To appear in 16th International Conference on Space Operations (SpaceOps 2020)

Integrated Mission Operation Concepts for the *Dream Chaser*® Cargo System

SpaceOps

May 2021

Jason Gabbert – Sierra Nevada Corporation

Madeline Devereaux – Sierra Nevada Corporation

Jeremy Owen – Sierra Nevada Corporation

David Kortenkamp – TRACLabs Inc.

Scott Bell – TRACLabs Inc.

Gilles Kbidy – L3Harris

Introduction

Schedules, procedures, commands, and telemetry are at the heart of operating complex spacecraft. Schedules dictate when to do something, procedures tell operators how to do it, and the command and telemetry system enables both. In many space operations, procedures and schedules are stove-piped with little or no information exchanged between largely manual systems.

In the work presented in this paper, procedures, schedules, real-time telemetry, and commanding are integrated using modern distributed computation approaches and commercial off-the-shelf (COTS) solutions with the result applied to Sierra Nevada Corporation's *Dream Chaser*® spaceplane operations servicing NASA's International Space Station (ISS) in support of the Cargo Resupply Services 2 (CRS2) contract.

The system consists of an electronic procedure system which connects directly to a web-based scheduling system, as well as spacecraft telemetry and commands. The scheduling system can initiate procedures and track their status. Operators developing schedules have electronic access to all procedures with estimated durations based on prior execution. These interfaces are provided by web-based technologies such as REST and JSON. Operator interfaces to the schedule and electronic procedures are also web-based, which allows for simultaneous sharing of information across all operators and access from any device with a web-browser. The result is more efficient operations with less manual labor overhead and fewer errors.

This paper will present the motivation, an overview of the technologies, how each is integrated, and describe the integrated use case.

Traditional Space Operations Tools

The authors have experience operating several spacecraft in multiple organizations ranging from very large, well-funded programs like ISS and Space Shuttle, to university space programs working with minimal resources and staff. Some common themes persist across these programs and organizations and are likely common to spacecraft operations.

To date, each organization has independently developed their own suite of tools over time to support the following functions:

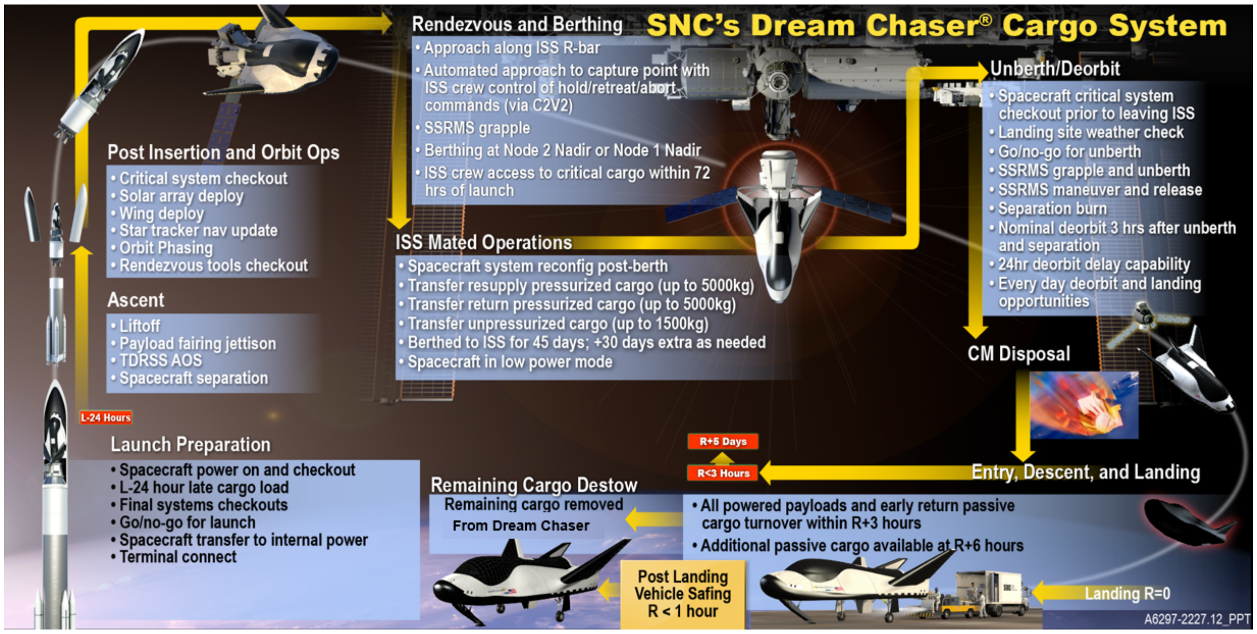
- Planning and scheduling
- Constraint checking
- Commanding tools and products
- Spacecraft and operational procedures
- Real-time telemetry display
- Historical telemetry analysis

These tools are often designed in-house at great cost and lengthy development times and sometimes in isolation from one another, resulting in unique interfaces to be mastered and requiring manual correlation between products and command and telemetry systems. On satellite programs, scheduling tools are commonly used more for mission planning than real-time execution and status tracking. There has also been no linkage of command timing or current telemetry values to human-centric procedures. Operators regularly end up relying on paper-based procedures to make product review and execution easier; however, these products become nearly impossible to sort and search for as-run notes as the scale of operations grows. Finally, consistent formatting of products has been manually enforced, requiring additional labor to maintain. Due to the high costs and development time associated, these highlyintegrated tool suites are typically reserved only the largest of space operators; therefore, a fully integrated, easy to learn, commercially available system attainable by most space operators was sought.

