17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.

SpaceOps-2023, ID # 415

Challenges in Operations Preparations when transitioning to EGS-CC based systems

Wolfgang Heinen[a], Steve Pearson[a], Pablo Beltrami[a], Alastair Pidgeon[a]

[a] *RHEA Group, Robert-Bosch-Str 7, 64295 Darmstadt, Germany, w.heinen@rheagroup.com, s.pearson@rheagroup.com, p.beltrami@rheagroup.com, a.pidgeon@rheagroup.com*

Abstract

At some point in time, software must evolve as the technologies it is based on become obsolete. This is especially relevant in mission operations, where development times are long and there is a need for operational continuity. The problem when migrating to a new control system from an operations point of view is not only the necessary retesting and a steep learning curve for the team but also an intrinsic loss of knowledge that can come with such a change.

The European Ground Segment Common Core (EGS-CC) is an initiative of ESA, European prime industry and national space agencies with the objective of developing and promoting a new generation of ground systems. Its architecture is component-based and service-oriented and has been designed to be used in pre- and post-launch applications. It consists of a core and a reference implementation. Components in the Reference Implementation can be updated, and new components added to extend the functionality. One such concrete system is the EGSE Reference Facility (ERF), currently maintained by ESTEC and made available to users in the ESA Member States.

The EGS-CC is replacing the legacy SCOS-2000 Mission Control System. This paper will show how the transition to EGS-CC is eased with an upgrade path from the legacy SCOS-2000 dedicated flight control procedures to an open procedure format that can execute autonomously or interactively, directly communicating with the EGS-CC.

1 Introduction

The intent of EGS-CC is to provide the core of monitoring and control elements. It is then up to the operator or manufacturer to integrate other tools that support their concept of operations or testing approach. For ESA mission operations, this is EGOS-CC, which tailors EGS-CC integrating existing tools in the ESOC environment. RHEA is addressing commercial and other missions with its ASTRAL offering, to provide operators and manufacturers with an integrated set of tools to support mission operations or testing. Operators and manufacturers may have legacy procedures for operations or testing that they wish to reuse with an EGS-CC system as well as spacecraft databases based on the SCOS-MIB or other formats. RHEA has therefore set-out to provide a set-of tools, adopting new technologies, that can support this transition and the ongoing operations and testing activities.

In the following sections, we briefly present ASTRAL and its main components and then (as the focus of this paper) the key components relevant to the mission preparation environment. These are within ASTRAL Prepare (as a standalone tool this is RHEA's well-known MOIS) the new procedure format for EGS-CC support and the RHEA developed Mission Model Editor (which is now part of EGS-CC) providing support for database conversion and editing in a under-friendly way, hiding the complexity of the EGS-CC TDM (Technical Data Model).

2 ASTRAL Overview

ASTRAL is RHEA's component-based Ground Segment Offering responding to the changes and needs of the commercial & institutional markets. It provides an integrated environment for the different systems and components of the ground segment. The figure below shows the main components of the ASTRAL offering:

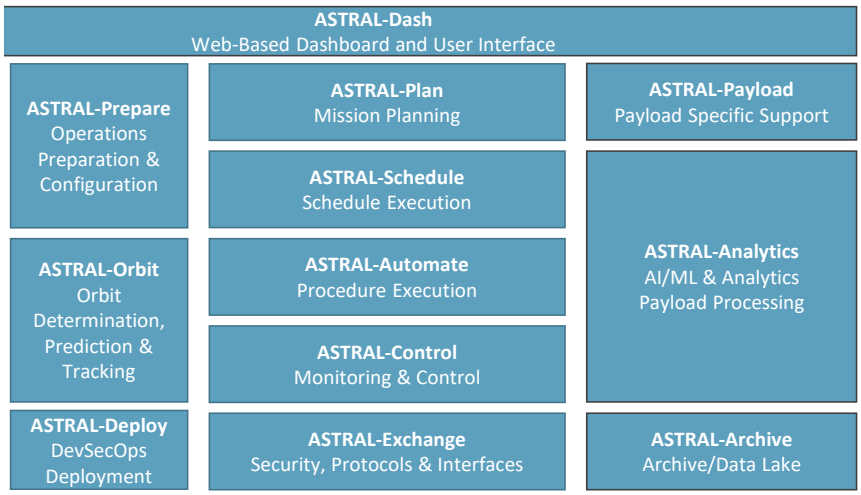


Figure 1 ASTRAL Component Concept

ASTRAL consists of components, each supporting ground segment elements. Each can be replaced by customer provided or third-party components.

Aligned with the ongoing trend in Europe, where European Ground Segment Common Core (EGS-CC) is going to be the default mission control system for institutional and agency-related space missions, ASTRAL is already compliant with EGS-CC needs. ASTRAL's components support the functionality to monitor and control a spacecraft but we recognise our customers might have their own heritage, preferences or specific needs for some of the components. A key advantage of the system is that almost all ASTRAL components can be swapped-out to meet these needs. This has been demonstrated with mission planning systems, flight dynamics systems, operations simulators and even the control system itself where we support EGS-CC and SCOS-2000. The following provide brief descriptions of the main ASTRAL monitoring and control components:

ASTRAL-Dash provides a high-level browser-based view of the overall spacecraft(s) and ground segment status. This is developed in conjunction with the customer to fit their operations concept.

ASTRAL-Prepare as the latest evolution of RHEA's flagship operations preparation product, MOIS, that has been used to support over 100 spacecraft ranging from a "flying laptop" through to a satellite constellation of ~30 spacecraft. Validated operations procedures, whether manual, semi-automated or automated are essential to ensure safe, reliable operations of your space segment and corresponding ground segment elements.

