Focussuite

Focussuite WHITE PAPER

Prepared by:	***Focussuite*** team
	Digitally signed by Mercedes Sánchez Manzano Date: 2026.03.09 18:03:42 +01'00'
Approved by:	QA Manager
	Digitally signed by Felipe Jiménez González Date: 2026.09.08 08:50:51 +02'00'
Authorized by:	***Focussuite*** team
	Jesús Robles Aguera 2026.03.09 18:15:42 +01'00'

Code:	GMV-FOCUSSUITE-WP
Version:	3.0
Date:	06/03/2026
Internal code:	GMVAD 20463/12 V7/26

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

1. INTRODUCTION TO FOCUSSUITE

Focussuite is GMV's advanced Commercial Off-The-Shelf (COTS) flight dynamics solution for satellite operations. It provides a comprehensive and scalable framework supporting both operational flight dynamics activities and the development of customized mission-specific systems. The product addresses the concrete operational needs of satellite operators while simultaneously offering a flexible framework for system evolution, extension, and integration. ***Focussuite*** is positioned both as a ready-to-use COTS solution, minimizing deployment time and operational risk, and as a customizable framework enabling mission-specific adaptations without the cost and risk of fully bespoke development. This dual approach reduces programme schedule, costs, and technical risk while improving operational efficiency and minimizing human error. ***Focussuite*** is a mature, flight-proven solution capable of addressing diverse mission profiles and operational concepts. The system reflects extensive operational experience and has been validated across multiple satellite platforms and mission types.

GMV acknowledges the fact that every customer has different needs and requirements when it comes to covering their flight dynamics needs. This reflection comes from our privileged position as the main European supplier of flight dynamics systems and services after over two decades of uninterrupted successful services to a large variety of clients. Consequently, and consistently with this we are pleased to offer to the market ***Focussuite***, the ultimate flight dynamics solution for satellite operations. Our solution is characterized by the following unique high-level requirements:

- Advanced off-the-shelf, multi-mission, multi-satellite flight dynamics solution for flight dynamics satellite control that sets a new standard in functionality, reliability, flexibility and user friendliness.
- Capability to provide full lifecycle flight dynamics operations support through our unsurpassed collection of flight-proven mission independent and mission specific functionality. Flight-proven support of commercial satellite platforms is provided with accuracy fully consistent with native systems.
- Availability of a large collection of plug-and-play components providing unprecedented functionality and being able to customize the system to fit your specific needs. GMV provides custom solutions with strong customer focus and customer support.
- Provision of a generic framework that allows further product development and evolution, including the ability to integrate external applications with unprecedented ease. Our open framework dramatically boosts productivity, system usability, accessibility and stability.

Focussuite has therefore been conceived as a real framework as it is demonstrated by its major functionalities: **a computation layer** based on the extensive reuse of existing and improved software, a **client/server architecture**, a **database driven** system, an **advanced GUI** (based on common applications philosophy: "*everything-in-one-working-area*" and "*all-one-click-away*" and using a GUI toolkit that allows a development of GUIs through configuration files rather than through code), procedures automation capability through ***autofocus,*** advanced **graphical capabilities**, **portability** and **extensibility**.

Focussuite is designed with a forward-thinking approach, incorporating a robust architecture that ensures **scalability, flexibility, and seamless deployment** across Kubernetes standard environments, physical environments, and clouds. By adhering to industry best practices, we have crafted a solution that not only maximizes efficiency but also facilitates effortless deployment on multiple cloud providers and self-managed instances.

Focussuite supports all the functionalities required to perform the generic operations tasks, using **operational**, **flight-proven**, **off-the-shelf components** that have been used successfully in multiple space missions. This approach ensures **minimum risk**, **maximum reliability,** and **minimum cost** for the end customer.

Some of the key features of our product include:

- **Services Architecture**: Our product is built on a (micro)services architecture, dividing complex functionalities into smaller, independent services. This approach enhances flexibility, as each microservice can be developed, deployed, and scaled independently, promoting rapid development and easier maintenance.

- **Scalability at its Core**: Emphasizing scalability, our product is architected to handle varying workloads. By scaling specific microservices based on demand, our solution seamlessly accommodates growing user bases and spikes in traffic without compromising performance.
- **Best Practices**: Our product guarantees consistent environments and minimizes deployment issues. This streamlined approach not only **simplifies the deployment** process but also ensures seamless portability across different environments, making it effortless to deploy on multiple cloud providers and self-managed instances.
- **Flexibility Across Multiple Cloud Providers**: Thanks to our forward-looking design, our product can be deployed with ease on various cloud providers, offering our customers the freedom to choose the platform that best fits their needs. This **flexibility** avoids vendor lock-in and ensures a smooth migration experience, empowering businesses to adapt to changing requirements.
- **Seamless Deployment on Self-Managed Instances**: Recognizing the importance of on-premises solutions, our product is equally compatible with self-managed instances. This capability grants organizations the **autonomy** and control to host the application within their own infrastructure while still enjoying the benefits of our advanced microservices architecture.
- **Secured**: Our product is built on a comprehensive **security** framework, implementing multiple layers of defense to safeguard against potential threats and vulnerabilities. All sensitive data is encrypted both at rest and in transit, ensuring the highest level of data protection and privacy compliance. Our solution incorporates RBAC to enforce granular **access controls**, limiting user privileges based on their roles, thus mitigating unauthorized access risks.
- **Redundancy**: ***Focussuite*** is designed to provide both hot and cold redundancy.

Focussuite includes a suite of generic operational flight dynamics systems. They share the same architecture and many common components. Additionally, GMV has a long experience in developing customised ***Focussuite*** -based solutions for specific needs.

The existing framework and the multiple development tools associated to it make this process extremely efficient and reliable. GMV will be pleased to study with you ways to develop a ***Focussuite***-based solution for your own needs, both in operational and mission analysis contexts. At the same time, several **stand-alone products** based on ***Focussuite*** infrastructure are available. A good example is ***Focuscloseap***, a generic tool for automatic detection of close approaches. An independent document providing complete information on ***Focuscloseap*** is available at ***Focussuite*** web page: http://***Focussuite***.gmv.com/

2. REFERENCES

2.1. APPLICABLE DOCUMENTS

The following documents, of the exact issue shown, form part of this document to the extent specified herein. Applicable documents are those referenced in the Contract or approved by the Approval Authority. They are referenced in this document in the form [AD.X]:

Table 2–1: Applicable Documents

Ref.	Title	Code	Version	Date
[AD.1]				

2.2. REFERENCE DOCUMENTS

The following documents, although not part of this document, amplify or clarify its contents. Reference documents are those not applicable and referenced within this document. They are referenced in this document in the form [RD.X]:

Table 2–2: Reference Documents

Ref.	Title	Code	Version	Date
[RD.1]				

3. *FOCUSSUITE* GENERAL DESCRIPTION

3.1. SYSTEM DESIGN PRINCIPLES AND KEY REQUIREMENTS

The design of ***Focussuite*** is driven by a set of core technical and operational requirements that ensure long-term sustainability, interoperability, and operational efficiency.

Openness and **Interoperability** are fundamental principles of the system. ***Focussuite*** adopts open standards such as XML for product exchange and standard communication protocols for interaction with external systems. The system exposes a well-documented REST Application Programming Interface (API), enabling external systems to execute ***Focussuite*** functions programmatically. In addition, external applications can be integrated into the operational environment without requiring modifications to the core software or additional software builds. This ensures seamless interoperability within heterogeneous ground segment ecosystems.

Portability has been addressed by minimizing and encapsulating operating system–dependent modules. The selected technical solutions are compatible across Windows, UNIX, and Linux environments. Client and server components can operate on different operating systems in heterogeneous configurations. In addition, the web-based client enables access from any platform equipped with a modern browser, including desktop and mobile devices.

Flexibility, Expandability, and Scalability are achieved through a modular and service-based design. The system allows rapid updates and the inclusion of new modules without requiring a full rebuild. Containerized deployment further enhances scalability and adaptability, enabling deployment across cloud-based or on-premises infrastructures. Services can be independently scaled and configured to meet mission-specific workload requirements.

User Interface and Usability have been designed with operational efficiency in mind. The graphical user interface is highly configurable and supports simultaneous visualization and manipulation of multiple datasets. The web-based interface ensures accessibility and ease of use while maintaining performance and responsiveness.

Full Lifecycle Support is embedded in the system design. ***Focussuite*** supports both mission-independent and mission-specific functionalities across all operational phases, from early mission operations to end-of-life activities. GMV provides structured customization support when mission adaptations are required.

The system also supports multi-user, multi-satellite, and multi-environment operational concepts, enabling concurrent operations across constellations and distributed operational teams.

Finally, operational efficiency is reinforced through **advanced automation capabilities**, reducing manual intervention and increasing repeatability and robustness of operational processes.

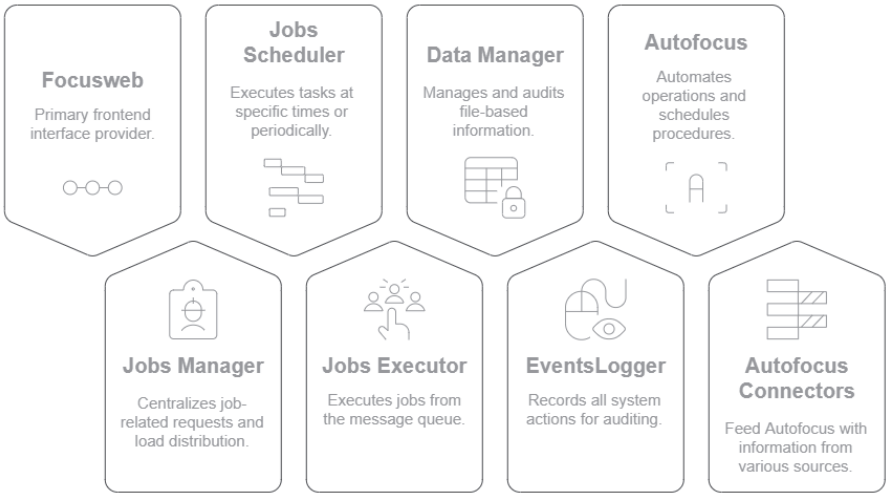
3.2. *FOCUSSUITE* ARCHITECTURE

Focussuite is based on a service-oriented architecture designed to provide high configurability while maintaining a set of core central services that ensure system coherence and integrity.

The backend is implemented as a Java Spring application, ensuring portability and ease of deployment across operating systems. It exposes a well-documented REST API that allows operators and integrators to develop custom scripts or external applications interacting directly with the ***Focussuite*** server.

The system has been engineered to simplify installation, deployment, and maintenance activities. GMV provides standardized deployment mechanisms using technologies such as Ansible, Helm charts, or Kubernetes manifests, depending on customer infrastructure preferences. This approach enables automated, reproducible deployments and allows the complete system to be installed from scratch with minimal manual intervention.

Figure 3-1: *Focussuite* services



The architecture is composed of the following main components.

- ***Focusweb*** provides the primary frontend interface and acts as an orchestration layer between the user interface and backend services. It adapts and routes incoming requests to the appropriate services while maintaining a consistent interaction model.
- ***Focussuite Jobs Manager*** centralizes the management of all job-related requests, including job creation, termination, and status monitoring. It persists job metadata in the database and uses a message queue mechanism to distribute workload among execution services.
- ***Focussuite Jobs Executor*** is responsible for executing computational processes requested by the system. It retrieves jobs from the message queue and performs execution accordingly. The executor scales dynamically based on workload. As a serverless component, it is instantiated only when required, optimizing resource utilization.
- ***Focussuite Data Manager*** manages all file-based information within the system. It provides file auditing, version control, conflict prevention, and historical traceability of data products, ensuring data integrity and operational consistency.
- ***Focussuite EventsLogger*** records all system activities, including manual and automated actions, for traceability and auditing purposes. The logged events are made available to aggregation services, enabling event-driven workflows and automated reaction mechanisms.
- ***Autofocus and Aggregation Service*** provides event-driven automation capabilities. It triggers job execution based on predefined rules and external or internal events, such as new data availability, completion of dependent processes, or detection of anomalies requiring corrective actions. External systems can publish events through the REST API. Scalability is ensured through message queue mechanisms that allow horizontal scaling when event volume increases.
- ***Jobs Scheduler*** executes tasks at predefined epochs or periodic intervals. Together with Autofocus, it forms the backbone of the system's automation framework, supporting both time-driven and event-driven operational concepts.
- ***Autofocus Connectors*** are a set of services that feed ***Autofocus*** with information from multiple sources. Connectors for EventsLogger and Jobs Scheduler are included in the baseline deployment, and additional connectors can be implemented as required.

4. *FOCUSSUITE* BACKEND CORE SERVICES

These services are the most important part of the system; it is the core of the FDS. This part of the system has been modularly designed to easily allow the integration of a new functionality in the FDS by the addition of new computational modules without changing any of the services. Just configuration changes are needed. Thus, the system is formed by a generic kernel, which contains the main functionalities of the server and defined modules containing suitable functions (for example procedures related to ***Autofocus***) oriented towards the management of the applications.

The next subsections are dedicated to describing each component in detail.

4.1. DATA MANAGER

The ***Focussuite Data Manager*** is the component that handles the *focusDB* data in ***Focussuite***. This data is all the data related to the Flight Dynamic functions, such as orbital information, attitude information or manoeuvres. Static and dynamic user configuration of the computational modules is also part of this service.