Dream Chaser

Sierra Nevada Corporation's (SNC) Space Systems Group is leading an effort to create a low-cost space system to support the transport of cargo to and from low-Earth orbit, including the ISS. This effort called the *Dream Chaser*® Program is focused on the development of the Dream Chaser Cargo System.

The Dream Chaser Cargo System consists of the uncrewed Dream Chaser and Cargo Module (CM). The Dream Chaser is a reusable, autonomous, lifting-body spacecraft that launches vertically on top of a launch vehicle and lands horizontally on a conventional runway. The CM is an expendable module that attaches to Dream Chaser serving as a platform for solar arrays and thermal radiators and carries pressurized and unpressurized cargo. The CM is jettisoned for atmospheric burnup prior to Dream Chaser entry.


SNC's Dream Chaser® Cargo System
Rendezvous and Berthing
• Approach along ISS R-bar
• Automated approach to capture point with ISS crew control of hold/retreat/abort commands (via C2V2)
• SSRMS grapple
• Berthing at Node 2 Nadir or Node 1 Nadir
• ISS crew access to critical cargo within 72 hrs of launch
Post Insertion and Orbit Ops
• Critical system checkout
• Solar array deploy
• Wing deploy
• Star tracker nav update
• Orbit Phasing
• Rendezvous tools checkout
Ascent
• Liftoff
• Payload fairing jettison
• TDRSS AOS
• Spacecraft separation
L-24 Hours
Launch Preparation
• Spacecraft power on and checkout
• L-24 hour late cargo load
• Final systems checkouts
• Go/no-go for launch
• Spacecraft transfer to internal power
• Terminal connect
ISS Mated Operations
• Spacecraft system reconfig post-berth
• Transfer resupply pressurized cargo (up to 5000kg)
• Transfer return pressurized cargo (up to 5000kg)
• Transfer unpressurized cargo (up to 1500kg)
• Berthed to ISS for 45 days; +30 days extra as needed
• Spacecraft in low power mode
Unberth/Deorbit
• Spacecraft critical system checkout prior to leaving ISS
• Landing site weather check
• Go/no-go for unberth
• SSRMS grapple and unberth
• SSRMS maneuver and release
• Separation burn
• Nominal deorbit 3 hrs after unberth and separation
• 24hr deorbit delay capability
• Every day deorbit and landing opportunities
CM Disposal
Entry, Descent, and Landing
R+5 Days
R<3 Hours
Remaining Cargo Destow
Remaining cargo removed From Dream Chaser
• All powered payloads and early return passive cargo turnover within R+3 hours
• Additional passive cargo available at R+6 hours
Post Landing Vehicle Safing R < 1 hour
Landing R=0
A6297-2227.12_PPT
Figure 1. - Dream Chaser CRS2 Mission Overview

Launching inside a standard payload fairing requires the wings and solar arrays to be folded for launch and deployed once on orbit. Following a series of checkouts, the Dream Chaser rendezvous with the ISS where it is captured by the ISS robotic arm and berthed to one of the ISS nodes. The ISS crew then ingresses the vehicle and exchanges cargo and payloads. External, unpressurized cargo is transferred using the ISS robotic arm. Once the berthed mission is complete, the Dream Chaser is unberthed and released from the ISS and prepares for entry, descent and landing. The Dream Chaser executes this phase autonomously after the authority to proceed is provided by the flight controller team, landing at the Shuttle Landing Facility (SLF) runway at NASA's Kennedy Space Center. Ground crews then retrieve the payloads from the Dream Chaser and quickly return them to the scientific teams.

As a commercial aerospace program, SNC has extra incentive to develop tools that enable streamlined operations, keeping labor requirements and overhead costs to a minimum. The SNC Space Systems team recognized mission planning and execution for the Dream Chaser Cargo System spacecraft presented both unique challenges and great opportunities to rethink and improve upon the traditional operations concepts.

OnTime

The OnTime scheduling platform was created to provide a web-based, distributed, and collaborative planning and scheduling solution for the Dream Chaser mission, capable of real-time integration with the procedure execution and command and control (C2) systems.

Running in the cloud or on premises, OnTime lets operators and planners coordinate activities for multiple vehicles (Dream Chaser, ISS), optimize communication paths across dedicated or shared data links, evaluate resource availability, and coordinate ground staff activities at multiple locations. The application is typically deployed as a set of instances configured for sandbox development, training, mission simulation planning and execution, and most importantly real-time mission planning and execution. OnTime instances can be one of three types: planning, live, and playback. A planning instance allows collaborative creation and editing of multiple timelines. It also includes activity templates and rules to standardize scheduling across multiple timelines. A live instance allows real-time execution and statusing of executing activities. The playback instance allows real-time or accelerated replay of archived as-run timelines. Timelines are captured as JSON documents which can easily be exchanged from one platform or instance to another. In addition, a built-in dynamic publishing capability allows planners to easily "push" changes between instances as needed (for regular publishing of approved timelines, contingency operations, or just-in-time updates).

Key differentiators for the Dream Chaser mission include: 1) ability to perform concurrent editing on shared timelines from distributed locations ("Google docs for timelines"), 2) live integration with procedure and C2 systems, 3) near real-time constraint checking, conflict resolution and use of planning AI algorithms to optimize scheduling allocations.

PRIDE

The PRIDE electronic procedure platform represents standard operating procedures as an eXtensible Markup Language (XML) file that can both be displayed to an end user in a web page and be executed by an autonomous system. The XML representation, called the Procedure Representation Language (PRL), was derived from and augments the NASA International Space Station (ISS) International Procedure Viewer (IPV) representation.