ASTRAL-Prepare supports authoring of procedures, consistency checking of telemetry (TM) and telecommands (TC) in the procedures with different spacecraft database versions, validating the procedures and finally publishing them to a flight operations manual or to operations languages used for automation. It also supports spacecraft database management functions. Procedures can be authored in tabular, flow-chart or script editors. The flow-chart view is synchronised with the tabular and scripting views, so changes made in one are reflected in the other. Procedures are not limited to spacecraft TM/TC but can also be used to automate ASTRAL components, ground station interactions and other ground segment systems.

ASTRAL-Prepare supports powerful configuration management capabilities to manage different versions of procedures and databases for multiple spacecraft and multiple users. It can be used standalone on a PC or deployed on a server for multi-user/multi-mission support (this is the typical deployment). Role-based access controls govern

user access to missions and the actions they can perform. A validation workflow allows procedures to be approved once they have been validated prior to release to the users.

ASTRAL-Plan supports constraint-based planning for platform and payload operations. ASTRAL-Plan supports planning multiple spacecraft operations and multiple domains (e.g. ground segment, spacecraft). Each spacecraft can have different payloads and different resource constraints. Because ASTRAL-Plan is integrated with the other ASTRAL-Prepare component, it is aware of available spacecraft operations procedures and spacecraft commands. The latter are used when preparing an onboard timeline to upload to the spacecraft.

Where there is a separate dedicated payload planner for a spacecraft, the overall operations constraints can be provided to that planner from ASTRAL Plan and the resultant payload operations requests can be imported into the ASTRAL Plan.

ASTRAL-Schedule supports the execution of the schedule generated from ASTRAL-Plan or another planning tool. For most automated missions, this is typically the primary view used by operators to monitor progress, only using the mission control system to look at parameter values or send commands manually. Each activity in the schedule corresponds to a procedure that will be executed at a pre-planned time, when an event occurs (e.g. eclipse entry, acquisition of signal by a ground station) or when another procedure completes execution (depending on the constraints in the plan). The user sees a view of the executing timeline and the status of the procedures being executed. If a procedure fails to complete execution, it will turn red and the user can click on the procedure to open it and can see where the execution stopped.

ASTRAL-Automate supports the execution and debugging of procedures developed in ASTRAL-Prepare. Automated procedures are normally executed without user interaction but during validation or troubleshooting activities it is highly desirable to be able to step through a procedure to understand what is happening and debug accordingly. ASTRAL-Automate interfaces to the underlying mission control software (ASTRAL-Control) which in turn can be connected to the operational simulator, a spacecraft undertest (or flat-sat), a test simulator (ASTRAL-Simulate) for verification & validation and troubleshooting, or the spacecraft in orbit for operations. Once validated, procedures are typically initiated from a schedule executing in ASTRAL-Schedule, but individual or multiple procedures can be executed from the ASTRAL-Automate user interface.

As well as sending TC and monitoring spacecraft TM, procedures can also be used to automate ground segment activities, such as ground station passes. For LEO spacecraft, this is the primary use of automation, especially for passes occurring over evenings or weekends.

ASTRAL-Control – ASTRAL is designed to be agnostic to the mission control system kernel. RHEA has successfully deployed ASTRAL with ESA's Community Source SCOS-2000 and ASTRAL also supports the EGS-CC. RHEA has expertise in configuring, adapting and deploying both these mission control systems.

ASTRAL-Orbit is a component that can use a third-party product that supports an automation interface. The choice of tool depends on the customer's specific needs and budget. RHEA has integrated with FlexPlan from GMV (commercial licence) and GMAT from NASA (open source), which provides a low-cost capability suitable for many missions. Other COTS tools that have an automation interface - such as FreeFlyer from a.i. solutions - can be readily integrated into ASTRAL

The components of interest for this paper are ***ASTRAL-Prepare*** introducing a new generation of editors for procedures and the mission database; ***ASTRAL-Automate*** providing an automation solution on the data items maintained by those new editors; ***ASTRAL-Schedule*** that gives a lights-off automation scheduling solution with the mentioned data items.

3 Technical Overview for the new generation editors in ASTRAL-Prepare

3.1 The ASTRAL-Prepare Framework

The ASTRAL-Prepare component is a framework which manages all configuration data items in a structured way and provides an advanced MMI which allows specific editors for each type of data to be integrated.:

-

The global model of configuration data starts with a classification of elements that share common properties and which can be discerned from other types of data via their properties, these are the **Data Types**. A **Data Item** is one specific instance of a Data Type. The Data Model is a plug-in framework that defines a set of interfaces that a Data Type must conform to:

- Framework. The Framework provides generic configuration control features and an API at the Data Type level. All files and metadata that constitute a particular Data Type will then be managed as a group – i.e. as a single Configuration Item. The framework takes care of Data Item creation and editing and delegates the various methods over a set of Data Items to their Data Type.
- Common Properties. These are common for all Data Types and are typically Name, Description and type. They can be used for a common search & browse throughout the entire MDM and across all Data Types
- Specific Properties. These are Data Type specific. They can be used for further processing (for example inside a properties window of a procedure editor). Another usage can be a Data Type specific search & browse.
- Methods. The following predefined interfaces must be implemented for each Data Type: consistency checks, query, generate products and publish. These are particularly important when actions on the entire MDM dataset are performed by the Framework, such as checking overall consistency or publishing a FOP release.