The Data Manager consists of two cooperating services:

- **State Manager Service.** Maintains the system state over time as an ordered sequence of snapshots, enabling linear progression and exact replay of past conditions. This allows, for example, comparison of the system state before and after an orbit determination, full traceability of inputs and outputs, and the ability to revisit or restore historical states for validation or analysis. A configurable retention policy controls storage growth by pruning older states by age or by count.
- **Data Storage Service.** Provides the persistent store for all datasets linked to those states. It ensures that flight-dynamics data (orbits, attitude, measurements, plots, and related artefacts) are securely stored, efficiently retrieved, and correctly associated with the corresponding state maintained by the State Manager.

The system allows the user to create branches derived from a source branch, the data in this branch is isolated, so actions performed in one branch do not affect other branches, thus allowing the execution in parallel of different scenarios. Branches can be created dynamically from the user interface.

In order to ensure the data integrity of the system a lock mechanism is provided by the state manager. Locks are applied per branch, so for example a new branch created from another branch will not be affected by the locks of the latter.

Locks are not required to advance a branch state into a new one. Locks can be of two types: write or read, following the next rules:

- Only processes with write lock can update an item in the state, generating a new derived state.
- Only processes with a read lock can read an item in the state, but many processes can apply for a read lock. i.e. an item can have many read-lock.
- A write lock cannot be applied to an item with a write-lock or read-lock.

Locks are optional and can be configured for each element (e.g. Orbit, panel) of a module execution (e.g. Propag, Bahn…) in order to balance parallelization and data integrity. The manual panel edition in the user interface might lock the underlying panel using the write lock this way it is ensured that what the user is modifying and visualizing is what will be stored in the system. After making use of a lock, the affected item lock is removed. In order to avoid problems with idle locks that might block the system, locks have a timeout that can be renewed in order to extend them if necessary, e.g. by a long running process.

As large amounts of data can be generated by the system, a cleaning mechanism is implemented to limit the stored data, this mechanism is applied at two levels:

- Data Storage Service: the items stored in the service can have an associated expiration date to remove them from the storage. Also they can be programmatically removed.

- State Manager: Branches are configured with a limit in the number of commits or days, based on these parameters old commits are removed and orphan items are then cleaned from the data-storages. Branches can also be deleted, although operational branches can be configured as protected to avoid deletion by mistake.

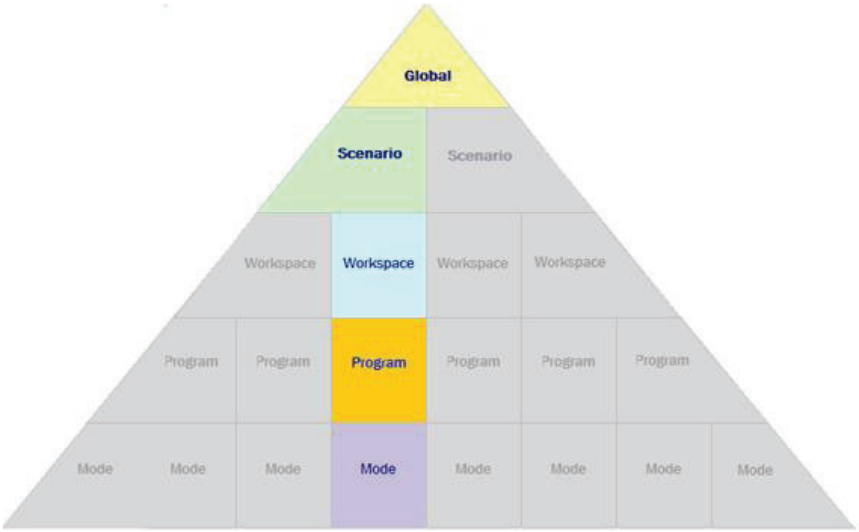
This component interacts with following ***Focussuite*** services:

- ***Focussuite Jobs Executor***: The ***Focussuite Data Manager*** receives the requests for retrieving information needed by the jobs -including computational layer- to execute the needed operations according to the business logic and the user configuration. After the execution the information processed by the jobs is sent back to the ***Focussuite Data Manager***.
- ***Focusweb*** request configuration and Flight Dynamics data for different purposes like configuration edition, results analysis, auditing or graphical representation of the Flight Dynamics data. In addition, information is sent to the ***Focussuite Data Manager*** in case of editing configuration or uploading new files by operators.
- ***Focussuite EventsLogger***: All the information related to the operations performed with the data such as information updated, new information upload or configuration changes is logged into the ***EventsLogger***.

In ***Focussuite***, data are organized in a tree structure as follows:

- A set of scenarios exists for a given ***Focussuite*** deployment. This concept is typically used to separate environments, such as operational and offline (simulation/testing/development); or to partition satellite groups, such as by planes or platforms, especially in constellations or large fleets. In such cases, segregated environments are common practices to ensure isolation between operational systems and non-operational activities (offline).
- Each scenario includes a set of workspaces. A typical use of the workspace concept is to define a different one per satellite.
- For every program associated to the workspace, several modes can be defined for each function to be performed by the executable. Each Mode can be understood as a configuration set of the program.
- Each mode includes a set of datasets. A dataset refers to a logically grouped set of related data (*focusDB* data). In standard ***Focussuite*** products, a dataset corresponds to a file, which can be of any type (e.g., plain ASCII, PostScript, XML) or a structured file in a standardized format known as TKF. The TKF format supports various data types, such as real numbers, integers, enumerated menus, and record lists containing fields of these types. For structured data, the user is provided with both a graphical user interface (frontend) and a REST API that is JSON-friendly, ensuring ease of integration and accessibility through both manual and automated means.

Figure 4-1: FDS scenarios and workspaces (focusDB levels)



In distributed deployments like constellations this data is saved in objects storage (S3 compatible).

The data manager is closely related to the application layer, managing the access of the computational programs to the database component, and storing the configuration of each separated module. This part of the system is also in charge of defining the final list of files to be used by any program. The configuration of the application layer is linked to each workspace and the data accessible by the different components is also limited by the workspace selection.

In terms of dynamic data used during the operations, it is composed by the following levels (notice that the data used by the programs and the data used by the databases are in different directories):

- Common level: This level contains all the data related to the programs, databases (FDS databases) and modules information (TKF files and data containing info from the panels contents) which are shared with all the different scenarios and workspaces.
- Scenario level: This level contains all the data related to the programs, databases (FDS databases) and modules information (TKF files and data containing info from the panels contents) which are shared within all the workspaces defined below that scenario.
- Workspace level: This level contains data related to the programs, databases (FDS databases) and modules information (TKF files and data containing info from the panels contents) only applicable to this workspace, it does not contain any shared data.
- Mode level: this level contains configuration data for each program mode which, as a first approach, is related to the program configuration.
- Incoming directory: This directory contains all the incoming files which are inputs for the FDS and come from other entities (RINEX files, EOP data, Telemetry...). It can be defined at common level, one per scenario or one per workspace depending on the needs of the customer.
- Outgoing directory: This directory contains all the outgoing files which are generated by the FDS and are sent to other entities (all different types of products).

The tree structure allows to define functions or data at a level higher than the workspace (e.g., scenario, or even global). They will be shared by all the workspaces under that branch. So, a certain function may be defined at the highest level (global), which means that all the workspaces in the system will share it. Another option is to define it at the workspace level, which means that it will only be available when the user accesses that workspace (e.g., satellite specific functions). Although most datasets are defined at the workspace level. Only some files (e.g., gravitation coefficients, EOPs, etc) and configuration defining information common to all the satellites are defined at a higher level.

This structure facilitates the scalability of the system, assigning different scenarios and workspaces to each S/C in the mission. Special care must be taken in this case defining the data files in a level accessible for all the workspaces that may require it.

Given that the system allows multiple users working at the same time, a system of locks and consistency is needed to avoid conflict due to concurrent access to same files. The system includes warning and conflict detection mechanisms to prevent concurrent access from causing inconsistencies. It's important to note that such a system could limit parallelization, which is why hard-lock mechanisms are generally not implemented. The exception would be for information where the benefits of a hard lock significantly outweigh the limitations of parallel use. In any case, this feature would be configurable, so it can be changed on user needs as explained at the beginning of this section.

4.2. JOBS MANAGER

The ***Focussuite Jobs Manager*** is responsible of centralizing all the job requests, maintain their statuses and logs (standard output and error). In addition, provides capabilities to cancel/stop the jobs and gives information about the status of the jobs queue like size or average time in queue.

This component interacts with following ***Focussuite*** services:

- ***Focussuite Jobs Executor***: The Jobs Executor sends the standard output and error as well as the status to the Jobs Manager which is responsible for saving it. The Jobs Manager provides the API so the executor can interact with.
- ***Focusweb***, ***FocAS*** and ***Focussuite Jobs Scheduler***: The former creates job requests in the Jobs Manager. An API is provided so that they can add, cancel, get status or get logs of any job. Potentially, the API could be used by any new element or service to add new jobs or check status. This element can be both external or internal.
- ***Focussuite EventsLogger*** to notify all events related with the relevant actions performed internally.

This component interacts with the following horizontal services:

- Message Broker: It publishes jobs into the jobs queue that will be consumed by the Jobs Executor. In addition, it creates two exchanges so that ***Focusweb*** can be subscribed to receive real time notifications of the logs and status of the different executions. Any interested actor can be also connected to this exchange.
- Relational Database: The job requests are saved here for future auditing. It saves who created it and when. In addition, it saves the history of the different status and the logs. In this way auditing is possible through the history. Cleaning mechanisms are put in place to avoid filling the disk space.
- Authorization and authentication service: To check if the request can be authorized according to the role of the user as well as the organization the user belongs to. For example, a user tries to execute a module in a workspace that is not authorized.

4.3. JOBS EXECUTOR

The ***Focussuite Jobs Executor*** is responsible of executing all the jobs triggered either by user requests or automated processes such as ***Autofocus***. These jobs correspond to Flight Dynamics tasks and involve the execution of computational programs from the Computational layer as well as Python-based procedures defined in ***Autofocus***. For each job, the service creates an isolated temporary execution environment and retrieves all necessary input files from the Data Manager. It also monitors the execution process, forwarding job status and logs to the Jobs Manager, and supports user-initiated termination of running tasks. The service can handle various types of jobs, including:

- ***Focussuite*** Computational Programs: These programs implement most of the required functionalities and all purely related to Flight Dynamics computations. Further details of these programs are provided in Section §8.

- ***Autofocus*** procedures: The Jobs Executor runs ***Autofocus*** procedures, which are implemented entirely in Python. These procedures use the Focuspy library to interact with ***Focussuite*** via the Focusweb RESTApi, enabling full automation of Flight Dynamics operations. Any task that a user can perform manually through the interface can also be scripted and executed automatically, allowing seamless integration into mission workflows

The service is designed to make the addition of new kinds of jobs straightforward so it can be easily extended to cope with new functionalities on demand.

This component interacts with the following ***Focussuite*** services:

- ***Focussuite Jobs Manager***: The Jobs Executor sends the standard output and error as well as the status to the Jobs Manager which is responsible for saving it.
- ***Focussuite Data Manager***: The Jobs Executor retrieves the data needed for executing the *job* from this component.
- ***Focussuite EventsLogger*** to notify all events related with the relevant actions performed internally.

This component interacts with the following horizontal services:

- Message Broker: It consumes Jobs from the jobs queue generated by the Jobs Manager (producer). It creates a specific queue for instance to read cancel request from that queue.

4.4. EVENTSLOGGER

Focussuite EventsLogger is the events logging facility available as either stand-alone or fully integrated module for all ***Focussuite*** products. It is in charge of receiving, archiving, managing and distributing all information, warning or error messages to operators of all the events that are happening in the system. It recollects information from all the ***Focussuite*** Services. Although it is an important component the system can work regardless of the status of this component.

This component interacts with all the ***Focussuite*** services receiving events from them. In addition, it provides these events to ***Focusweb*** for graphical representation as well as to the ***Autofocus*** so any event can potentially trigger an action on the system. Finally, all events are forwarded to ***CentralLog***, ensuring their availability within the centralized event logging and alerting system.

This component interacts with the following horizontal services:

- Relational Database: The events are stored here during a configurable period. Therefore, the user can inspect what happened in the system in the past. Cleaning mechanisms are put in place to avoid the disk to be filled up.
- Authorization and authentication service: To check if the request can be authorized according to the role of the user as well as the organization the user belongs to. Only the information the user can access is provided.
- Message Broker: The real time information (Events received) is posted here so it can be consumed by the ***FocAS*** instances.

4.5. ***FOCUSSUITE*** NOTIFICATIONS SERVICE

Focussuite Notification Service provides a real-time channel for notifying clients about system changes. Using secure WebSocket (WSS) connections, it delivers immediate updates to front-end applications and other subscribers without polling. Subscriptions are filter-based to reduce overhead and send only relevant notifications to each client (e.g., a client may subscribe only to updates of the *operational orbit* or to changes affecting a specific dataset). Although important for user experience and timely awareness, the platform continues to operate if this component is unavailable; clients can fall back to on-demand queries.

This service consumes signals from all ***Focussuite*** components and routes them to interested parties. It feeds ***Focusweb*** for real-time UI updates. This component interacts with the following horizontal services:

- Authorization and Authentication Service**.** Validates user identity and enforces access control by role, organization, and data scope. Only notifications the user is allowed to see are delivered.
- Message Broker**.** Receives change signals, which are filtered and fanned out to subscribed clients in real time.

