is possible because the memory addresses that support the Timeliner data reads and writes in the execution environment are resolved by the Timeliner Compiler on the ground and the execution binding is accomplished at runtime.

The Timeliner Compiler system architecture provides several key abstractions. The procedure script definition is provided to the Compiler as a simple ASCII text file, and therefore produced from any text editor, word processor, or planning system. The Compiler also supports independent definition of system data, object and command formats, types and names definitions (names to be used in the script). In this way, the system definition database may be developed, maintained, and provided for script compilation independent of the Timeliner Compiler environment software build. This allows script algorithmic development in abstraction from detailed system data formats. In addition the Compiler provides 'cross-platform' compilation that allows the Compiler and Executor to be platform independent.

The Timeliner Executor system architecture also provides several key features. As indicated earlier, a compiled bundle may be 'rolled-in' or INSTALLed, executed, and REMOVED independent of execution environment software. In addition, the Executor supports execution and independent control of multiple bundles, in parallel, which may itself contain multiple sequences that execute in parallel (either in synchronous or asynchronous fashion) and also can be independently controlled. For the ISS Timeliner, the Executor software also provides several key Command and Data Handling (C&DH) roles necessary for the ISS system architecture (e.g. bundle memory management, sequence execution time management, input and output command and message queuing, multiple machine format system data conversion).

As mentioned previously, the Executor supports several unique features to monitor and control executing scripts. The Executor in concert with the Crew displays provides the ability to precisely track, view, annotate, and interactively control an executing Timeliner procedure. Figure 2 shows the main Sequence Control display provided for the ISS crew.
Through the ISS Timeliner displays, the crew can monitor the status of an executing procedure, and receive crew messages from the executing script. The message capability provides a key link between the autonomous and manual portions of a crew operation. For example, automated commanding (e.g., TURN ON, SELECT GAS, IGNITE, TURN OFF) and automated system monitoring (e.g., WHEN POWER IS ON, WHenever PRESSURE > ACCEPTABLE RANGE, IF DIAGNOSTIC_STATUS = FAILED) can be sequentially integrated with a crew MESSAGE to perform a manual operation (e.g., MESSAGE/PAUSE “Please replace gas filter 1, then Resume sequence”). This interactive utility will be further exemplified in the next section.

**Part 3: Timeliner Utilization for ISS Payload Operations**

The concept for the Timeliner system for payload operations in the US segment is to provide the payload experiment teams a service of automated command and control and to ensure time critical commanding is accomplished. The satellite coverage for ISS has intermittent dropout periods when there is no data transmission to or from the ISS.

Timeliner is the solution for seamless operations for payloads and ground controllers to operate time critical experiments by ensuring that the commands reach the payload at the desired event or time.

The Timeliner executor for US payloads is located in the Payload Executive Processor MDM. Timeliner procedures are built and tested by the Payload developer and stored at MSFC in the Payload Information Management System (PIMS). During an ISS operational increment, the time period when the onboard payload configuration is stable, payloads that use Timeliner procedures will have the Timeliner files loaded and ready for use. The payload control center will provide the payload developer the capability to edit the Timeliner procedures as the experiment is in flight. The edit function allows an experimental environment of receiving science data and adjusting the procedures in near real-time to accomplish the goals of the payload.
The payload developer will build Timeliner procedures to operate the payload that will run several steps in an experiment (see Figure 3). The payload developer will write the Timeliner procedure in a standard ASCII text editor and through the Huntsville Operations Support Center (HOSC)-PIMS interface compile the text to an onboard format. The payload developer can store the compiled procedures in the PIMS system for uplink onboard ISS at a later time.

The following paragraphs and illustration delineate a scenario for a specific experiment, the U.S. Payload Furnace Experiment.

The payload developer will build several Timeliner procedures to accomplish the general control and operation of the experiment. The one in this scenario is a payload experiment run that is controlled by a Timelier procedure (see Figure 4. Sample Timeliner Procedure).

The Timeliner procedure will 1. turn on the payload 2. start an internal diagnostic test 3. heat the chamber to a pre-defined temperature 4. instruct the crew member to load sample material 5. heat the sample for a specific time 6. cool down 7. alert the crew member to retrieve and log the material 8. deactivate and shut down the facility.