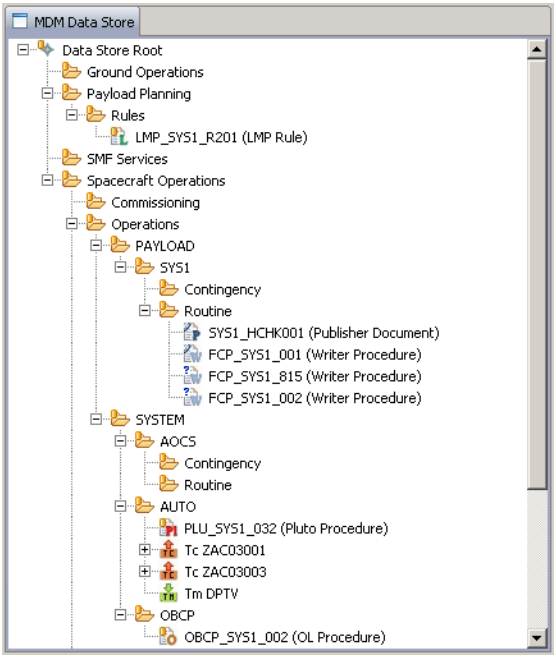


Figure 2 Classification of elements within the data model

A number of data types and editors have been already implemented: editors for OBCPs (On Board Control Language Scripts), Mission Planning Rules, Pluto Scripts, Writer Procedures & Word Documents and Synthetic-S2K Scripts. The Mission Model Editor (MME) and the PME procedure editor that are subject of this paper also implement such a Data Type acting as ASTRAL-Prepare plugins:

3.2 The Mission Model Editor (MME)

The Mission Model Editor (MME) is an editor for EGS-CC Technical Data Model (TDM) based data. It is an EGS-CC specific tool developed by RHEA to make editing of the TDM much simpler for the users. The MME adds a presentation layer on top of the TDM defining and organising the items that one would expect to manipulate, such as packets, parameters and procedures, and locates them in a Mission Model tree. This helps the MME portray the TDM data in a simple, useful and intuitive way. This layer stores additional information such as the location of packets and displays in the Mission Model tree.

The MME hides the management of the TDM Monitoring and Control (M&C) Elements, their definitions and contents, the mappers, the mapped implementations and the file-backed Configuration Items (CIs) that contain them. It imposes a policy of one M&C Element per CI (file) to facilitate the modular management of stored sub-tree data. If the MME is used exclusively, then any M&C Element sub tree will consist of an independent (and thus replaceable) set of CI files. This means that other people/organisations can work on different branches of the tree independently and they can be easily integrated. See Ref [1].

The TM Parameter Editor edits TM parameters in the Mission Model. These correspond directly to TDM Engineering Parameters. From the Mission Model Navigator a new TM Parameter is created in the selected element as shown here:

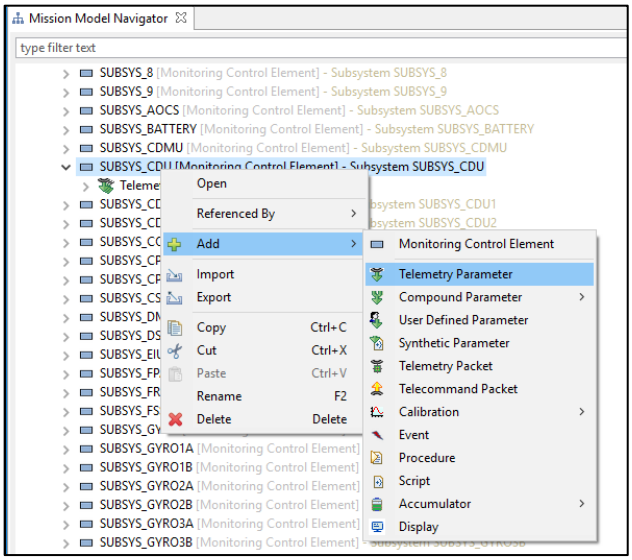


Figure 3. TM Parameter Creation

This opens the TM Parameter editor where mandatory and optional fields are shown:

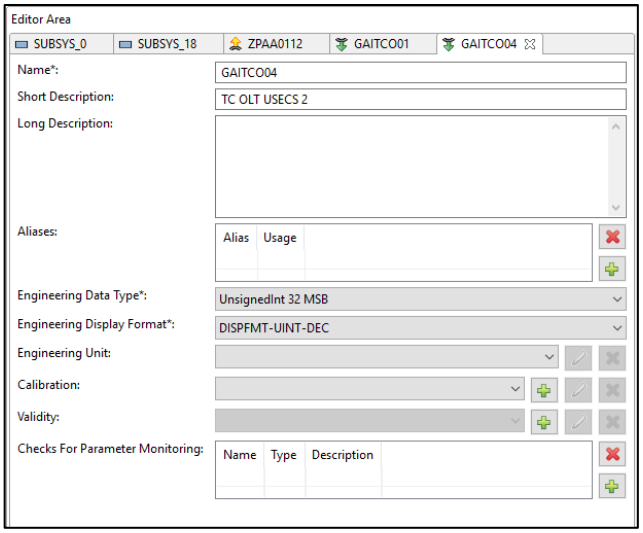


Figure 4. TM Parameter Editor

The MME extends the OPEN framework developed by ESA and thereby benefits from the SCOS-2000 MIB importer functionality.

3.3 The Procedure Model Editor (PME) – the new ASTRAL-Prepare Procedure Editor

The rationale for the PME procedure editor is to provide an environment to the user for the design of procedures. Conceptually, procedures are a formalised mechanism to achieve goals such as certain use cases in satellite operations and test. Procedures can be classified into product generating and executable procedures:

Product Generating Procedures – procedures can be simple high-level documents serving as operator instructions or they can be detailed enough to generate artifacts/products for a target system. In both cases, they are for authors and users who do not need to understand the final product format. High-level procedures can be considered as design documents to be completed for product generation after review. These design and documentation aspects are key to the mission preparation and testing cycles. The products generated from such procedures are:

- EGS-CC activity lists and SCOS-MIB sequences: These are linear mechanisms for the sequential execution of commands. They do not contain any logic (apart from relations between the commands) or other information that would guide an operator in their tasks. It is therefore imperative to start with a

procedure where additional information can be added. This includes comments for the operator, telemetry checks, waiting for events and telemetry packets.