This component interacts with all the ***Focussuite*** services through RabbitMQ exchanges.

4.6. FOCUSADMIN

The main task of this component is the definition of the user permissions and roles of the FDS at application level, allowing the FDS administrator to define which operations can be performed and which data can be access by each user.

Access control in the ***Focussuite*** environment is designed to provide a scalable, attribute-based mechanism for managing user permissions across missions, constellations, and satellite platforms. The configuration combines principles of Attribute-Based Access Control (ABAC) and Role-Based Access Control (RBAC) to simplify administration while maintaining explicit, auditable policies.

Each **workspace** in the system (typically associated with a satellite or mission) is characterized by a set of **domain attributes**—such as *Mission*, *Platform*, *Constellation*, or *Environment*—that define its operational context. Users and groups are assigned permissions over attribute values rather than individual items, allowing large satellite fleets to be managed with minimal configuration effort. For example, a user may have read-only access to all workspaces with attribute Mission:Mission1 but write access to those with Platform:Platform2. No permissions are granted by default; all access must be explicitly declared.

User privileges are derived from both **individual** and **group attributes**, which are consolidated upon login and propagated within the **JWT token** issued by Keycloak. The token carries the user's permitted attributes (e.g., program:bahn.RWX, mission: Lightspeed.R, application:dashboard.W), which are validated by each ***Focussuite*** components before granting access to data or services.

Programs, datasets, and operational procedures also inherit attribute tags (e.g., platform, program_group, mode) that determine their visibility and permitted actions. Access rules are defined through simple evaluation heuristics, such as verifying the user's attribute set for required read (R), write (W), or execute (X) rights. This approach eliminates the need for extensive per-item permission matrices and allows consistent enforcement across all components—Data Manager, Jobs Executor, Autofocus, and EventsLogger.

Administrative roles are predefined:

- Admin users: Full access to all attributes and operations.

Typical use cases include granting an operator read access to operational data without modification rights, assigning mission-specific permissions to user groups, or limiting write access to the operational environment while maintaining full read visibility.

This design ensures:

- Explicit and easily auditable policies
- Low maintenance overhead for expanding fleets
- Hierarchical and environment-aware authorization
- Compatibility with federated identity systems (Keycloak JWT)

FocusAdmin allows to configure the following:

- Attributes for the system elements such as scenarios, workspaces, applications.
- FocusAdmin also allows to configure in a user-friendly interface the access privileges associated to the different attributes for each users group and assign a user to one or several groups.

4.7. *AUTOFOCUS*

Autofocus is the automation engine of ***Focussuite***, designed to execute Flight Dynamics operations based on time-based schedules or external event triggers. It enables the orchestration of complex workflows through Python-based procedures, allowing seamless integration with both internal components and external systems. ***Autofocus*** is composed of two main services: the Jobs Scheduler, which manages scheduled executions, and the ***Focussuite*** Aggregation Service (FocAS), which reacts to event-driven triggers. Together, they provide a flexible and extensible automation framework for mission operations.

4.7.1. JOBS SCHEDULER

Focussuite Jobs Scheduler services will be used to cover all the requested ***Focussuite*** scheduler tasks. This component includes all the services needed to handle the automatic execution of all computational FD components. The ***Jobs Scheduler*** provides the following functionalities:

- Execution of ***Autofocus*** sequences based on time constraints:
  - Run once at fixed epoch.
  - Run at fixed epoch and repeat periodically.
  - Run it given a crontab expression.
- Generate events at epoch given by a crontab expression so a trigger can be executed. In this way multiple ***Autofocus*** scripts can be executed at same crontab expression.

This component interacts with following ***Focussuite*** services:

- ***Focussuite EventsLogger*** to notify all events related with the relevant actions performed internally.
- ***Focussuite Jobs Manager*** to execute ***Autofocus*** procedures and/or computational programs.
- ***Focussuite Aggregation Service***: To publish events periodically based on crontab expressions.

This component interacts with the following horizontal services:

- Relational Database: It is used to store scheduling information: who created the scheduling, who edited it or the periodicity of the job. In addition, auditing information about historical activities executed by the service is saved.
- Authorization and authentication service: To check if the request can be authorized according to the role of the user as well as the organization the user belongs to.

4.7.2. *FOCUSSUITE* AGGREGATION SERVICE

The ***Focussuite Aggregation Service*** (***FocAS***) is a service that is continuously listening for events from different sources like ***Focussuite Events Logger***, ***CentralLog*** or Apache Kafka (Topics Consumer) and it is able to trigger the execution of ***Autofocus*** procedures according to the information of these events.

The design provides an extensible framework at all levels: Event Sources, Trigger Conditions and Actions (aka. ***Autofocus*** procedures).

The triggers and ***Autofocus*** procedures. The ***FocAS*** provides a REST API. The different components can be modified without need of restarting the system. It is also integrated into the web HMI so users can configure them in a user-friendly way.

This component interacts with following ***Focussuite*** services:

- ***Focussuite Connectors***: It is also known as Event Source Subscribers, and they are composed of different services that connect to different streams like ***CentralLog*** or ***Eventslogger***. Each of the connectors is an independent element so that they can be easily scaled.
- ***Focussuite Jobs Scheduler***: This component is able to produce events, defined by Crontab jobs.
- ***Focussuite Jobs Manager***: After receiving an event this is checked against the triggers database. If the event matches any trigger a job will be executed. A task will be created into the ***Jobs Manager*** including the event and trigger information so that it can be potentially used by the ***Autofocus*** procedure.

Focussuite EventsLogger to notify all events related with the relevant actions performed internally.

This component interacts with the following horizontal services:

- Relational Database: It is used to store scheduling information: who created the scheduling, who edited it or the periodicity of the job. In addition, auditing information about historical activities executed by the service is saved.
- Message Broker: The events are published here so it can be consumed by the ***FocAS*** instances. It works as an external queue to manage all the event. It helps the services to be replicated since different events are automatically distributed between different replicas. ***FocAS*** provides a library to publish events directly on the queue or via the Rest API. Depending on who publishes the messages a different solution will be used for security reasons. For internal services the events will be directly published into the queue. For external services, the events will be published using the REST Api that will be integrated with the Authorization and Authentication service.

4.7.3. *FOCUSSUITE* AGGREGATION SERVICE CONNECTORS

Group of services that feed the ***Focussuite Aggregation Service*** information from different sources, the following ones are expected to be deployed:

- ***Focussuite EventsLogger***: The component itself feeds with this information. By doing so, it enables any action within the system to potentially trigger another action. It is included here by completeness although its works it is not just limited to the connector task. Refer to section §4.4.
- ***CentralLog***: It stores and displays relevant event messages, including alerts and information messages also referred to as events, generated from the different control centre components during satellite operations. The components that publish messages in ***CentralLog*** includes all the GMV's solutions such as ***Hifly***, ***Magnet***, ***Autofly***, ***Flyplan*** or ***Flexplan***.

4.7.4. *FOCUSPY*

Focuspy is a Python wrapper around the *Focusweb* REST API, designed to provide a simplified and user-friendly interface for interacting with *Focussuite*. It implements the full set of API functionalities, enabling complete automation of operations that would otherwise be performed manually through the ***Focussuite*** GUI. By abstracting the complexity of direct HTTP requests and session handling, Focuspy offers a more accessible and Pythonic way to manage data panels, execute programs, handle files, and schedule tasks within the ***Focussuite*** environment. Additionally, it is fully integrated with *Autofocus*, allowing seamless use in procedure scripts and automated workflows.

5. FOCUSWEB

Focusweb is the Human-Machine Interface (HMI) layer of ***Focussuite*** and constitutes the primary user interaction component of the system. It acts as a middleware layer between the browser-based frontend and the underlying ***Focussuite*** backend services, abstracting service complexity and exposing functionality in a manner aligned with operational workflows.

Focusweb provides its own user-oriented API, built on top of the core ***Focussuite*** REST services. While backend services expose low-level operational and computational interfaces, ***Focusweb*** aggregates, orchestrates, and adapts these services into higher-level operations consistent with user expectations and operational procedures. This layered approach ensures a clear separation between core computational services and user interaction logic, improving maintainability, security, and long-term evolution.

The user interface is fully browser-based and built on the ***Focussuite*** framework, ensuring a consistent look-and-feel across all modules. The web client supports the complete lifecycle of Flight Dynamics operations, from routine activities to automation supervision and mission configuration management. Because the client runs directly in a standard web browser, no local installation is required, enabling flexible deployment across thin clients, workstations, laptops, and mobile devices.

Focusweb is designed to fulfil the following high-level requirements:

- Support for the full lifecycle of Flight Dynamics operations, including scheduling and automation through ***Autofocus***.
- Management of multiple mission scenarios.
- Safe and controlled editing of configuration data, Autofocus procedures and system databases.
- It provides graphics utilities: plots, alphanumeric displays, dashboard, Gantts, etc.
- Embedded Event Logger to maintain a complete history of actions performed by users and services
- Export functionalities to facilitate issue investigation outside the customer’s secured environment.

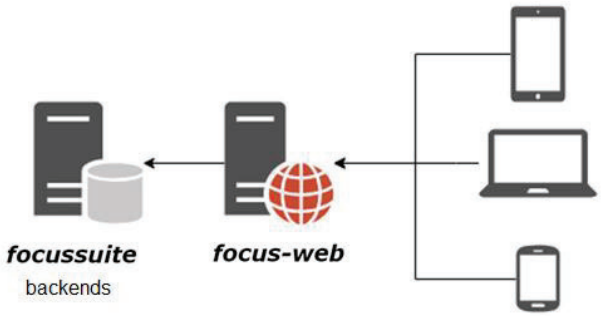
The web client includes the following HMI components:

- Focusweb client user interface
- Dashboard including Gantt charts, Plots, configuration data
- Eventslogger.
- Autofocus including procedures editor.
- ***Focussuite*** Jobs Manager.
- Visualfocus.

Communications between the web navigator and the backends are encrypted and securely transmitted via https using TLS1.3. The interface is exposed as a REST API with real-time events delivered through a STOMP WebSocket secure connection.

The web client is a frontend application that runs locally in the browser, this allows great flexibility in terms of deployment as web-browsers are available in a wide range of devices, such as thin clients, laptops, and mobile phones. This client is exposed by a backend (Focusweb) that also provides a secured interface with all the ***Focussuite*** backend services via a REST API.

Figure 5-1: FDS Client Side- Backends interface



The ***Focussuite*** client (***Focusweb*** client) provides the HMI which allows the operator to execute all FDS modules. The HMI allows the operator to set the inputs for a module, run the module and view the outputs, including plots and Gantt charts where applicable. The client also incorporates some basic features, like copy and paste, find and other state of the art functionalities.

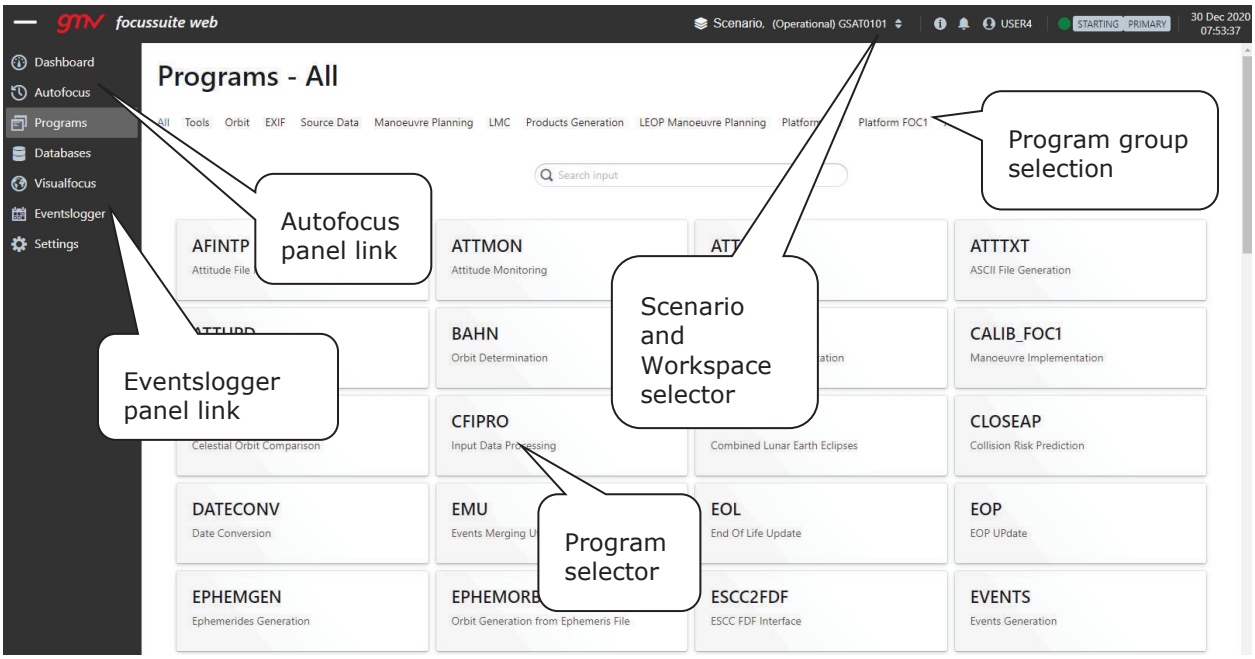
Users access the client by navigating to the ***Focusweb*** backend URL in a compatible web browser. Upon connection, the user is redirected to a login window. Authentication can be handled either via the system's native login or through Keycloak when OAuth2/SSO authentication is enabled.