The PRIDE electronic procedure platform consists of several key components: 1) a drag-and-drop procedure authoring tool; 2) a web-based procedure viewer; and 3) a procedure executive for automation. PRIDE stores all procedure interactions, whether by operators or automation, in a SQL database for auditing and analytics. Thus, after the PRIDE system has been running significant amounts of data will be available with respect to what procedures were run, how long they took, whether they were successful or not, and which vehicle did what procedures when.

PRIDE has a REST Application Programmers Interface (API) to provide information such as procedure content and status. The API can be used to provide a list of available procedures, which can be then used when creating OnTime activities and can provide a unique URL to each procedure (the *static* procedure link). The API can also be used to start any procedure either in manual or automated mode. Once a procedure is started, the API can be used to get a unique URL to that particular instance of the running procedure (the *dynamic* procedure link). OnTime can then replace the static link with the dynamic link after the procedure has started so that any operator clicking on the dynamic link will join the running procedure. The API can also provide an update as to the execution status of the running procedure, including percentage complete (based on number of instructions), estimated time to complete (based on prior history), and current status (running, failed, paused, finished, aborted).

InControl

The C2 system selected for Dream Chaser Cargo System is based on the InControl product which was also enhanced to provide a REST API to easily integrate with PRIDE and OnTime. Vehicle operations requiring specific procedures and commands can be planned in OnTime, executed as discrete steps in PRIDE and seamlessly injected in the C2 system to execute pre-checked command sequences. The execution status is then returned to PRIDE including telemetry values and overall procedure execution status can be tracked in OnTime at a very high-level.

Using loosely coupled interfaces and modern data-centric technologies was crucial early on in the project to quickly integrate the tools (less than a day in each case for proof-of-concept) and validate the overall technical approach. The next section provides insights into the data architecture and lessons learned along the way.

Integrated Architecture

The three applications (OnTime, PRIDE and InControl) are installed, managed and configured independently. They each have their own persistent storage layer and software Graphical User Interfaces. This level of independence provides a very flexible solution for mission planners, procedure developers and spacecraft operators, who can all work in parallel and release new work products in each tool as they become available.

The integration and automation aspect is achieved using loosely coupled REST APIs, combining typical request-response and polling patterns. All three tools were inherently designed with this ability. As soon as a new spacecraft command script is available, it is automatically added to PRIDE and can be used in existing or new procedures. Similarly, as soon as new PRIDE procedures become available, they are automatically added to OnTime and can be used in existing or new mission plans. The data sent back and forth over REST is formatted in JSON, which is a natural fit for loose coupling. The architecture diagram below illustrates the loose coupling concept and the independence of the three applications.