- Executable artefacts: A number of Operations Languages exist, such as On-Board Control Procedures expressed in OL, Automation procedures expressed in Tcl/TOPE, SPELL and other languages. Some of these Operations Languages are relatively readable while others look more like impenetrable programming code. In all cases they are exported for execution in a remote system. MOIS has supported the generation of these products for many years via its Writer Editor. Within ASTRAL Prepare the design of the PME procedure model (XPM) provides a solution that is fully compatible with these exports, and it is extendable to support the generation of new product types.

Executable Procedures – ASTRAL Automate supports direct execution of script-based PME procedures rather than submitting procedures to a control system. Direct execution makes the solution more flexible and responsive and means that the requirement validation loop can be closed within the ASTRAL Automate itself. See Ref [1] for more details.

The objective is that the ASTRAL Automate is independent from, but compatible with, the TDM.

What is important is that the procedure can be read and reviewed by authorities that are not software experts. Therefore, they need to be easily readable, and they should be presented to the user or reviewer in different views: textual, flowchart and possibly tabular views to ease the review process for everybody. The approach is therefore to not start with a programming language but to start designing the procedure format in a model that allows the various representations to be views of the same model, which we call the Extensible Procedure Model (XPM):

*3.3.1 The Extensible Procedure Model Ref[1]**

The Extensible Procedure Model (XPM) consists of 2 parts: A minimal model with Steps, Test Requirements and branching; and an executable extension to this minimal model. The minimal XPM model is an 'interface' that procedures must implement to fulfil the structural requirements of a structured procedure. It has **verifiable steps**, **executable statements** and basic 'procedural' **branching**.

Procedures that implement the minimal XPM model can then be represented in a flowchart and used to verify test requirements. The minimal XPM has a Procedure base Item which is a container for the procedure, stored in individual files. The Procedure base Item consists of *Steps* which contain *Statements*. A step is a defined action with start & end point, and it has an optional precondition and a confirmation part. Statements contain the procedure logic, these are elements for *Branching* (*SelectCase*, *ForEach*, *For*, *While* or *If), Control (*Synchronise, set variable, prompt, log) and *Execution (ExcuteActivity/TC, CheckState, WaitForState, WaitForEvent, ExpectPacket, SetMontoring).*

The minimal XPM provides a minimum "common denominator" for an exchange language and it represents essential procedure data either for export to other formats such as Activity Lists (or sequences in SCOS), or for direct execution. The model is insufficient on its own for execution, which needs some kind of language, but it does capture the structure necessary for flowchart representation, because it implicitly defines an execution flow with its steps, statements and enclosed branching statements.

3.3.1.1 Domain Specific Language for PME Procedures

One way to populate the XPM is to define a Domain Specific Language (DSL) grammar that provides the user with a readable and editable script. Xtext provides the means to develop such a grammar based on an EMF model that then becomes its Abstract Syntax Tree (AST). The advantage of starting with a model and using it to define a grammar, rather than the other way round, is that the model is then clear and can be used for other types of editors, such as flowcharts or tabular editors, which provide alternate views of the same procedure.

* The XPM and the procedure DSL, and corresponding editors and server applications were originally called Procedure Management Environment (PME) which are mentioned in the references but are now part of ASTRAL Prepare and Automate.

The DSL syntax – called ***PDSL*** later in the paper – developed for the PME Procedure Editor, is one of many possible implementations with this model. Keywords, which map directly to model items, can be changed as well as other structures. Through the concept of an extensible procedure model, it can support a variety of customised Domain Specific Languages (DSL) to support customer's specific requirements and compatibility with legacy operations and test procedure scripts.

3.3.1.2 Executable Procedure DSL

Compared to most languages this is a straightforward syntax that should be immediately understandable to most Test and Operations users. The editor prompts for all keywords as when typing minimising the need to remember all language constructs and syntax. The procedure itself and all branching statements are Step based. Boolean and arithmetic expressions are delegated to a scripting language (currently Groovy but Python also possible) and a standard editor is used to update them in situ.

The principles of this PDSL are unsurprisingly the same as the model on which it is based: formal verifiable steps with pre- and post-conditions and basic procedural branching. Activities are called asynchronously and can be explicitly synchronised on execution completion. In other respects, it is single threaded. Expressions are delegated to a standard editor. Parameters and variables are identifiers that can be referenced in arguments, log messages and expressions. The editor prompts for available activities, arguments and parameters. Editing capabilities include code assist / syntax highlighting, hover text for descriptions and automatic formatting

PDSL Debug Tracing – ASTRAL Automate supports debug tracing of its PME procedures between the PDSL and the executing (e.g. Groovy) script using corresponding annotations in the generator. A user is therefore able to step through procedures directly in the PME procedure editor, setting break points and examining variables, without having to debug the underlying scripting language directly.