Figure 5-2: *Focussuite* native login window

After logging in the user can view all the available workspaces. The workspace can be selected at any time from the navbar menu, allowing to explore different workspaces within the same user session.

A screenshot of the main client HMI is shown in the figure below.

Figure 5-3: Focussuite HMI



The ***Focussuite*** client has been designed to have an 'all in one' working area in order to minimise the number of screens and windows, and the overall GUI has been carefully designed to allow a high degree of efficiency and reliability. The ***Focussuite*** working area includes four major elements as follows:

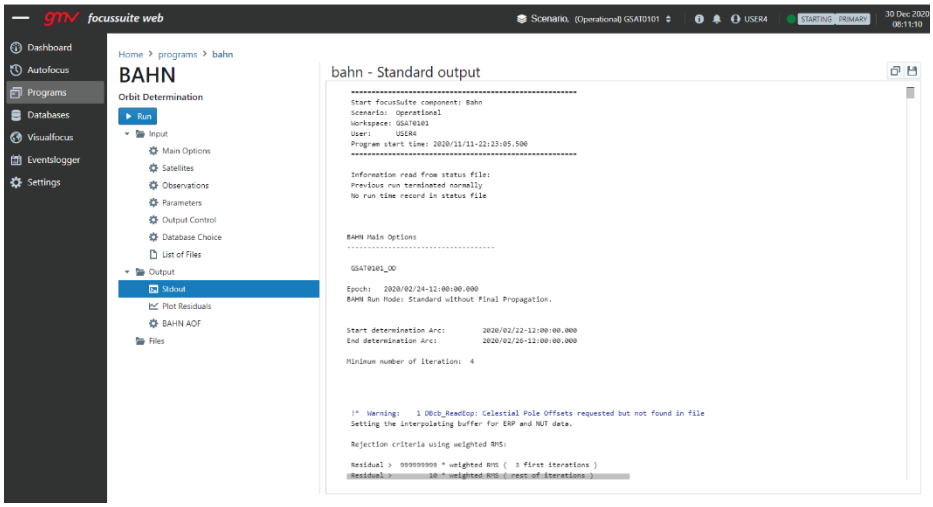
- Navigation bar with workspace, user and general information.
- Application selection menu on the lateral side, allows the user to select the desired application, e.g., programs, dashboard, Autofocus.
- Application panel, showing the interface of the selected application.

The complete ***Focussuite*** set of functions are logically grouped at the time of setup (the product allows full flexibility on the number of integrated functions and how they are grouped together. This flexibility is allocated at the time of the design, and it is not user configurable) in a set of tabs that the user can select in order to switch from one set of functions to another.

Within each programs tab, there is a set of computational programs or functions. To go from one program to another, the user shall simply 'click' on the corresponding accordion element. This deploys the main controls associated to the corresponding function, namely:

- Access to input datasets
- Button for execution of the corresponding program
- Access to output datasets, plots, etc
- Selection of a component element is shown in the figure below.

Figure 5-4: *Focussuite* main window program selection



To run a program, the user shall simply press the run button from the corresponding panel. This initiates program execution through the corresponding Process Manager. Once the corresponding program has been initiated, the Run button deactivates and "Stop" activates instead.

HMI provides the user with a flexible User Interface. It shall use the ***Focussuite*** framework and will provide its look-and-feel. It supports, in a user-friendly way, all interactions between the operator and the other components, supporting the following activities:

- System configuration and scenario management. In particular the user interface allows the user to swap between scenarios.
- Input data definition and maintenance, with a distinction between:
  - Input data for the different FDS functionalities.
  - Data stored in the FDS database.
  - Access to results, and when applicable access to a graphical representation of them. In particular it can display on a canvas any plot generated by the system.
- Access the online Help. This consists of two components:
  - Help pages.
  - Help line displaying a brief explanation of the input parameter being focused with the pointer, for input data.
  - Means to send outputs, plots, and panel snapshots to a Postscript printer or file.

From the interaction point of view, the execution of tasks is performed according with the approach established in FDS. The basic unit in the implementation is the module, which is formed by an executable binary and several associated data files.

For a manual user, the HMI presents for each of the programs a list of files representing the specific data needs (input and output) for that program. Upon submission of the task, the list of files is automatically generated based on the program specific files and the database files, which are common for every configured program.

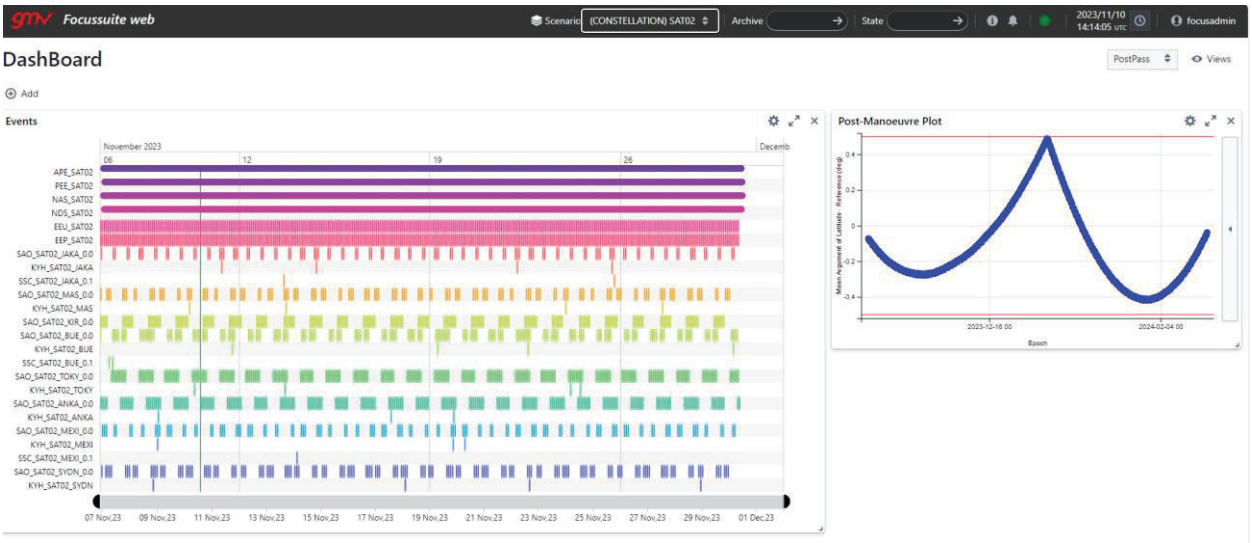
5.1. *DASHBOARD*

Dashboard is a configurable panel that allows the representation for different data sources to the user.

Figure 5-5: Dashboard visualization

The ***Dashboard*** is divided into different views, each of which can contain a series of elements or widgets that may or may not be linked to the current workspace (satellite). If they are linked to a workspace, the user can switch workspaces, and the content will automatically update to reflect the status of the selected workspace

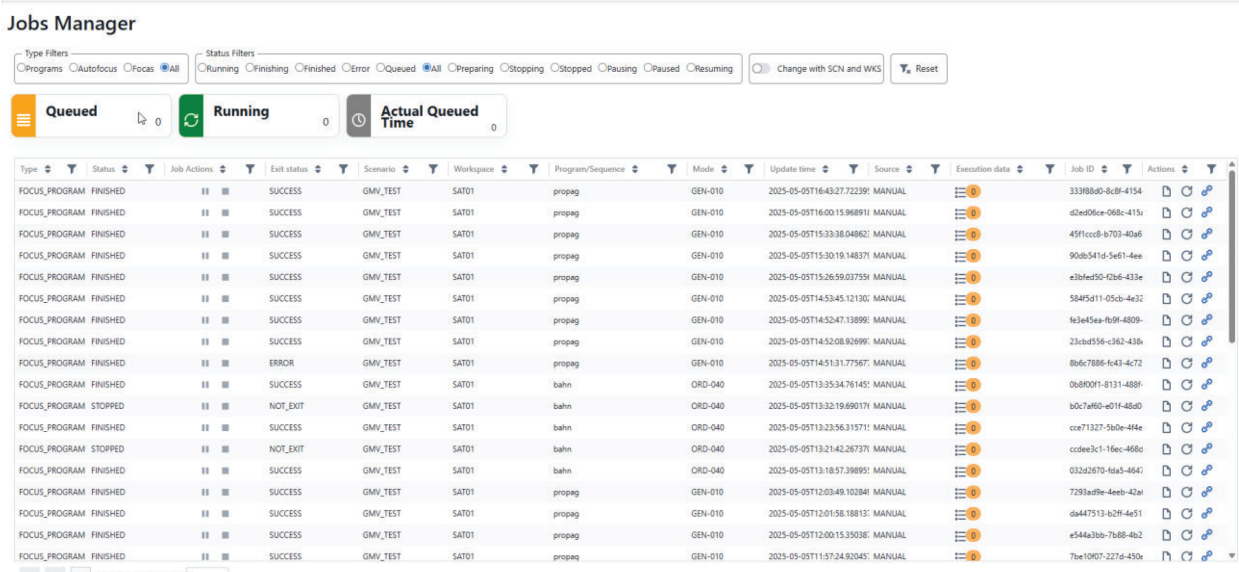
Figure 5-6: Dashboard visualization



5.2. *JOBS MANAGER*

It provides a centralized interface for monitoring the global status of all executions, offering real-time insight into ongoing and completed tasks across the system. Through this view, users have direct access to job details, including configuration parameters, execution logs, and controls to restart or cancel jobs as needed.

Figure 5-7: *Jobs Manager* main window.



6. *AUTOFOCUS*

The ***Autofocus*** application has been designed to be embedded in the main front end application (Focusweb). The ***Autofocus*** component is used to create and edit procedures, schedule them, and view the results of the procedures.

Autofocus application provides the human machine interface for the scheduler package of the flight dynamics system. It provides two different user environments:

- Procedures tab view: integrated sequence definition environment allows to write, to validate and to test sequences.
- Agenda tab view: It allows monitoring and controlling scheduled procedures as well as running procedures.

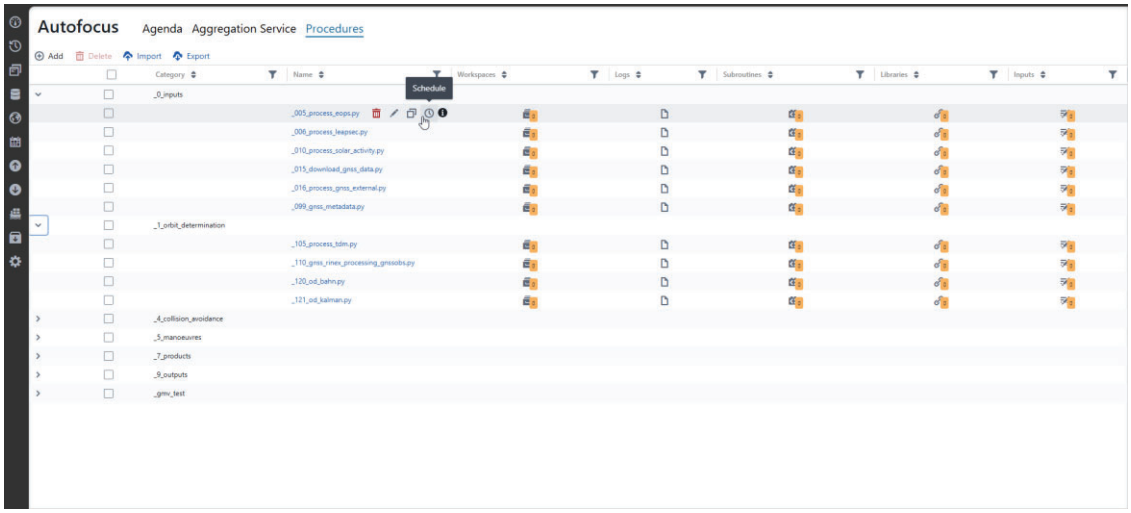
Aggregation Service View: It contains 3 tabs

- Scheduler tab view: For configuring periodic events generation. It includes a Gantt chart to display the scheduling time for every programmed sequence.
- Triggers tab view: For creating and editing triggers.
- Connectors tab view: It allows users to check the status of the connectors and pause/resume them.

Procedures tab

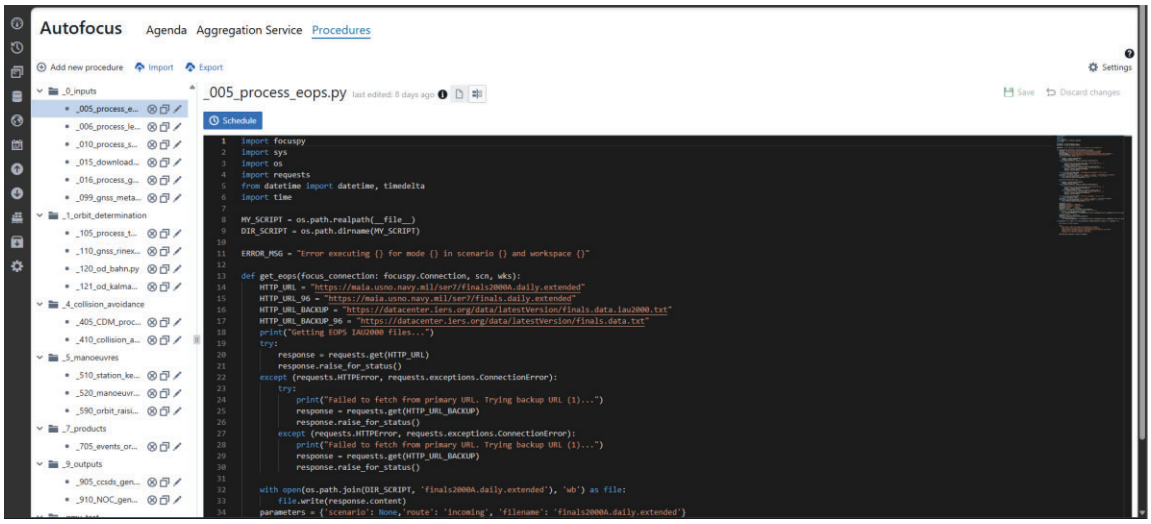
The following figures show the procedures views.