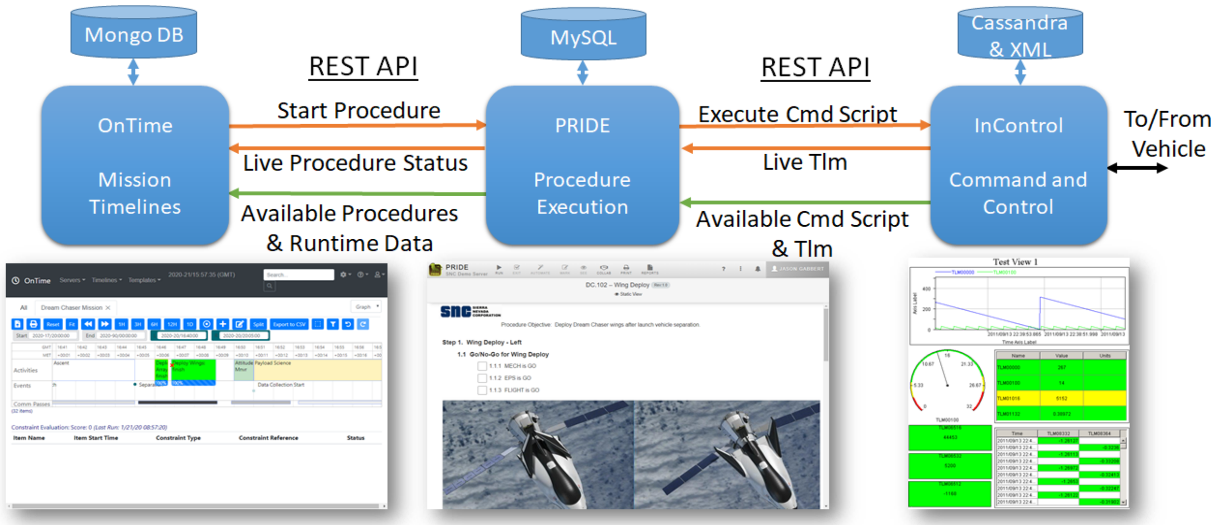


Figure 2: Application Interface Architecture

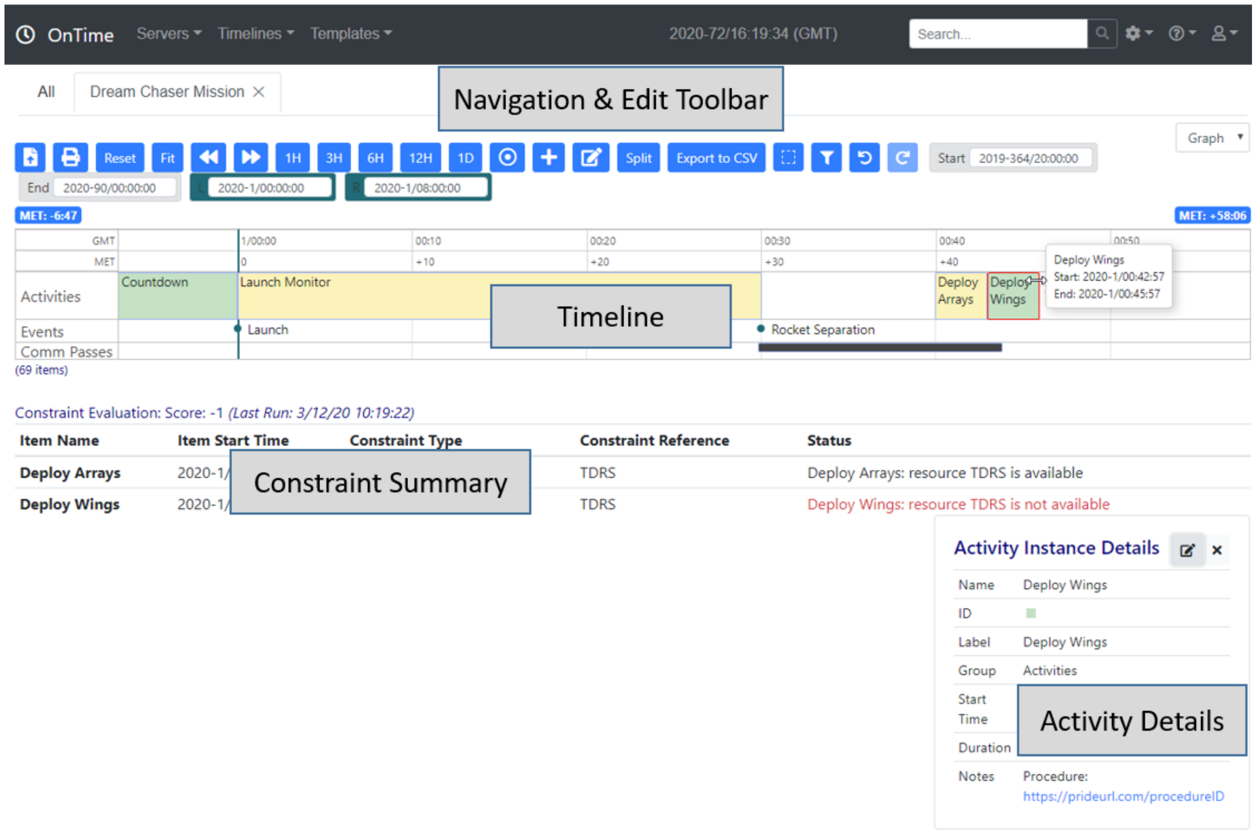
The benefits of such an approach (REST API and JSON) include:

1. Ability to change an existing REST API endpoint without breaking existing API consumers. Of course, API designers must use caution and only augment the API with additional, optional arguments. This approach works extremely well and lets different vendors or providers release new versions of the API safely, without breaking existing interfaces
2. Using JSON also provides a means to easily augment content without breaking existing consumers. Additional key-value pairs (including complex data types such as arrays and embedded objects) can safely be added to the API.

The development team observed a huge productivity increase compared to a more traditional approach. On previous programs, it was often necessary to rely on interfaces based on CORBA, SOAP, RMI or other more rigid, tightly coupled definitions. The scope of such an integration would have typically taken multiple weeks. Using REST and JSON, the team was able to drastically reduce integration time and achieve full automation within a few days.

Mission Planning and Scheduling

Mission planning begins with scheduling activities across a pre-defined timeframe to meet various resource and vehicle constraints. The pre-mission part of the planning cycle involves creating and maintaining a full mission timeline as mission needs and procedures are defined. During mission execution, sections of the timeline are regularly refined and updated with the latest resource schedules, procedures, and results from previously executed activities. **Figure 3** shows the web-based OnTime planning view where users schedule activities and events on a timeline from pre-defined templates or as one-off items. Items are added using the *navigation & edit toolbar* and assigned groups that are organized as rows on the *timeline*. Resources such as communications schedules and trajectory information such as orbit events are imported into an OnTime timeline from a JSON or CSV file. Using AI scheduling scripts, activity placement can also be automated and optimized based on user-defined rules. As items are added to a timeline, real-time constraint checking is performed and reported in the *Constraint Summary.* Constraints are then resolved or acknowledged as an exception.



Constraint Evaluation: Score: -1 *(Last Run: 3/12/20 10:19:22)*

Item Name	Item Start Time	Constraint Type	Constraint Reference	Status
Deploy Arrays	2020-1/		TDRS	Deploy Arrays: resource TDRS is available
Deploy Wings	2020-1/		TDRS	Deploy Wings: resource TDRS is not available

Activity Instance Details

Name	Deploy Wings
ID	🟩
Label	Deploy Wings
Group	Activities
Start Time	
Duration	
Notes	Procedure: https://prideurl.com/procedureID

Figure 3: Planning View

Timelines are stored in a fault tolerant and shared database for access across various instances. Once a timeline is approved and ready for execution, part or all of the timeline is published to a live instance where the REST interface allows procedures to autonomously or manually execute. The application can be configured to run multiple instances to support all phases of the mission: planning, simulation (with simulated and accelerated time), live execution, and playback (for post-mortem analysis if desired).