The Timeliner procedure can be initiated by the ground controllers via commanding, the crew from the Portable Computer System (PCS), or the Onboard Short Term Plan (OSTP). The Timeliner procedure, once started, will automatically step through the execution of the experiment.

The sample Timeliner procedure shows the format of the procedure that the crew member will see on the PCS display. The Timeliner procedure will run automatically, wait for crew input, and continue. The goal of the Timeliner procedure is to save the crew as much time as possible in daily operations of payloads and to ensure time critical commanding. Messages to the crew are sent from the Timeliner procedure to instruct the crew when an experiment has reached the proper state or event that a crew action is needed to continue operation. The ground controllers can monitor the execution of the Timeliner procedure and assist or command the experiment if needed.

Another type of Timeliner procedure is one that the OSTP or ground controllers will start and no crew interaction is needed. These type of procedures are autonomous and run in the payload MDM to conduct a payload experiment.

The procedure can read data from the Current Value Table (CVT) memory of the payload MDM to determine if an external event has occurred (example: If oven temp is less than 10 deg). Also, the Timeliner procedure can read and act on internal data or logic. In either case,
reading internal or external data the Timeliner procedure can be developed to accomplish the needs of the payload experiment teams.

The Timeliner capability allows payload developers to integrate into the ISS program without the need to build and operate a separate processor for the operation of the payload. The services of the Timeliner system and payload MDM give the experiment teams a powerful and cost effective environment to conduct payload operations onboard ISS.

```plaintext
BUNDLE FURNACE_EXPERIMENT "U.S. PAYLOAD FURNACE EXPERIMENT-142b"
SEQUENCE MASTER
-- This sequence will provide major sequence control for experiment, parallel sequence -- will monitor furnace for acceptable operating ranges.
DEFINE HEAT_DURATION AS 00:04:00
START THERMOSTAT
COMMAND Power_On_Heater, Init_Temp=>20
COMMAND Perform_Heater_Diagnostics
WHEN Heater.Diagnostics_Complete = TRUE WITHIN 10 CONTINUE
OTHERWISE
    MESSAGE "HEATER DIAGNOSTICS FAILED TO COMPLETE IN TIME, EXPERIMENT ABORTED"
    CALL HEATER_OFF
    STOP
END
IF Heater.Diagnostics_Status = FAILED THEN
    MESSAGE "HEATER DIAGNOSTICS FAILED, EXPERIMENT ABORTED"
    CALL HEATER_OFF
    STOP
COMMAND Set_Heater_Temperature, Temp=>40, Mode=>IMMEDIATE
MESSAGE "PLEASE LOAD SAMPLE FOR EXPERIMENT 142b, RESUME WHEN READY" PAUSE
COMMAND Set_Heater_Temperature, Temp=>90, Mode=>LINEAR, Time=>00:01:40
WHEN Heater.Temperature within (89.5..90.5) WITHIN 00:01:45 CONTINUE
OTHERWISE
    MESSAGE "HEATER PRECISION FAILURE, NEEDS RECALIBRATION, EXPERIMENT ABORTED"
    CALL HEATER_OFF
    STOP
END
WAIT HEAT_DURATION
COMMAND Set_Heater_Temperature, Temp=>20, Mode=>IMMEDIATE
WHEN Heater.Temperature within (19.5..20.5) CONTINUE
CALL HEATER_OFF
MESSAGE "U.S. PAYLOAD FURNACE EXPERIMENT-142b COMPLETE"
CLOSE SEQUENCE
SEQUENCE THERMOSTAT
DEFINE Thermostat_Outside_Accetable_Range AS Heater.Temperature OUTSIDE 0..100
WHENEVER Thermostat_Outside_Accetable_Range
    MESSAGE "HEATER TEMPERATURE DETECTED OUTSIDE ACCEPTABLE RANGE, SHUTTING DOWN"
    CALL HEATER_OFF
END
CLOSE SEQUENCE
SUBSEQUENCE HEATER_OFF
COMMAND Power_Off_Heater
WHEN Heater.Status = OFF CONTINUE
MESSAGE "LABORATORY HEATER TURNED OFF"
CLOSE SUBSEQUENCE
CLOSE BUNDLE
```

Figure 4: Sample Timeliner Procedure