Figure 6-1: *Autofocus* main window –Procedures view



The procedures for ***Autofocus*** are written in Python.

Figure 6-2: *Autofocus* main window –Procedures editor view

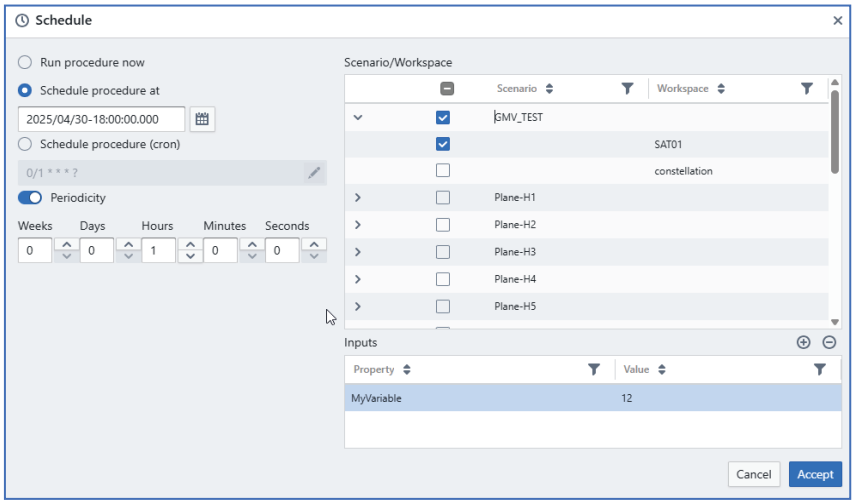


The Procedures tab contains all the defined procedures for the actual scenario, alphabetically sorted by category.

The procedures can be used to run any of the ***Focussuite*** modules, either individually, or with multiple modules per procedures. The configuration of each module can be directly modified by the ***Autofocus*** procedure to customize the execution of the program. Similarly, after the module runs, there are several parameters that can be verified by the procedure to make sure that they are within user-defined thresholds. If any threshold is violated, an alarm can be triggered, and several actions can be programmed.

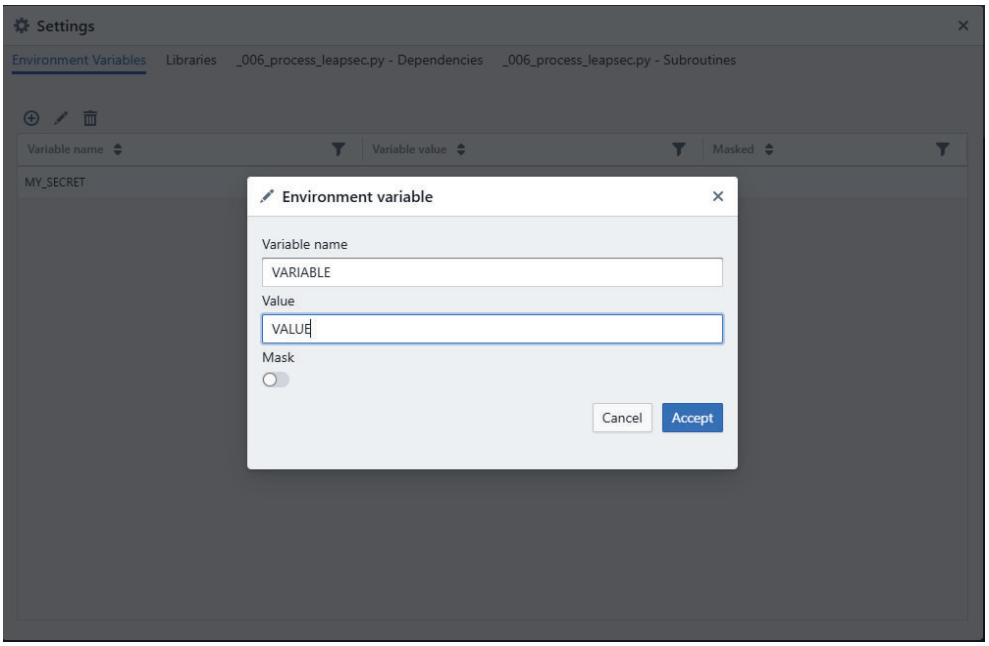
A procedure can be scheduled to occur at a specific time and can be set with a configurable periodicity so that it gets automatically rescheduled.

Figure 6-3: Autofocus schedule dialog



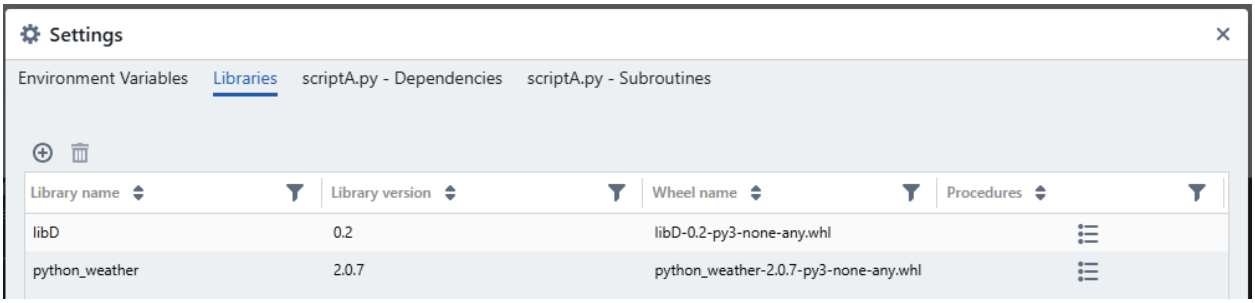
Apart from scheduling, it is also possible to define environment variables that can be used by the system. These environment variables can be optionally masked to hide sensitive content. While these variables are accessible to scripts during execution, their values are hidden from the user and can only be overwritten, not viewed.

Figure 6-4: Autofocus. Procedures. Settings Dialog. Environment variables.



Additionally, both internal and third-party external libraries can be imported and made available to any procedure that declares them as an associated library. This enables reusability, modularity, and the integration of external functionalities across different procedures.

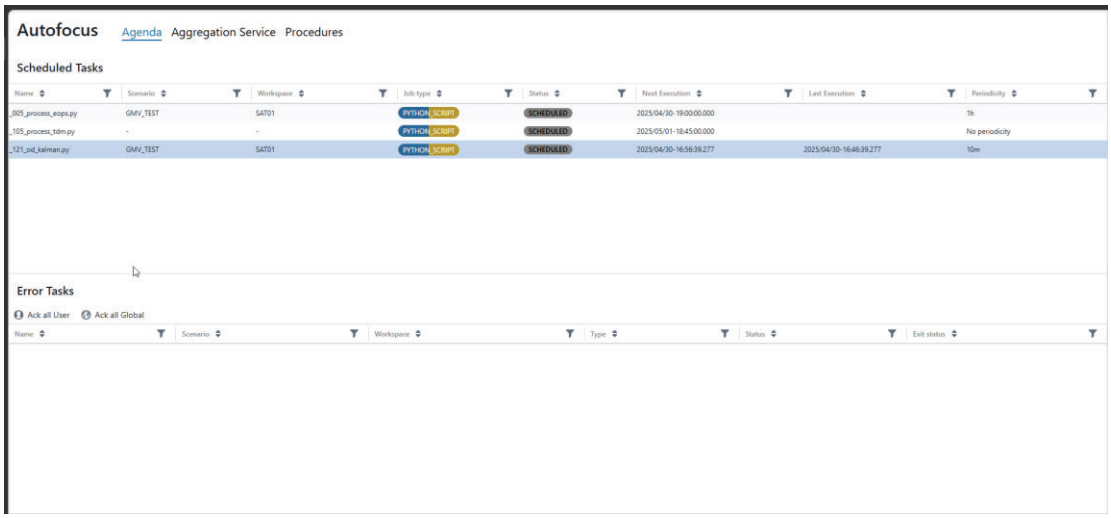
Figure 6-5: Autofocus. Procedures. Settings Dialog. Settings variables.



Agenda Tab

The agenda tab contains all the currently scheduled procedures for the actual scenario, alphabetically sorted. Details are provided on whether the procedures can be automatically rescheduled at certain times.

Figure 6-6: *Autofocus* main window – Agenda view.

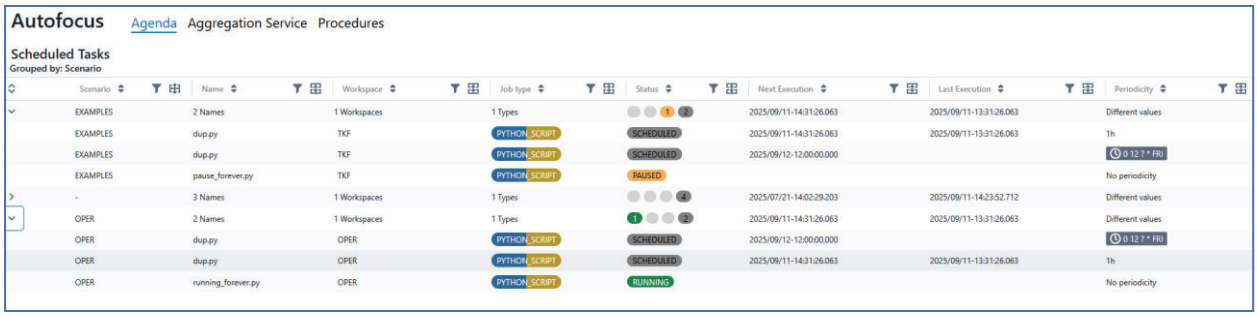


The agenda supports row grouping by column values. In practice, this means that instead of displaying multiple rows for the same entity, it can collapse them into a single aggregated row. For example, the agenda can be configured to show:

- One row per workspace or scenario (e.g., a single row representing a satellite).
- One row per procedure, consolidating all related entries.

This grouping capability improves readability and provides a higher-level overview, while still allowing users to drill down into the underlying details when needed.

Figure 6-7: Autofocus. Grouping capabilities.



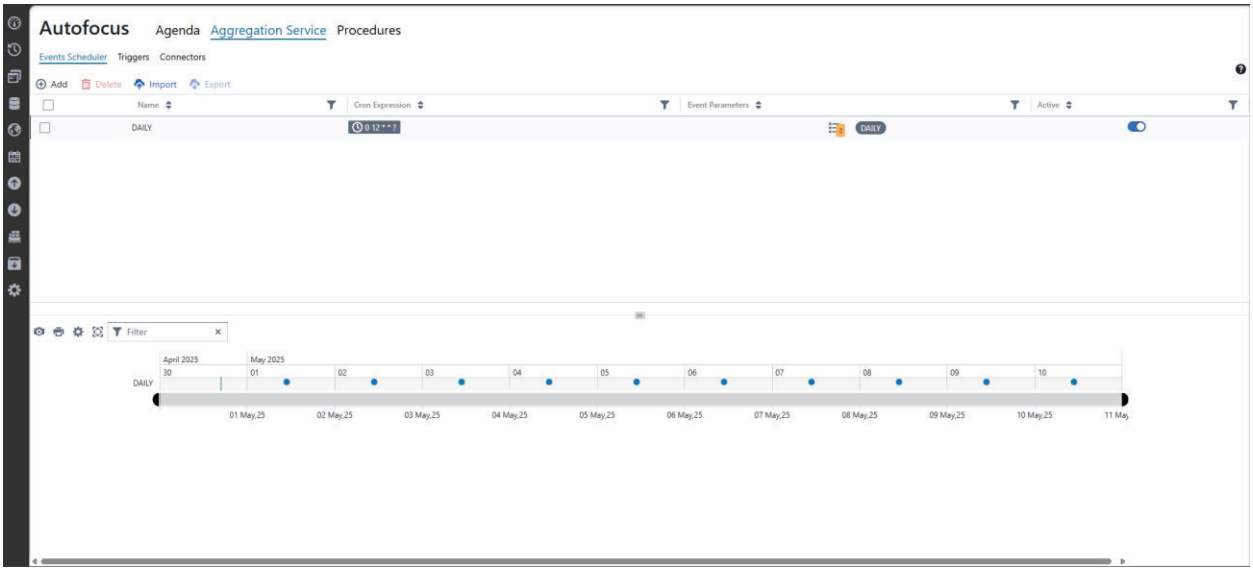
Additionally, a summary of scheduled tasks that ended with errors and were not yet acknowledged is displayed in the lower section of the interface, allowing users to easily review and acknowledge any issues once analysis is complete.