Authoring Procedures

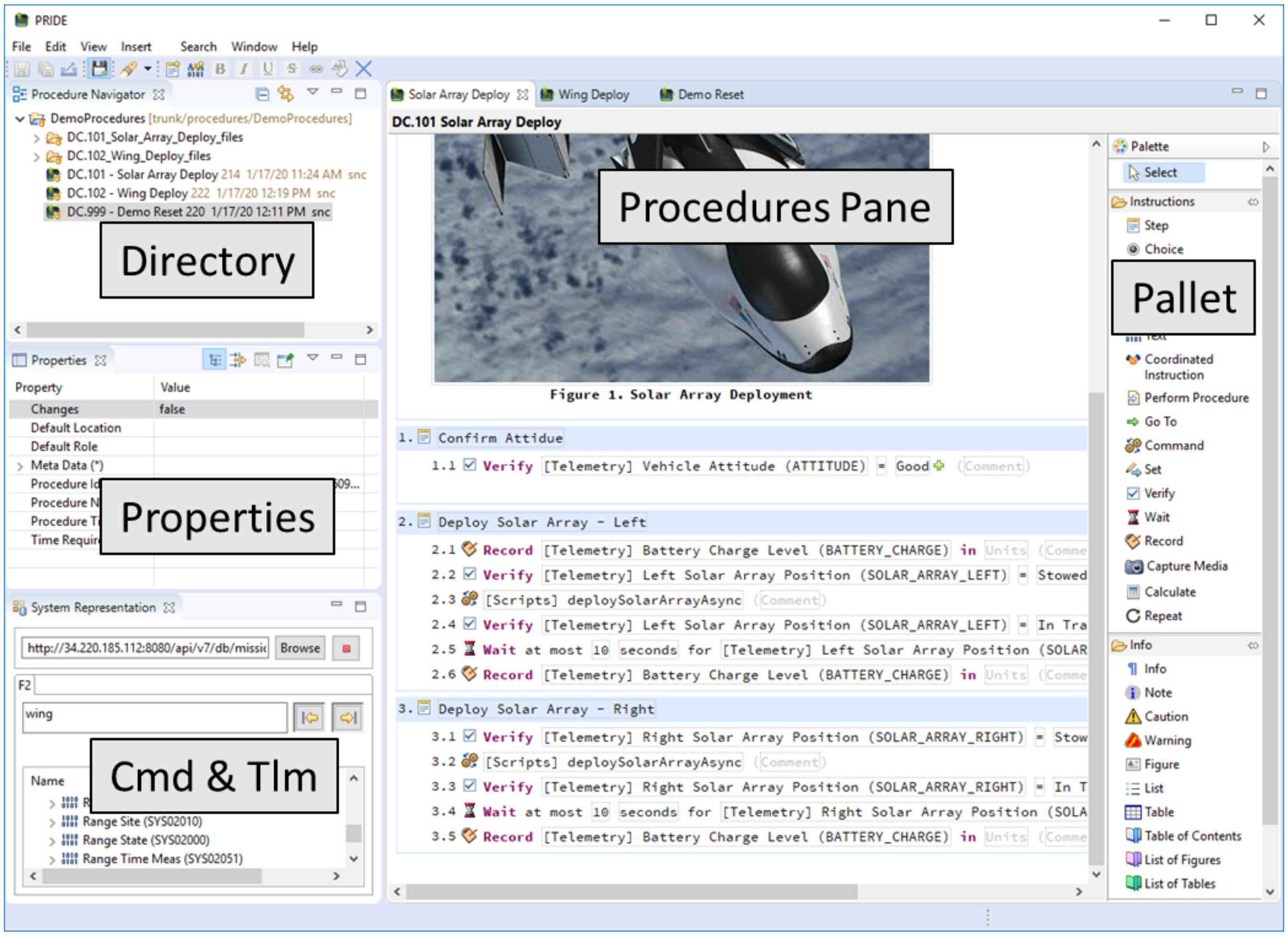
Activities in the timeline are tied to a PRIDE procedure. A PRIDE procedure can contain the following types of information:

- Meta-data that describe aspects of the procedure such as title, author, revision, etc.
- Steps that group activities into sub tasks
- Text instructions that describe actions (these are not able to be automatically executed)
- Command instructions that are passed to the system
- Verify instructions that test sensor data coming from the system
- Wait instructions that pause for either a period of time or until a particular expression evaluates to true
- Record instructions that take input from operators or sensors
- Calculate instructions that perform mathematical computations on recorded data
- GoTo instructions that branch actions to a specific step
- Conditional blocks that branch actions after evaluation of a true or false expression

- Choice blocks that branch actions after evaluation of an expression with multiple answers
- Call procedure instruction that activates a second (or child) procedure
- Informational statements such as figures, lists, tables, warnings, cautions, and notes

By combining the above elements, procedures can be written to perform a variety of tasks that can include both manual (human) and automated steps. These elements are defined in an XML schema called the Procedure Representation Language (PRL) [Kortenkamp *et al* 2007; Kortenkamp *et al* 2008].