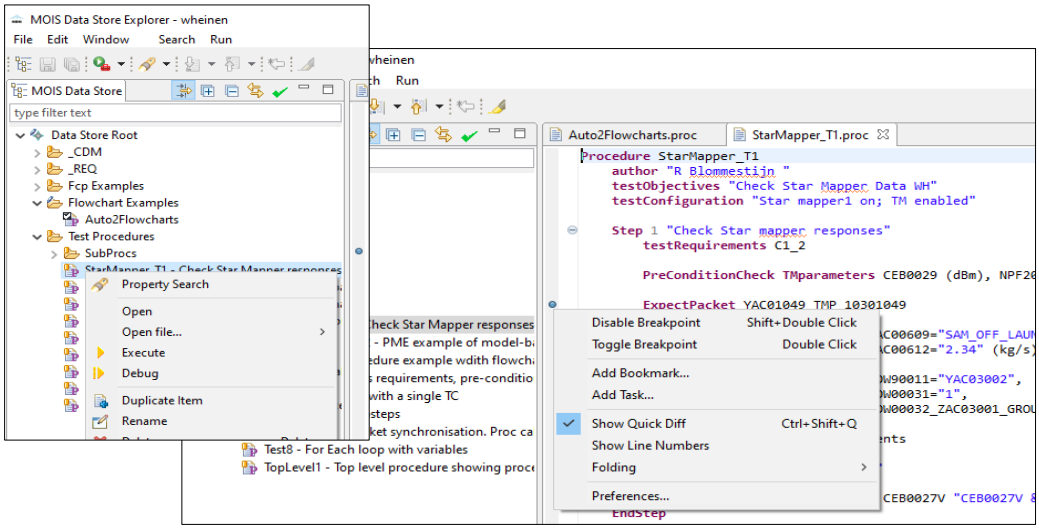


Figure 5 Execution Debugging

A procedure DSL that can delegate execution to a standard scripting language for boolean and arithmetic expressions provides a clear solution that is both constraining in structure and execution, yet flexible in variable management. Apart from a simple expression syntax for basic expressions there is very little to learn for a new user. Starting with the underlying model and not mixing syntax and content makes it clear how the script maps to the model and how it executes.

The use of a procedure DSL significantly reduces procedure testing effort since the scope of what is possible is hugely constrained. The mechanics of execution are hidden in the generated executable Groovy script and need not be a concern to the procedure developer.

This is considered a more suitable solution than adapting an off-the-shelf language, with clarity of functionality in a DSL set against familiarity with a known syntax (an off-the-shelf language may also create expectations of more possibilities than are practically available).

Different DSLs can be created for the XPM procedure model. The PME Editor within ASTRAL Prepare currently supports 2 DSLs, one for AIT test procedure execution and another for generating ESOC Flight Control Procedures. The same tools - such as a flowchart editor and data queries - can be used in both cases because the procedure model is common. Relational checks against the Mission Model and other semantic checks are common as well.

The procedure model is not only used to generate executable scripts (or Activity Lists/Sequences for FCPs). Other executable forms can be generated from it. An Xtext DSL has been chosen because it provides complete control over the language and the better opportunity for added functionality such as content assist, validation, formatting, quick fix and labelling in the DSL editor. A major advantage of starting with a data model and deriving the DSL is that other data representations are possible such as flowchart and tabular views, by simply accessing the data model – exposed clearly and simply by EMF.

3.3.1.3 Flowchart Design

The flowchart representation concentrates on high level logic and not implementation details. Based on the XPM procedure model it is implemented using Sirius. The flowcharter provides a means to view and edit Step & Branch object descriptions. All structural types available in the procedure model are represented.

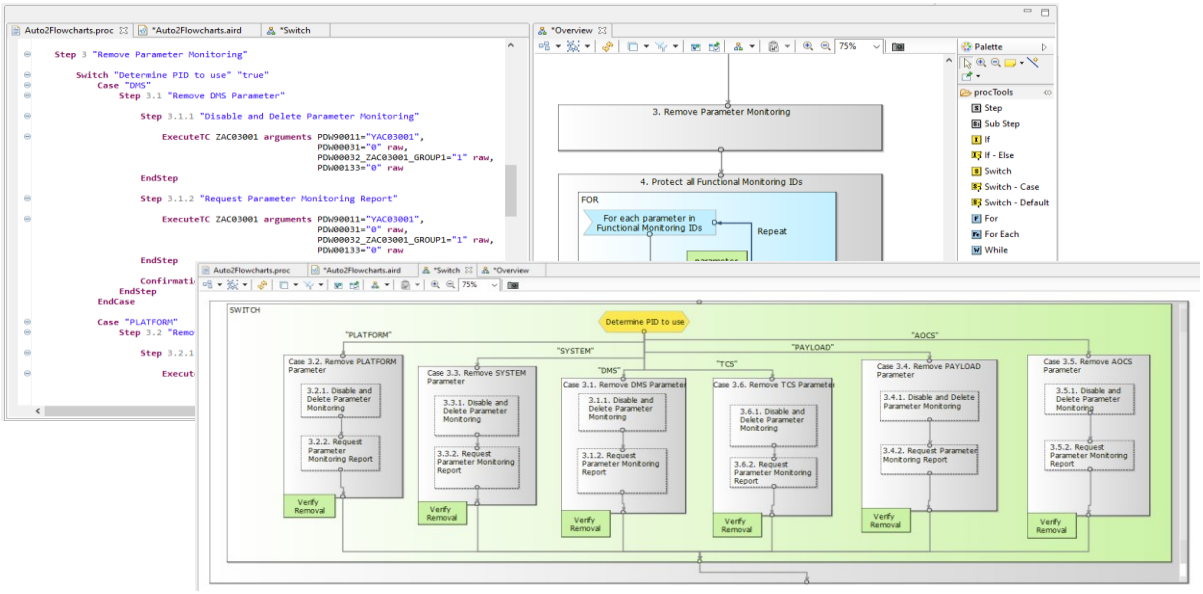


Figure 6 PME procedure in Text & Flowchart view

As shown in the picture above, editing can be done in parallel between the flowchart view and the DSL view and mutual saves are reflected in each of the editors. Several flowcharts are possible, and the level of granularity of each of them can be decided by the procedure author. In this example the first flowchart "Overview" only shows one Step-box in grey for step 3, the second flowchart shows the step3 in more details – a switch with 6 cases.

3.4 The ASTRAL Procedure Data and Automation Servers

3.4.1 Client server architecture for the data access – the ASTRAL Procedure Data Server

ASTRAL Prepare is designed to support both a stand-alone RCP editor and a web-based editor. Both have to access the TDM-based Mission Model data. The chosen solution is to have a client-server architecture with the Mission Model queried via the server. A client-server architecture is preferable for a procedure editing and execution environment and is in any case imperative for a web client.

Some abstraction is needed to keep the editor implementation simple, clear and independent from the TDM: the ASTRAL procedure data server is a service provider for the Mission Model data. An EGS-CC adapter is implemented that returns simple objects from queries to the TDM. This means that the ASTRAL procedure data server uses EMF, but the PME Procedure Editor does not. And it also means that a SCOS-2000 adapter is conceivable without affecting the service provision for the data. See the DA-Service API shown in Figure 8.