Aggregation Service Tab

The Aggregation Service tab is used to create and manage triggers, configure periodic events, and monitor the status of event connectors.

Periodic events are configured using cron expressions. At each defined execution time, an event is generated and evaluated against the registered triggers to determine whether any should be activated.

Figure 6-8: *Autofocus* main window. *Aggregation Service* Events Scheduler.



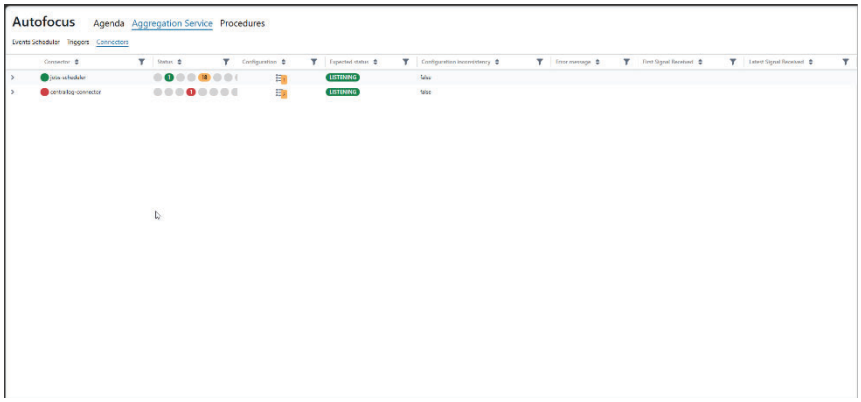
Triggers are defined in a dedicated section. Each trigger consists of an event condition (e.g., an event from CentralLog with the message "Manoeuvre cancelled") and the procedure to be executed in response.

Figure 6-9: *Autofocus* main window. *Aggregation Service* Triggers View.

Similarly to the Agenda view, there is a dedicated interface to review any unacknowledged error conditions resulting from procedures executed via triggers.

Finally, the Connectors view provides real-time status monitoring of the available event connectors.

Figure 6-10: *Autofocus* main window. *Aggregation Service*. Connectors view.



7. *VISUALFOCUS*

Visualfocus is GMV's ***Focusweb*** component able to display the flight of one or more satellites in real or simulated time. The different displayed elements, such as input orbits, attitudes, and events, are obtained from the configuration files stored in the ***Focusweb*** server data structure. Every scenario contains a database and one or more configuration panels. The user can use all the available elements (planets, satellites, ground stations and sensors) defined in its scenario database for every configuration contained in that scenario. The modified databases and configurations are automatically saved in the server so other users may re-use them. When the configuration is ready, the user may launch the graphical tool and the graphical display is started.

The time evolution can be scaled to make the simulation run faster than real time (a time factor scale 1 means real time speed), and even to move backwards (negative time factor). This time factor can be also changed from the graphical windows once they are launched. It is also possible that ***Visualfocus*** reads the simulation time directly from the host machine's clock. In this case the simulation will progress at real-time rate, and it will also be synchronized with current time reference.

Graphical representation of selected elements is performed in 2D and 3D, with capability to switch between them during the simulation:

3D visualization tool: This view supports/includes:

- A three-dimensional view of the Earth and celestial objects (e.g. planets, Sun, Moon).
- A 3D model of the satellite
- Satellite orbits and attitude representation.
- Sensors field of view (conical/pyramidal/polygonal).
- Ground stations, with visibility cone.
- Station-satellite and satellite-satellite links.
- The user can control the position of the camera.

2D visualization tool: This view supports/includes:

- Ground-track of all active satellites and sub-satellite point.
- Day/night terminator, eclipses, Moon shadow projection.
- Ground stations, their visibility areas, and links to visible satellites.
- The swath of the sensors pointing towards the Earth.

Figure 7-1: *Visualfocus* 3D Display (Earth-centered)

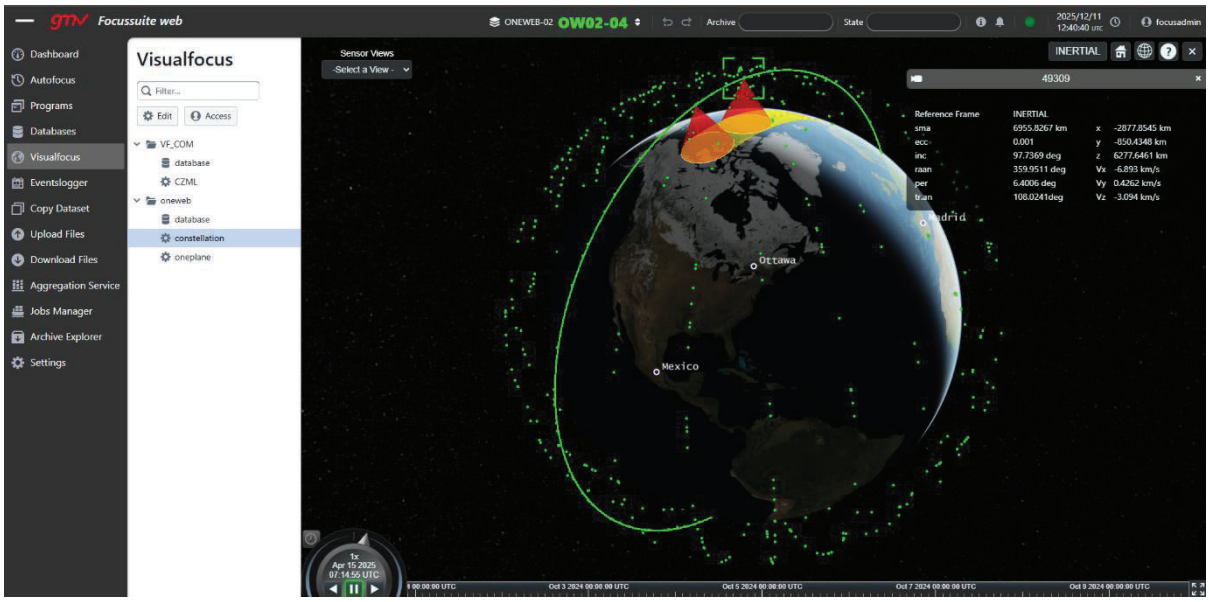


Figure 7-2: *Visualfocus* 3D Display (on-board camera)

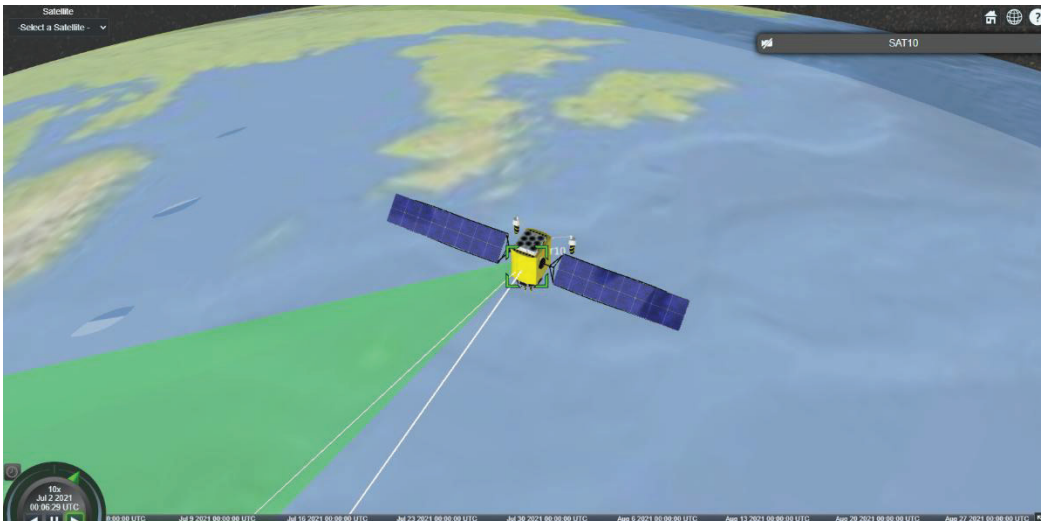
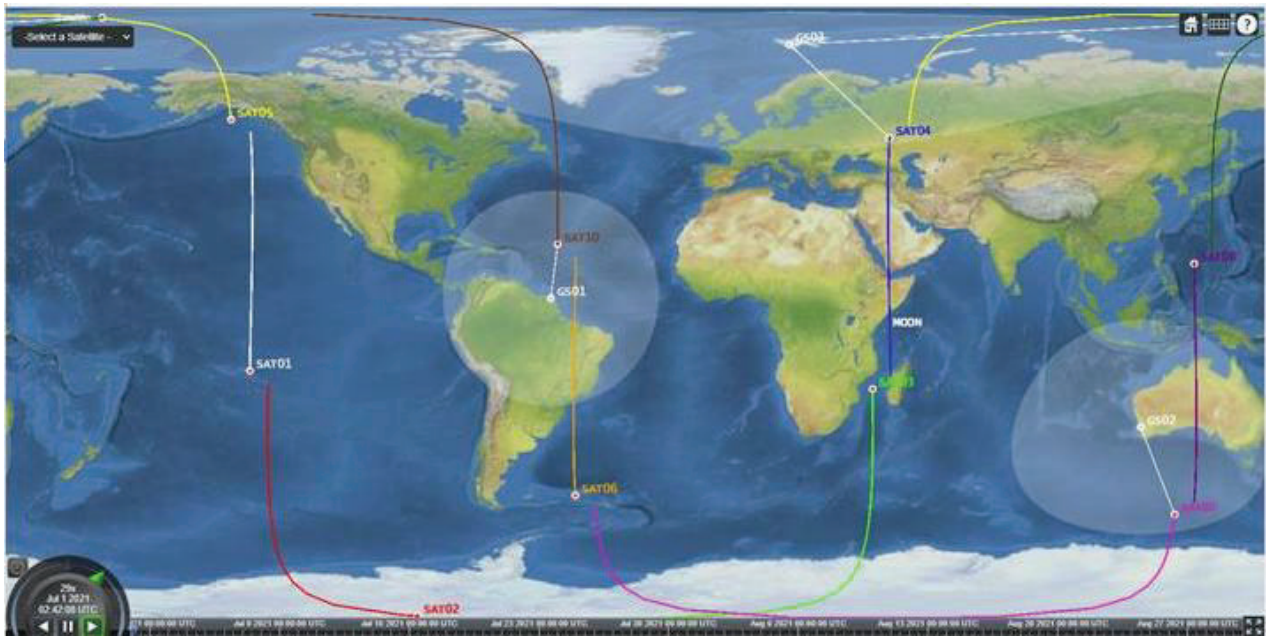


Figure 7-3: Visualfocus 2D Display



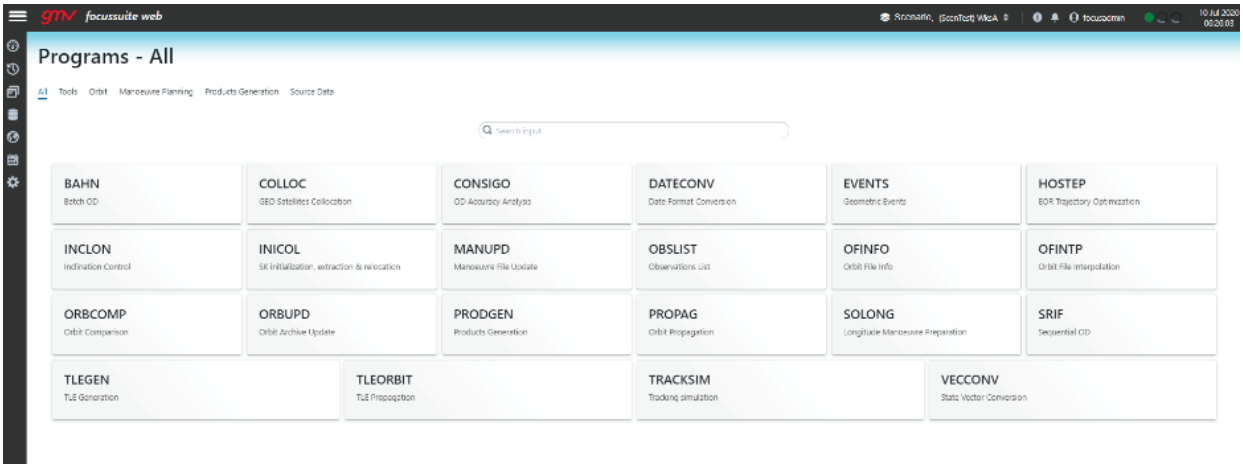
For both 2D and 3D, the Cesium open-source visualization library is used, licensed under the permissive license Apache 2.0. Cesium is a JavaScript library that can be used for creating 3D globes and 2D maps in a web browser, without a plugin. It is a cross-platform, cross-browser library, capable to visualize dynamic-data.

Cesium is built on several new HTML5 technologies, the most important of which is WebGL, used for hardware-accelerated graphics. Even though these new standards are quickly becoming widely adopted, some browsers and systems need to be updated to support them. To run Cesium applications, a local web server to host the files of the applications must be installed (e.g., Node.js). Cesium provides a rich set of features for displaying and interacting with geographic data and representing a wide range of geometric objects, as well as many plugins that add functionality.

8. COMPUTATION LAYER

The evolution of the system over the last years has provided important enhancements not only in the infrastructure part but also in the computational aspects. In this new generation GMV has achieved the harmonization of the computational layer. Therefore, ***Focus*** from this point on is considered a unique generic product able to deal with the different orbit regimes (GEO, LEO, LEOP, MEO, HEO, CN).