TRACLabs has developed a desktop authoring system for creating procedures using an easy drag-and-drop interface (see Figure 4). The authoring tool connects to a procedure repository that stores all versions of a procedure and is displayed as a procedure *directory*. Various procedure elements are available in a *palette* for use in a procedure. The procedure author can drag-and-drop these elements onto the *procedure pane*. PRIDE Author also connects to a *system representation* so that the procedure can reference system telemetry and commands. Details of each procedure element can be set in the *properties* pane. PRIDE Author hides the complexity of the PRL XML representation from the procedure creator [Izygon *et al* 2008].



```
                 Figure 1. Solar Array Deployment

1. Confirm Attidue
   1.1 Verify [Telemetry] Vehicle Attitude (ATTITUDE) = Good (Comment)

2. Deploy Solar Array - Left
   2.1 Record [Telemetry] Battery Charge Level (BATTERY_CHARGE) in Units (Comme
   2.2 Verify [Telemetry] Left Solar Array Position (SOLAR_ARRAY_LEFT) = Stowed
   2.3 [Scripts] deploySolarArrayAsync (Comment)
   2.4 Verify [Telemetry] Left Solar Array Position (SOLAR_ARRAY_LEFT) = In Tra
   2.5 Wait at most 10 seconds for [Telemetry] Left Solar Array Position (SOLAR
   2.6 Record [Telemetry] Battery Charge Level (BATTERY_CHARGE) in Units (Comme

3. Deploy Solar Array - Right
   3.1 Verify [Telemetry] Right Solar Array Position (SOLAR_ARRAY_RIGHT) = Stow
   3.2 [Scripts] deploySolarArrayAsync (Comment)
   3.3 Verify [Telemetry] Right Solar Array Position (SOLAR_ARRAY_RIGHT) = In T
   3.4 Wait at most 10 seconds for [Telemetry] Right Solar Array Position (SOLA
   3.5 Record [Telemetry] Battery Charge Level (BATTERY_CHARGE) in Units (Comme
```

Figure 4: PRIDE Author

Real-time Execution

During real-time execution, flight controllers start with the schedule using OnTime to determine when to execute procedures manually or verify the completion of autonomously initiated procedures. Similar to planning, multiple operators can access the executing timeline, start procedures, and status activities. Operators can start manual activities by selecting the procedure link from the activity on the timeline. For autonomous activities, the procedure opens and starts execution as soon as the current time bar crosses the start of a scheduled activity as shown in Figure 5 below.

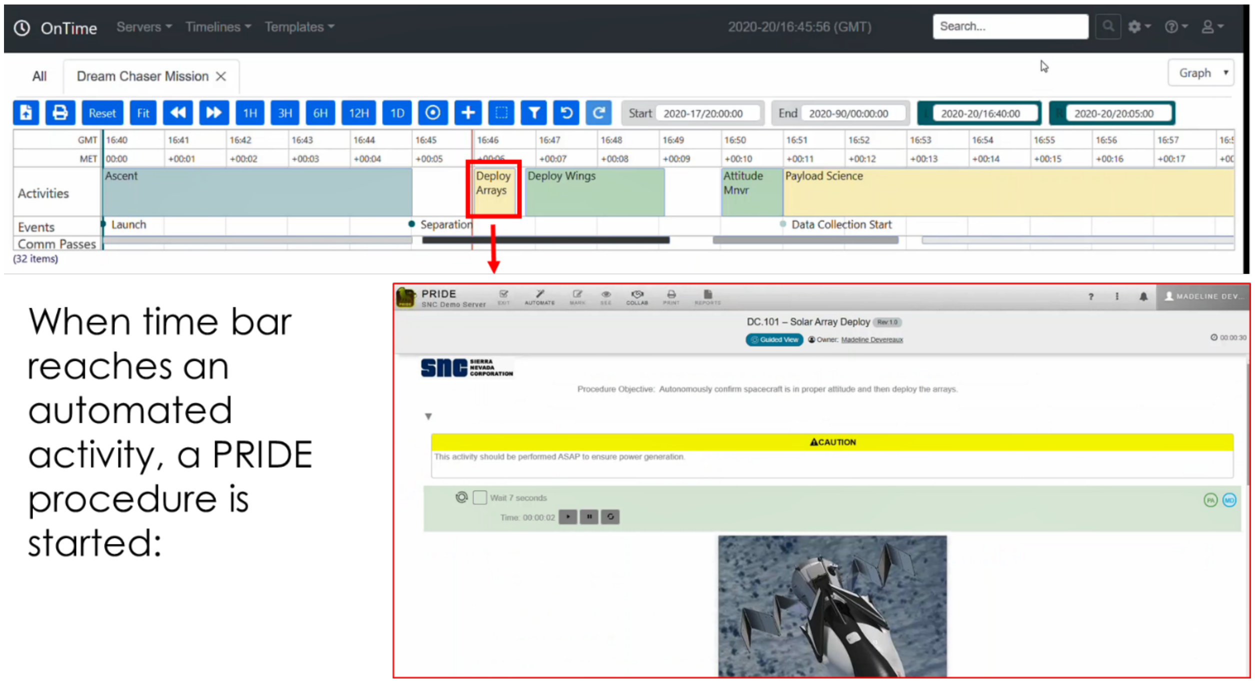
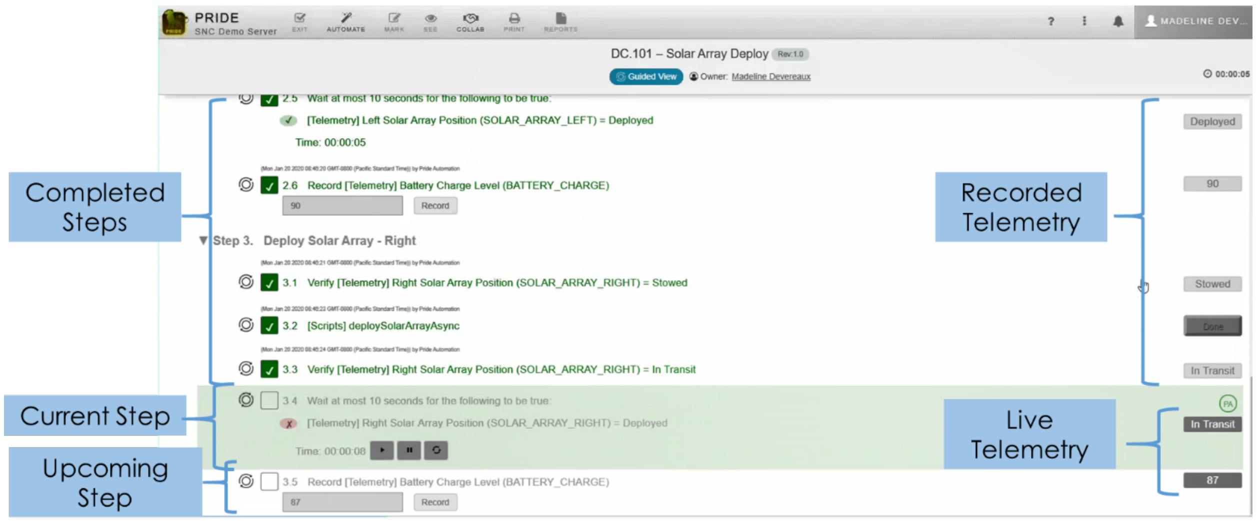