Note: In the context of SCOS-2000 automation with PME procedures, it is not mandatory to have a SCOS-2000 adapter in the ASTRAL Procedure Data Server. The MME supports the MIB-import, and the SCOS-2000 Automation Server does not require any MIB access.

3.4.2 Client server architecture for automation – the ASTRAL Procedure Automation Server

The same pattern is applied for ASTRAL Automate for PME procedure execution. Conceptually, the Automation Server component can be split into 2 sub-components: the ***Runner*** (CTL-Service API) and the ***Commander*** (EXE-Service API) The Commander deals with the execution of atomic actions like sendTC, checkTM while the Runner executes the (Groovy) PME procedures on the server-side without any user interface.

A client can therefore connect to the Automation Server for various purposes:

- To upload the latest version of the procedure-set from the ASTRAL Prepare for remote execution by the Automation Server (Runner).
- While editing a procedure, the user can run his local edited copy of the procedure in debug mode. This means that the editor runs the procedure locally and it connects as a client to the low-level Commander services (to send TCs etc…).
- Other clients can connect to the Runner for executing procedures remotely, to monitor them and to act on them at high level (get procedure list, start-, stop-, pause-, resume-, abort-procedure,…). The implemented clients are the Scheduler and a Supervisor. This Supervisor is a web-application that does the M&C of procedures including a log display:

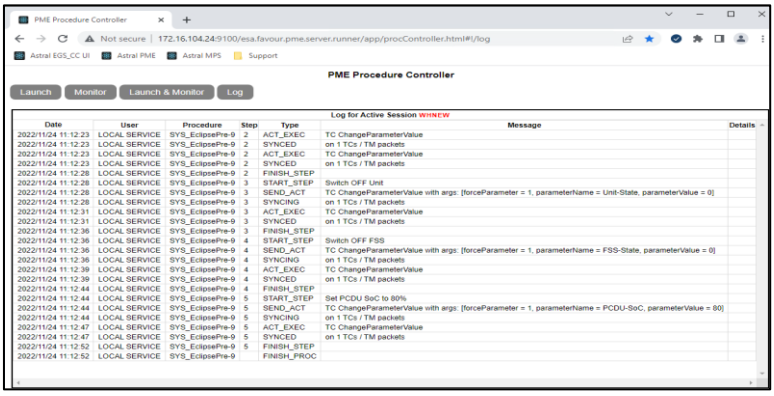


Figure 7 Web-based Supervisor

Inside the Commander, one adapter has been implemented for the EGS-CC. And very much like the Data Server, the Automation Server can implement a SCOS-2000 adapter without affecting the execution service provision. The following picture shows the ASTRAL Procedure Server components with their EGS-CC & S2K adapters as well as the 3 service groups:

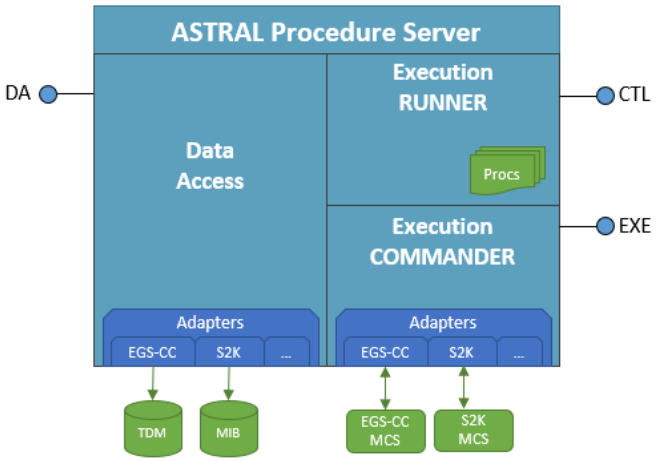


Figure 8 ASTRAL Procedure Server Components & Adapters

3.5 The ASTRAL-Schedule application

As explained beforehand, the Astral-Schedule is an application that presents a timeline/Gantt view from where the user can insert & monitor procedures in the schedule. The scheduler starts procedures at a pre-planned time and provides high-level monitoring & control of the procedures – but it is unaware of the content of each procedure.