Figure 8-1: Computational modules (Example sharing GEO and LEOP modules)



The computational layer of ***Focus*** implements all the Flight Dynamics functions, providing the different services required by the FDS. As explained above, it is divided in different modular components (or programs), all of them implemented using common libraries, that communicate with each other through internal files.

Focus provides many mission-independent modules supporting all kind of regimes for the following activities/operations:

- **Orbit determination:** reconstruction based on external observations, compatible with all kinds of station tracking or pointing and inter-satellite observations, especially for GNSS data, or post-processed position solutions. The perturbation models used in the determination are very accurate and can be used for POD applications when accurate input data is available. Orbit covariance evolution, required for probability of collision computation, is also an output from this component.
- **Orbit propagation:** orbit propagation is closely related to orbit determination and operationally seldom used as an independent module, since its output can be directly generated from the determination and/or the maneuver optimization functions. However, it can be used for dedicated covariance analysis or for assessment of the maneuver effects during a collision avoidance maneuver computation.
- **Attitude prediction**: The nominal attitude is computed based on pre-defined laws.
- **Orbital event prediction:** generic event prediction function, based on the computation of zeroes and extremes of the geometrical functions defining each specific event. This component provides the capability of computing orbital events (such as node crossings, orbital apogees, eclipses, Sun zenith angles, Earth zone overflight…), station related events (AOS/LOS, mask occultation, collinearity with the Sun or other satellites that may generate interferences…) and sensor/instrument/transponder events (Sun/Moon blinding, visibility of points over earth, visibility of areas of interest, inter-satellite links…).
- **Orbit maintenance and maneuver planning**: Depending on the orbital requirements and the satellite propulsive system different components can be deployed in ***Focus*** for the station keeping and maneuver planning management. For GEO satellites with chemical propulsion the station keeping manoeuvres are computed by INCLON and SOLONG while the relocation manoeuvres are generated by INICOL.

- **Data ingestion and Product generation**: ***Focussuite*** provides support to several standard interfaces (generally CCSDS, but eventually also others, as antenna CORTEX), both for input and output data. It requires some external data (Earth Orientation Parameters, space weather…) that can be taken from public sources. Other standards are managed for both input and output data, such as the ingestion and generation of TLEs, generation of STDMs and other standards.
- **Other tools:** ***Focus*** provides some additional tools for analysis and support functionality, such as frame and time scale conversion, tracking simulation, and other features that would be useful for a flight dynamics operator.

Additionally, a set of mission-dependent modules will be developed to support a given mission:

- **Maneuver Implementation**: The conversion to a proper maneuver command and the computation of the whole acceleration profile and predicted mass consumption is heavily dependent on the propulsive system, so a dedicated module should be developed based on the S/C algorithms to be configured by the operator
- **Post-maneuver analysis:** Whereas maneuver calibration is performed during the orbit determination process, the reconstruction of the acceleration profile based on the satellite telemetry permits the computation of the mass (and, eventually, all mass properties). In case the propulsive model has internal parameters to be calibrated, the combination of the maneuver estimation and its orbital effect with the reconstruction permits the estimation of such parameters in the model.
- **Interfaces:** While generic interfaces are available in ***Focussuite*** (CCSDS standards and integration with other GMV Ground Segment products) a set of interfaces specific for the mission may be implemented if required.

Specific developments to customize the system for each mission are expected to be performed for each mission, so the final solution is provided as a turnkey element to be directly operated without requiring any additional development by the customer.

8.1. ORBIT DETERMINATION AND PREDICTION

The orbit determination in ***Focus*** is performed within BAHN component. This program implements a batch least squares method to estimate the orbital state vector and/or different parameters defining the dynamical model or related to the observations being used in the process.

The determination is performed through successive propagations of the estimated state vector at a certain epoch and its refinement in successive iterations. The orbit propagation is performed through PROPAG (also available as an independent component), which integrates the perturbation forces computed with very accurate models:

- Non-central gravity field: EIGEN-GL04C geopotential model set up by default, but can be configured
- Point-mass third body gravity (Sun, Moon and planets) and J2/Moon interaction
- Solid and ocean tides
- Relativistic gravitation
- Aerodynamic forces. Drag, lift and lateral components supported. Constant or variable areas, including boxwing model with the modelling of different satellite forces accounting for the attitude. Different scale factors ("step" or "linear") possible for one propagation arc.
- Solar radiation pressure: IERS formulation. Constant or variable areas including boxwing model.
- Radiation pressure (albedo, infrared, S/C thermal thrust)
- Maneuvers: can be considered as impulsive delta-V or long maneuvers with variable acceleration profile (in modulus and direction).
- Empirical accelerations (CPR in velocity frame and CODE in Sun-pointing frame).

Whereas these models are fixed, they should cover all the external perturbations having an impact on the orbit propagation. In fact, these models are used for POD services, achieving sub-centimeter accuracy.

The corrections to be applied to the parameters being propagated in every iteration are computed, minimizing the residuals of the observations with respect to their expected values. The nominal observations are calculated with TRACKSIM (also available as an independent component). The following type of observations are currently supported:

- Ground station (or second satellite) to satellite one/two-way range
- Ground station to satellite to second station and back four-way range
- Ground station (or second satellite) to satellite one (or two) way range rate (doppler)
- Ground station (or second satellite) to satellite one (or two) way range difference
- Ground station pointing angles (azimuth and elevation)
- Satellite to satellite phase
- Satellite positions (inertial or Earth-fixed)
- GNSS undifferenced carrier phase
- GNSS undifferenced pseudo-range
- GNSS double differences in phase between two orbiting receivers
- GNSS double differences in pseudo range between two orbiting receivers

High accuracy models are also used to implement different corrections to the observations (e.g., ionospheric, tropospheric, relativistic, solid tides corrections…).

With this process, BAHN can estimate the following parameters:

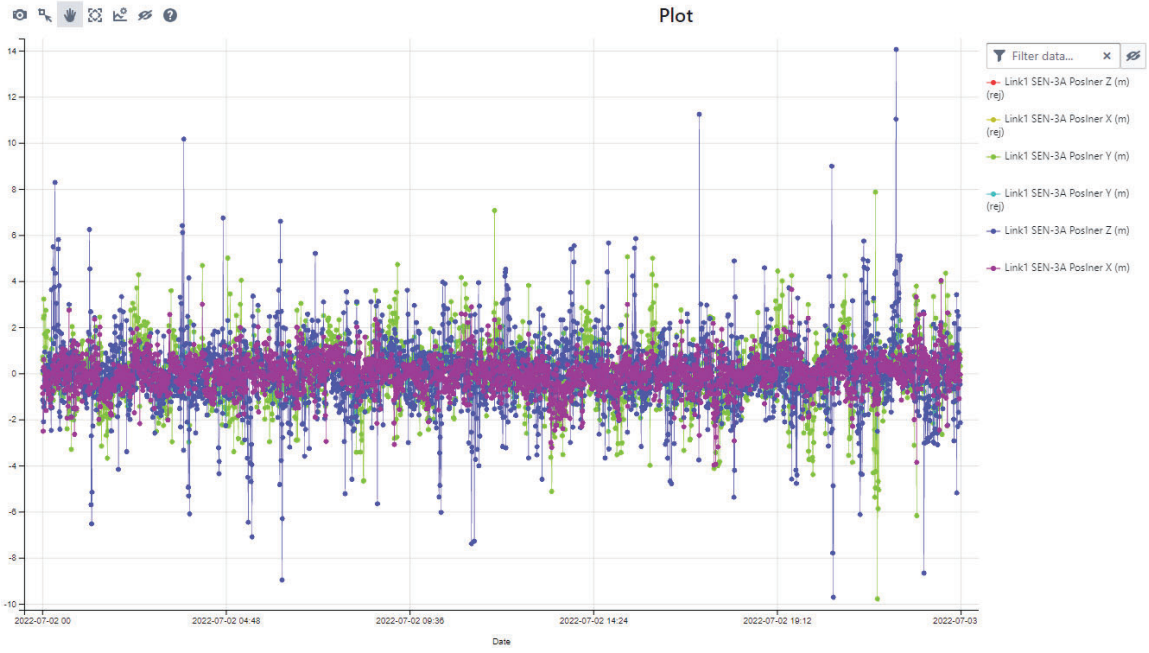
- Spacecraft Parameters:
  - Scale factors for maneuvers
  - Aerodynamic force coefficients
  - Radiation pressure coefficients
  - Errors in the execution of maneuvers
  - Clock bias on transponders
- Station Parameters:
  - Station biases per arc and per pass
  - Clock bias on tracking instruments
  - Tracking station positions
  - Central Body orientation
- GNSS Parameters:
  - Clock bias
  - GNSS ambiguity

Notice that the residuals can be monitored by the system to assess the evolution of the accuracy of the ground station data, determining the need of calibrating the ground sensors. If the mission counts with any additional observations (typically for a LEO mission, the on-board navigation solution of the GNSS receiver is accurate enough for this purpose), the system can calibrate the ground stations without needing any external data. If the ground station data is the only source of orbit determination observations, some external data would be required for this purpose in order to have the station parameters being estimated once fixed the satellite orbit.

Also, a natural result of the orbit determination process, the orbit covariance is generated, which can be used to further refine the probability of collision in case of close approach. Both orbit and covariance data can be used to generate orbit products to be exported to other subsystems.

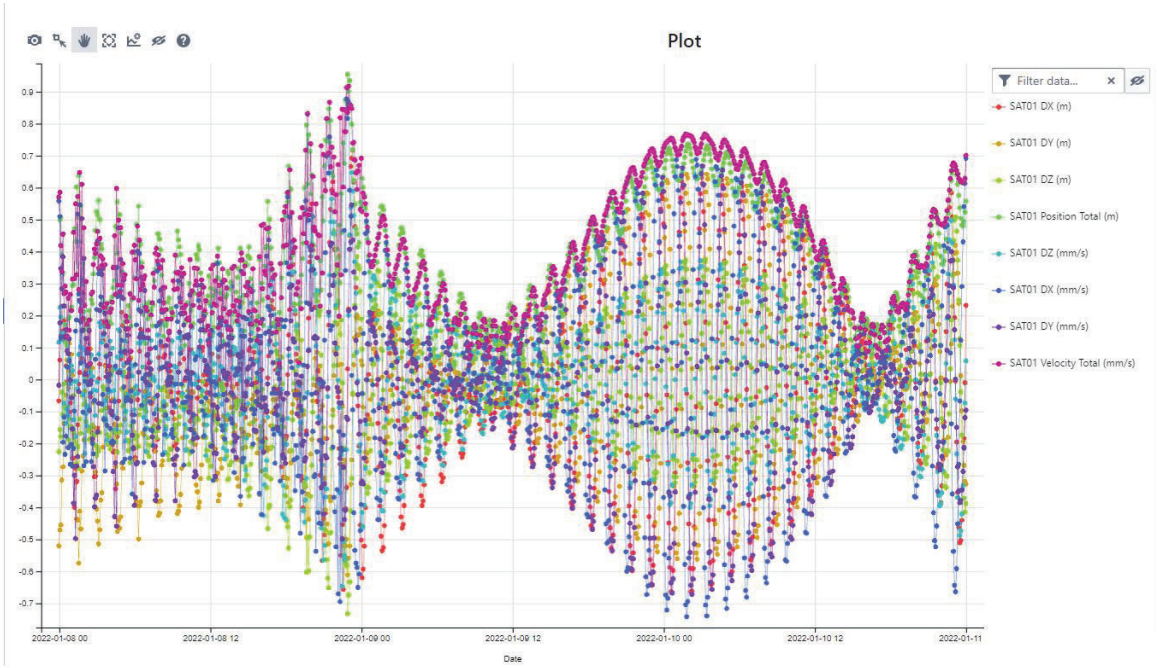
The whole system, but in particular the orbit prediction and determination components, are completely multi-satellite. Therefore, it is possible to perform the orbit determination of the customer satellites, even if the surface and mass properties of those objects are unknown (since BAHN can estimate the CpS/m coefficient instead of only the parameter).

Figure 8-2: Orbit determination residuals



During the commissioning phases the orbit determination is critical for the initial satellite acquisition. The injection state vector is propagated, and the first contacts with the ground stations are computed. Typically, the time offset value (TOV) reported by the ground stations must be considered for the orbit determination, since an along-track error (or time difference in the injection state vector) is very likely to be received from the launcher.

Figure 8-3: Orbit comparison plot



Together with the previous components, some additional basic tools are provided for analysis and robustness of the satellite operations, such as ephemeris comparison or merging, permitting isolating any possible issue in the orbit determination process leading to a wrong solution.