Figure 5: Autonomous Activity Execution from OnTime

The procedure opens in a browser-based interface that is shown in Figure 6. The browser connects to a PRIDE View server to retrieve an HTML version of the procedure and any actions taken by the operator are collected by the server and stored in a SQL database. The PRIDE View interface allows an operator to both complete procedure instructions manually or to supervise and control automated procedure execution. The interface marks the current step and indicates those steps that have already been completed and those that still remain. Live telemetry from the spacecraft is displayed in-line with procedure instructions. The PRIDE View interface also provides search mechanisms for operators to find appropriate procedures and start them independent of the timeline, if needed.



DC.101 – Solar Array Deploy Rev:1.0

Guided View Owner: Madeline Devereaux 00:00:05

2.5 Wait at most 10 seconds for the following to be true:
[Telemetry] Left Solar Array Position (SOLAR_ARRAY_LEFT) = Deployed
Time: 00:00:05 — Deployed

(Mon Jan 20 2020 08:46:20 GMT-0800 (Pacific Standard Time)) by Pride Automation
2.6 Record [Telemetry] Battery Charge Level (BATTERY_CHARGE) — 90 Record — 90

Step 3. Deploy Solar Array - Right

(Mon Jan 20 2020 08:46:21 GMT-0800 (Pacific Standard Time)) by Pride Automation
3.1 Verify [Telemetry] Right Solar Array Position (SOLAR_ARRAY_RIGHT) = Stowed — Stowed

(Mon Jan 20 2020 08:46:23 GMT-0800 (Pacific Standard Time)) by Pride Automation
3.2 [Scripts] deploySolarArrayAsync — Done

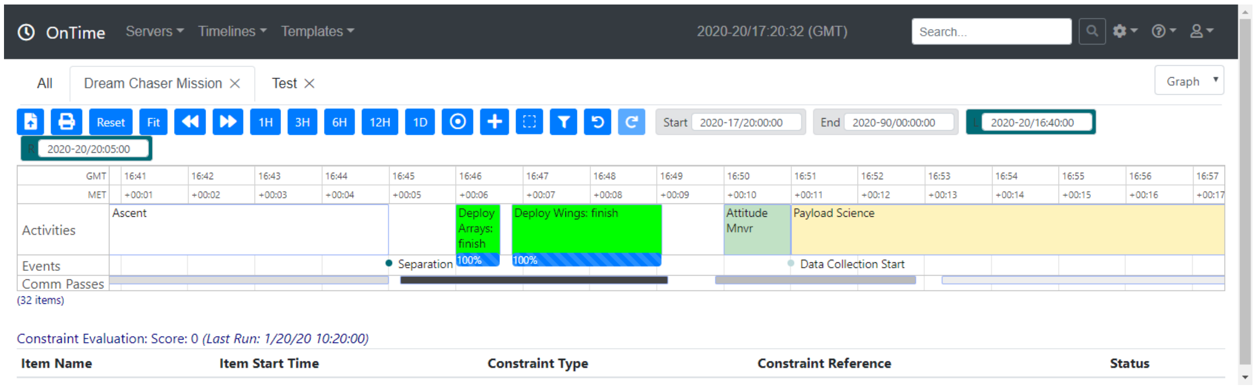
(Mon Jan 20 2020 08:46:24 GMT-0800 (Pacific Standard Time)) by Pride Automation
3.3 Verify [Telemetry] Right Solar Array Position (SOLAR_ARRAY_RIGHT) = In Transit — In Transit

3.4 Wait at most 10 seconds for the following to be true:
[Telemetry] Right Solar Array Position (SOLAR_ARRAY_RIGHT) = Deployed — In Transit
Time: 00:00:08

3.5 Record [Telemetry] Battery Charge Level (BATTERY_CHARGE) — 87 Record — 87

Figure 6: PRIDE View

While a procedure is being executed, OnTime receives the procedure status and indicates the status on the timeline by changing the activity color and showing a progress bar. Procedure status can either be in progress, completed, or aborted. Examples of completed activities are shown as green in Figure 7.