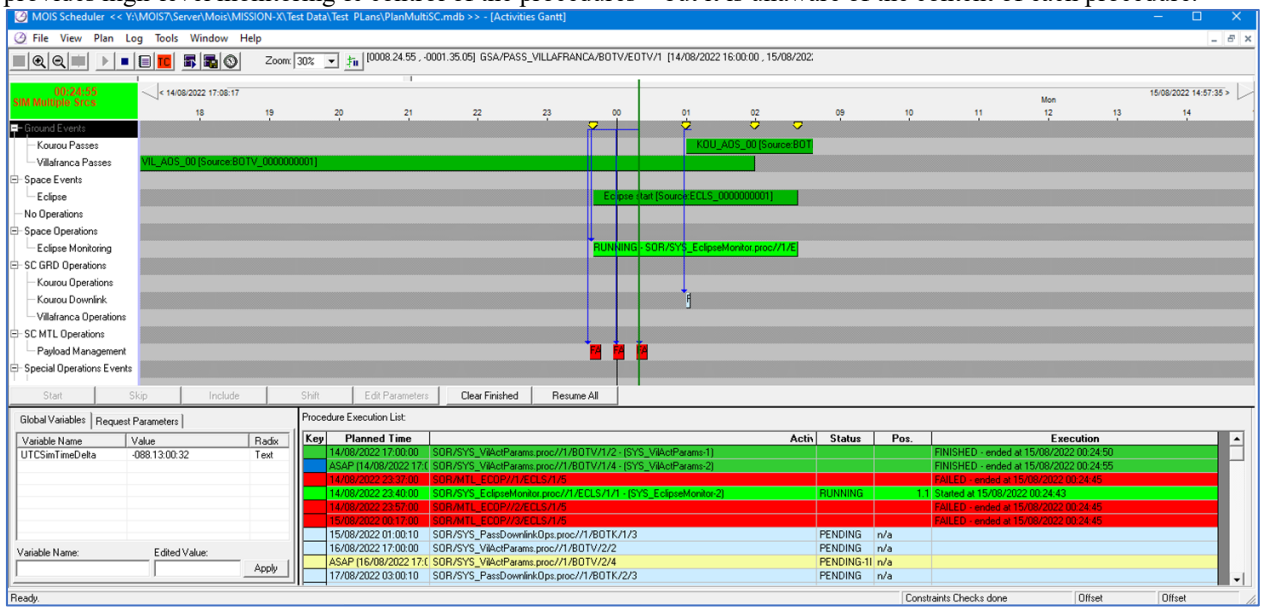


Figure 9 ASTRAL Schedule running PME Procedures

The Scheduler tool has been used at ESOC for many years to automate the execution of Writer procedures with SCOS-2000 via a client connection to the ***Writer Automation Server***. This ASTRAL Schedule client-connector has simply been augmented by another interface to the ASTRAL Procedure Automation Server (Runner) to monitor & control executing PME procedures.

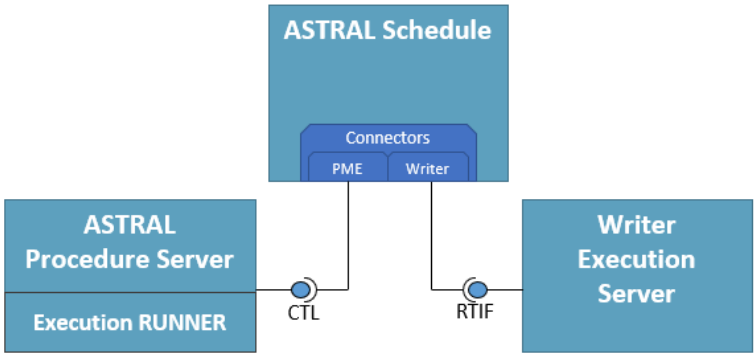


Figure 10 Scheduler & connectors

This means that the Scheduler can schedule & run Writer **and** PME procedures together in the **same** plan. Just like the ASTRAL-Prepare is an application that hosts Writer data items & editor and the PME procedures & editor. The full picture of the system is shown hereafter:

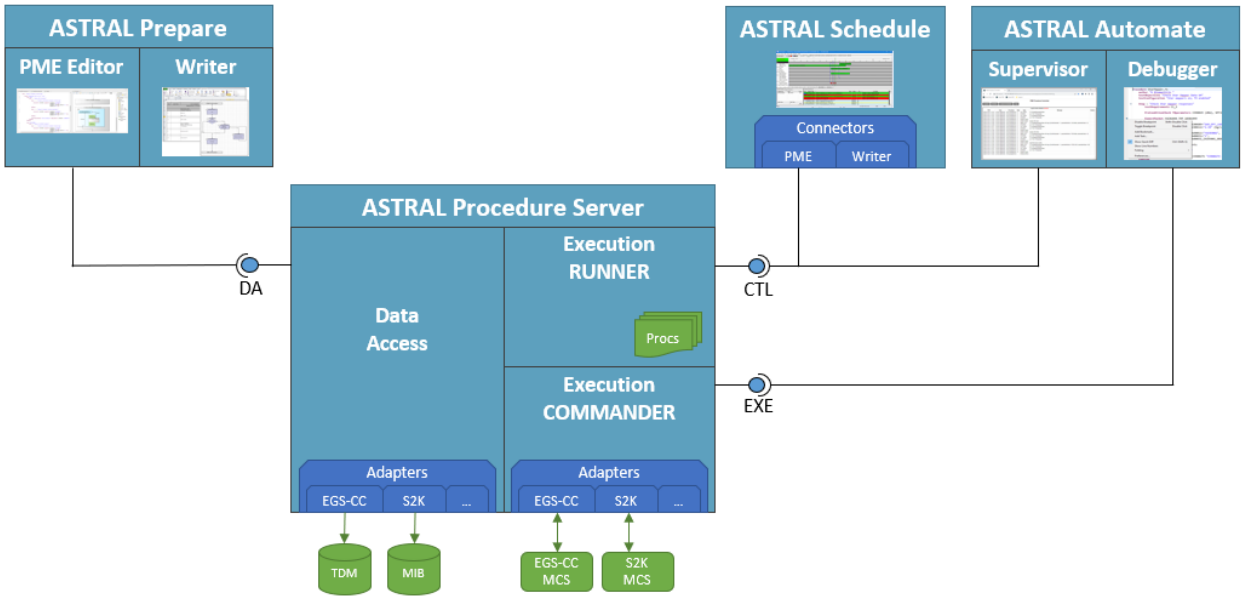


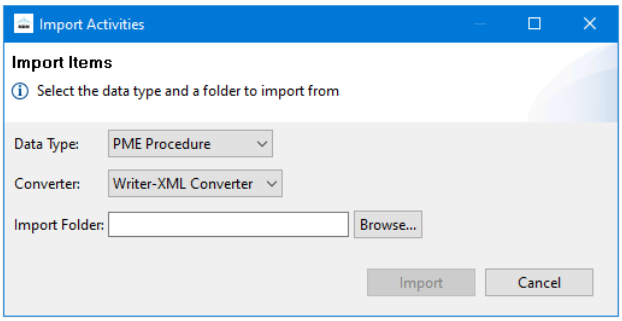
Figure 11 Full System with Procedure Server supporting integrated preparation and automation of the schedule and procedures.

This architecture opens the door for many different use cases that can be to phase-in PME procedures for SCOS, replace Writer with PME for SCOS or the Migration of a mission from SCOS to EGS-CC.

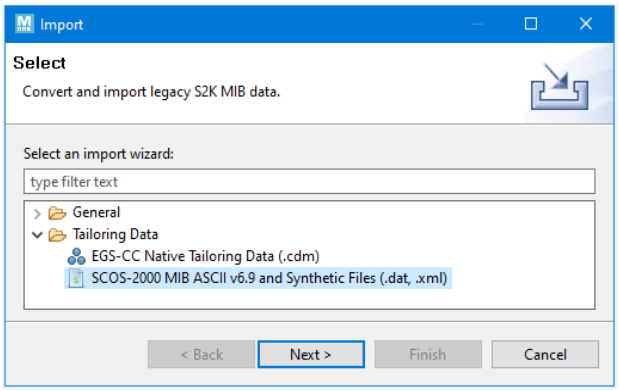
4 The Transitioning Scenarios

4.1 Data Conversion

Procedure Conversion – regardless of the intention to move from SCOS to EGS-CC or not, procedures can be converted from Writer format to PDSL (PME procedure format). The ASTRAL-Prepare application provides the feature when importing data into the mission dataset:



Database Conversion – When moving to EGS-CC, the SCOS MIB database is replaced by the EGS-CC TDM and the MIB data can be converted via the MIB-Importer which is part of the MME.



4.2 Phasing-in PME procedures for SCOS

Currently, the baseline SCOS-2000 procedure format is stored in a per-procedure database and the editors are in a tabular form (Writer) and Flowchart. The Writer Editor is forms-based and uses MS Excel for the visualization. A forms-based editor has some advantages (for example they can show many attributes & features of the edited item – like calibration curves with their value conversions), this is certainly a very comfortable way for designing mostly linear procedures. Where a DSL-based scripting approach like the PME scores points, is when procedures are of more complex nature including loops, branching and complex expressions. This is the case for a part of the automation procedures.

Some ESOC missions running SCOS-2000 have automated their routine operations with Writer procedures. They have further automated their pass activities via the MOIS Scheduler, where all routine pass operations are scheduled and the event monitor is used to react on issues coming from the control system (out-of-limits), starting Writer Contingency Recovery Procedures automatically on known events or alerting the operator on non-registered events. See also Ref [2].

Added Value for Automation – As the system is designed to support both, Writer and PME procedures at the level of individual procedure execution as well as from the scheduling, users can *phase-in* the new procedure format. They do not need to switch from one format to another. The team can decide to delegate some of the automation use cases to PME where a change brings added value. These cases can be tested without affecting operations and once validated, some PME procedures can replace their Writer counterpart. As explained beforehand, the change to PME is eased by a porting mechanism from Writer and the task of the team is then to validate the ported procedure against the Simulator.

Writer and PME procedures do not just co-exist in the overall dataset but they can be available for selection for the routine operations schedule. This means that the operations team can have a Scheduler plan that refers exclusively to Writer procedures and in parallel have a new version of the plan that includes the replacements of the Writer procedures with PME references. This minimizes the risk as the old plan remains available for the operations.

Added Value for Preparation – Like Writer, the PME Procedure Editor can generate products. For SCOS-2000, these represent the Command Sequences. A Text-based approach is certainly less "clicky" then a forms-based editor and it is perceived as faster for editing (copy/paste) by the user.

While the Writer environment is Windows/Office based, the PME Editor offers a solution that is operating system independent (a Windows and a Linux version is currently available). Finally, the gradual introduction of PME procedures allows the team to get familiar with the new procedure format and when it comes to move from SCOS-2000 to EGS-CC, then the procedures **do not require any conversion**.

4.3 Moving from SCOS to EGS-CC

The move from SCOS-2000 to EGS-CC is complete when the database (MIB) is also converted to the TDM format via the MME. Any problem in the import is flagged within the MME. As explained in the previous paragraph, the PME procedures do not require any conversion.

TDM Validation – ASTRAL Prepare uses a local (to the user) workspace backed by SVN. This means that some "sandbox" actions can be done without affecting the team: the Database Analyst can perform & review the import at TDM level. Following this import, they are then able to run a consistency check on the entire PME procedure-set. If the tests have passed, the Database Analyst would check-in the new TDM version. If not, then they can decide to revert the "sandbox" data, fix the problems or inform the team with a report from ASTRAL-Prepare.

Obviously, this use case is not only valid for the move of MIB to TDM but also for any further update to the TDM. Any edit can be done locally by the Analyst and ahead of the check-in (and release to EGS-CC), he can have an assessment about the impact of the change on existing PME procedures.

5 Conclusion

Clearly, EGS-CC together with this framework bring a lot of added value for both future and existing mission looking to transition to new technologies. The paper showed how the transition to EGS-CC is eased with an upgrade path from the existing legacy SCOS2000 MIB & dedicated flight control procedures to TDM & PME format procedures that can execute autonomously or interactively, directly communicating with the EGS-CC.

6 References

[1] SpaceOps-2021,4,x1341, FAVOUR – A new generation of editors for the EGS-CC, Wolfgang Heinen, Steve Pearson, Francesco Sgaramella
[2] SpaceOps-2023, ID #292, Multi-Mission Spacecraft Operations by a single operator with minimal impact on science return, M. Edirimanne, E. Coe, M. Kirsch, D. Webert, I. Benson, T. Godard, U. Weissmann
[3] ESAW-2021, B5, PME – A New Lightweight Procedure Management Environment for future Ground Segments, W. Heinen, S. Pearson
[4] SpaceOps-2018, The New-Generation Toolset Providing Comprehensive Support for Mission Operations Preparation and Validation, S. Reid, W. Heinen, S. Pearson
[5] SpaceOps-2016, New Generation Mission Operations Preparation Framework, W. Heinen, S. Reid, S. Pearson