GMT	16:41	16:42	16:43	16:44	16:45	16:46	16:47	16:48	16:49	16:50	16:51	16:52	16:53	16:54	16:55	16:56	16:57
MET	+00:01	+00:02	+00:03	+00:04	+00:05	+00:06	+00:07	+00:08	+00:09	+00:10	+00:11	+00:12	+00:13	+00:14	+00:15	+00:16	+00:17

(32 items)

Constraint Evaluation: Score: 0 (*Last Run: 1/20/20 10:20:00*)

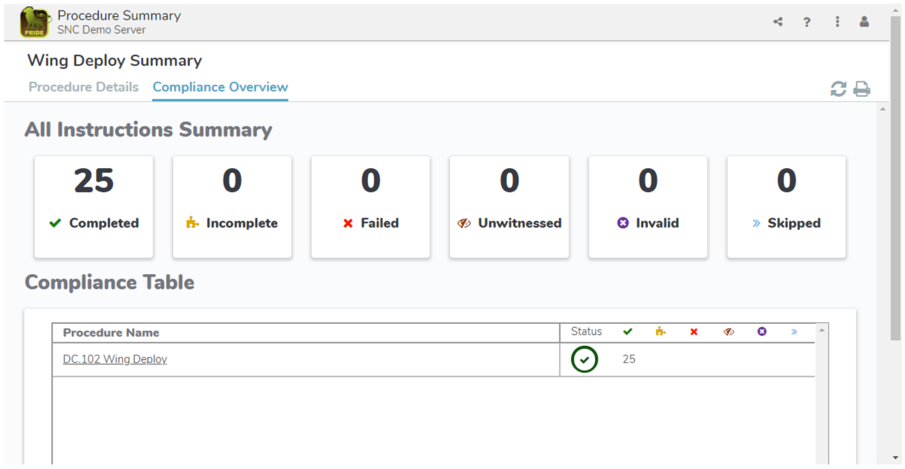
Item Name	Item Start Time	Constraint Type	Constraint Reference	Status

Figure 7: Procedure Status in OnTime

TRACLabs has also developed an executive called the Procedure Agent for eXecution (PAX) that can read the procedure XML file and execute it by issuing commands to a system and reasoning about telemetry from a system [Bell and Kortenkamp, 2011; Schreckenghost *et al* 2015; Schreckenghost *et al* 2018]. PAX operates on the same XML file that is displayed to an operator. Any actions taken by PAX are displayed to any operator viewing that procedure. PAX pauses on any parts of the procedure that require operator interaction. PAX can run without operator supervision if desired, although any instructions requiring operator interaction are pending indefinitely. Procedure authors and procedure operators can designate any instruction as requiring operator execution and this is respected by PAX as well.

Post Activity

PRIDE stores all procedure interactions, whether by operators or automation, in a SQL database for auditing and analytics. Thus, after the PRIDE system has been running, significant amounts of data become available regarding which procedures were run, execution duration, whether they were successful or not, and what vehicle did what procedures when. This information is available to data analytics tools that could be used to improve the efficiencies of the entire system or to influence mission plans. PRIDE has an API to retrieve this information. Some simple algorithms have been implemented to calculate, for example, the expected duration of a procedure based on information in the as-run database. The Figure 8 below shows a summary of a completed procedure.


Procedure Summary
SNC Demo Server

Wing Deploy Summary

Procedure Details Compliance Overview

All Instructions Summary

25	0	0	0	0	0
✓ Completed	Incomplete	✗ Failed	Unwitnessed	Invalid	» Skipped

Compliance Table

Procedure Name	Status	✓		✗			»
DC.102 Wing Deploy	✓	25					

Figure 8: Procedure Analytics

Results

Tools that build on integration flexibility from the start are more able to adapt to new innovations and mission needs of the ever-changing space industry. By starting from scratch and implementing lessons learned from previous programs, an integrated suite of commercially available tools has been developed, enabling multiple users to seamlessly flow through the full circle of operations planning and execution.

Starting from a timeline that guides the overall mission, users select an activity and view details, including a link to the procedure(s) to be executed. This procedure contains embedded key telemetry checks and links to commands as well as manual instructions. The embedded commands are then sent from the C2 software and the procedure then progresses to the next step and waits for telemetry confirming success of the command. The status of the procedure is fed back to the timeline so all team members are cognizant of the current state of operations. Additionally, multiple people can work simultaneously in the same procedure and timeline instance and see realtime updates from other operators, promoting better situational awareness throughout the operations team with less verbal communication required. The as-run data from the procedure and timeline is saved and informs future planning, creating a closed-loop planning and execution process.

References

[Bell and Kortenkamp, 2011] Scott Bell and David Kortenkamp. Embedding procedure assistance into mission control tools. In *Proceedings of the IJCAI Workshop on AI in Space*, 2011.

[Izygon *et al* 2008] Michel Izygon, David Kortenkamp, and Arthur Molin. A procedure integrated development environment for future spacecraft and habitats. In *Proceedings of the Space Technology and Applications International Forum (STAIF 2008), available as American Institute of Physics Conference Proceedings Volume 969*, 2008.

[Kortenkamp *et al* 2007] David Kortenkamp, R. Peter Bonasso, and Debra Schreckenghost. Developing and executing goal-based, adjustably autonomous procedures. In *Proceedings AIAA InfoSysAerospace Conference*, 2007.

[Kortenkamp *et al* 2008] David Kortenkamp, R. Peter Bonasso, and Debra Schreckenghost. A procedure representation language for human spaceflight operations. In *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-08)*, 2008.

[Schreckenghost *et al* 2018] Debra Schreckenghost, Scott Bell, David Kortenkamp, and James Kramer. Procedure automation: Sharing work with users. In *AAAI Spring Symposium on Designing the User Experience of Artificial Intelligence*, 2018.

[Schreckenghost *et al* 2015] Debra Schreckenghost, D. Billman, and T. Milam. Effectiveness of strategies for partial automation of electronic procedures during NASA HERA analog missions. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI) Workshop on AI and Space*, 2015.