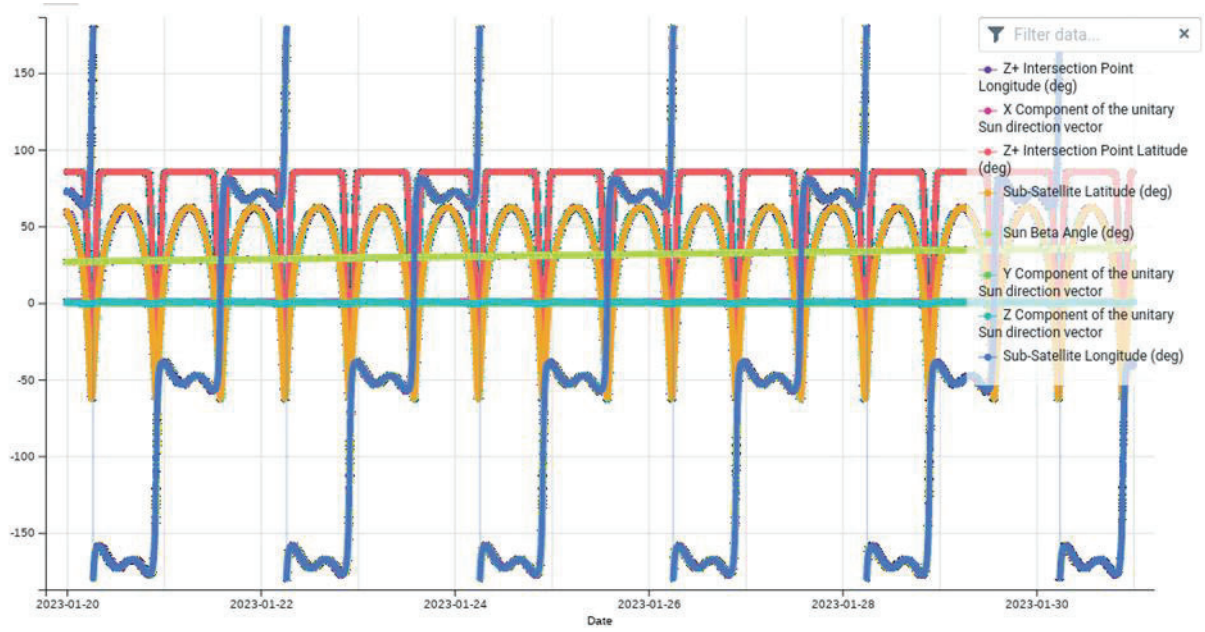
Focus also provides components for semi-analytical propagation (SAPO), permitting a fast epoch-to-epoch propagation of the perturbation forces. Similarly, some analytical models are available, in particular SGP4 in order to ingest and / or generate TLEs.

8.2. ATTITUDE PREDICTION

Attitude management in ***Focus*** differs from the usage of the orbital data. Since the AOCS system on board introduces control torques based on the feedback received by the attitude sensors in a closed loop, it is not possible to predict the internal torques, and hence, integrating the perturbations in the same manner than the orbit is propagated becomes impossible. Therefore, it is assumed that the on-board attitude control system can control the pointing with enough accuracy and the prediction is based on the commanded attitude.

ATTSIM (Attitude Simulation) is the component in ***Focus*** implementing the nominal guidance modes to predict the satellite attitude. Nominal modes (Sun-pointing, Nadir-Pointing, …) may be extended using biases.

Figure 8-4: Target pointing attitude in a HEO.



During GEO phase the attitude management used to be quite easier and there is no need to use specific modules to predict it. Usually just a bias with respect the Local Orbital reference frame is defined, and the maneuver computation programs are cable of taking it into account.

8.3. EVENTS PREDICTION

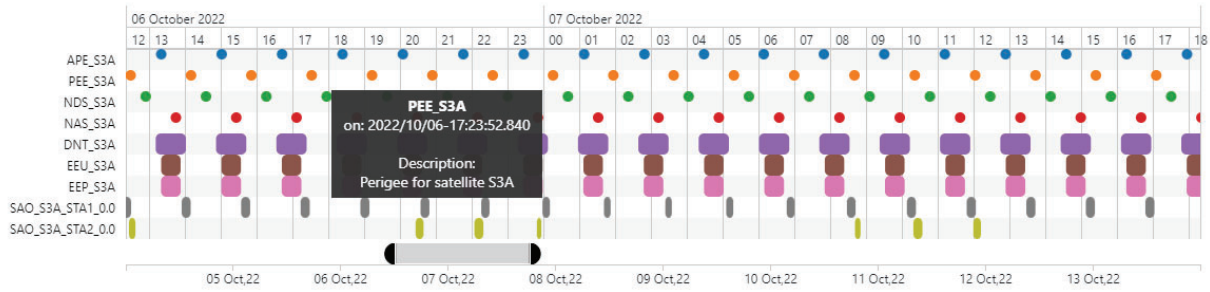
EVENTS is the component in ***Focus*** in charge of the geometric event prediction. Using the S/C attitude and ephemeris, it implements different mathematical functions characterizing the different events, which are computed by zeroing, maximizing or minimizing the corresponding functions.

The following events may be computed by this module:

- Apogee and Perigee crossing times
- Ascending and descending node crossing times
- Day night terminator crossing
- Umbra and penumbra crossing times of Earth
- Moon eclipses (including their maximum depth)
- Sun-Satellite-Earth Collinearity
- Latitude and longitude crossing
- Satellite local time
- Earth zone crossing
- Maneuvers
- Sun angles (zenith, elevation and azimuth)
- Dawn and dusk events (defined by Sun elevation)
- Phase angle
- AOS/LOS 0 and X degree
- AOS/LOS horizon mask

- Sun-Satellite-Station Collinearity
- Satellite-Station-Satellite Collinearity
- Antenna keyhole
- Switchover
- Satellites visibilities
- Station visibilities
- Transponder Activation and Visibilities
- Sun, Moon, planets, stars, other satellites, stations and points on Earth AOS/LOS for sensors

Figure 8-5: Output Events shown as a Gantt Chart



Notice that the event computation is used as a basis for different features required for the system, particularly command generation, where there may be a restriction for the execution of on-board activities or when interaction with external elements is required (e.g. Ka-band antenna pointing).

8.4. PRODUCT GENERATION

As in the previous section, ***Focus*** outputs can be adapted to any required format and interface. The system supports CCSDS standards, but the output data can be adapted to other formats and/or data streams. JSON files are typically sent through dedicated API, but other interfaces have been implemented depending on the mission (e.g. publishing the updated data in a Kafka topic). For file dissemination a dedicated FDSEXPORT component is provided.

Most of the FDS products are generated in the components described in the previous sections. However, depending on the platform, additional commands may be required (e.g. on-board propagator update, manual wheel off-loadings, sensor blinding commands…). Dedicated modules may be needed to cope with such computations, although their outputs will be typically sent to RTS.

As above, the following non-exhaustive list of output products is expected to be generated for a LEO mission:

- Command parameters for RTS. This includes maneuver parameters to be commanded for orbital manoeuvres.
- Orbit information for ground-stations: Typically, TLEs are provided to the ground stations, but other formats can be generated (e.g. Intelsat 11 parameters for GEO, OEMs…). Contact times and expected antenna pointing can also be generated (and formatted in TDM or STDM standard formats).
- Orbital events for mission planning / reservation tool: including GS contacts, eclipses and any other event computed internally. CCSDS does not yet provide a standard for this kind of information (a pink book is available but has not yet been completely defined or formalized) so the format for this information has to be agreed.

- Satellite ephemeris: OEM format is preferred, which also permits including the covariance for the CA service provider. Customer satellite information can be similarly transferred if the predicted orbit is needed by the customer.
- Satellite attitude: as above, AEM format is preferred.
- OPM including maneuver information can be also generated for external entities in case the high-level maneuver information is needed elsewhere.

8.5. DATA INGESTION

Focus is fully integrated with GMV's control center (including ***Hifly*** as the real-time system, ***Flexplan*** as mission planning system and additional systems for automation, reporting…). However, it has been also integrated within other ground segments (using, for instance, Kratos RTS).

The system can be easily adapted to cope with any kind of interface. Input files have been classically used as interface between the different systems in the ground segment, but this tendency is being replaced by modern interfaces such as REST APIs or message queues.

The FDS provides dedicated components to transfer files, such as FDSIMPORT. ***Focus*** makes use of some data publicly provided on the internet, such as NOAA's RSGA (report of solar and geomagnetic activity), IERS Earth orientation parameters and leap seconds. In case the FDS network is isolated from the public internet, a dedicated instance of FDSIMPORT can be deployed in a DMZ to download the data and transfer it to the server once the public access is restricted.

Expected inputs for the system are:

- Telemetry: processed engineering values from RTS. Interface can be adapted to any requirement as described above. The expected contents of the telemetry may vary depending on the platform and the available data, but will typically include:
  - GNSS data (raw data or navigation solution) to be used in the orbit determination process.
  - Propulsive system data: depending on the mission, on-times, tank pressure and temperature, number of activations… telemetry needed to reconstruct the maneuvers on-ground and estimate the mass consumption.
- Tracking data: ground station observations for the orbit determination process. Pointing angles, ranging, doppler….
- GS ancillary data: pressure and temperature at ground station location, if available, to properly compute the tropospheric correction. Can be provided in a CCSDS TDM. Averaged values can be used if not available.
- Customer satellite ephemeris: to compute possible contact times. OEMs are preferred, but other formats can be adapted.
- Auxiliary data: solar and geomagnetic activity (for ionospheric correction and, potentially, atmospheric model used for determination and propagation of the LEO customer satellites), EOPs, leap seconds…
- Satellite database: ***SPBImport*** tool can open a SPB (Satellite/Spacecraft Parameters Book) file and import the required data directly into the ***Focussuite*** database. The tool is designed to support different SPB formats and it is highly configurable by means of an import script associated to each scenario or workspace and named the SPB import profile.

Notice that the previous list does not intend to be exhaustive, since it has many dependencies with the mission interfaces and the satellite platform.

8.6. ELECTRIC ORBIT RAISING SUPPORT

To support fully low thrust orbit raising operations from the flight dynamics point of view, the FDS must be provided with the ability to:

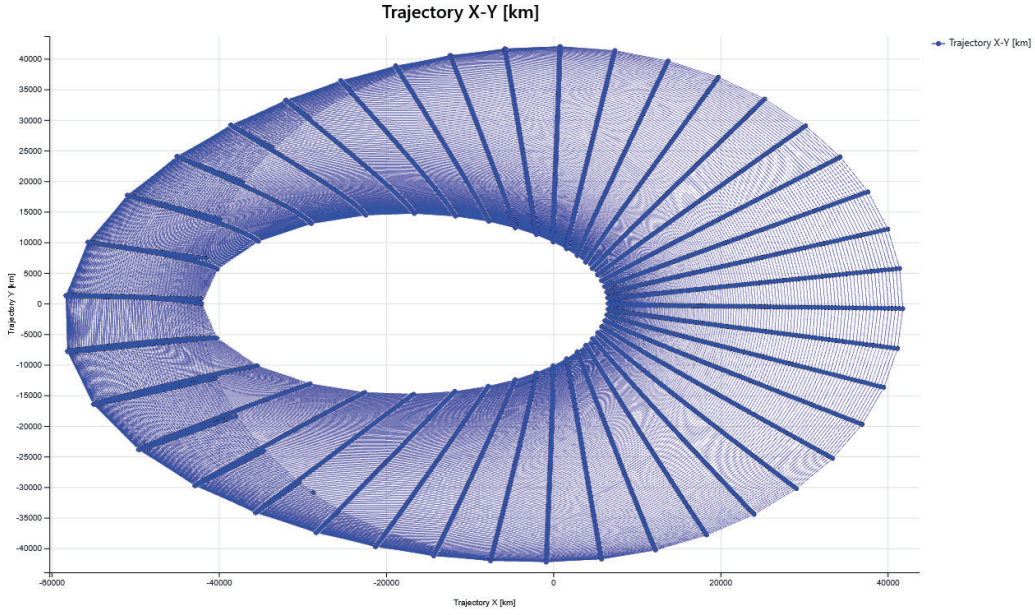
- define a low thrust orbit raising strategy to reach a target operational orbit;
- implement and command the maneuvers needed to follow the defined strategy;
- perform orbit determination and thrust estimation to monitor the evolution of the trajectory and compare it with the predicted strategy;
- periodically refine the strategy from the latest orbit determination information and assure that the final target is finally achieved.

FDS must therefore be enhanced to include the tasks related with the design and computation of such a strategy. ***Focussuite*** provides a platform independent set of computational programs is proposed as the first step of the problem. Later, the solution can be post-processed to account for additional platform-dependent constraint in dedicated programs.

Starting by the orbit raising strategy design, defined by a thrust profile (which can consider nominal, secondary or no thrust, depending on eclipse conditions), the FDS contains the following tools:

- **Initial Guess Electric Orbit Raising (ELPRIMO)**: to compute an orbit raising strategy that reaches the target with a simplified propagation model and considering different thrust strategies. It provides an initial guess of the strategy, including its flight time, to be used as input in ELPOTRO.
- **Electric Propulsion Transfer Optimization Software (ELPOTRO)**: to optimize the strategy initial guess in terms of flight time, considering a more realistic propagation model and a bigger set of constraints.

Figure 8-6: Example of plots generated by ELPOTRO.



8.7. PLATFORM DEPENDENT FUNCTIONS

While specific implementations may vary from one platform to another, ***Focussuite*** provides customised modules for these key functions in operations:

CALIB: Manoeuvre Implementation

CALIB is used to analytically predict the thrusters activity that the satellite on-board system will initiate for a given manoeuvre midpoint time and direction. CALIB models the performance and fuel flow rate of the thrusters based on telemetered measurements and mathematical expressions provided by the manufacturer.

MASSEVO: Mass Book-Keeping

MASSEVO is used to compute the propellant mass consumed by satellite on-board and to maintain the fuel book-keeping. MASSEVO will normally be run immediately following a manoeuvre to calculate the amount of fuel mass consumed so that a current estimate of the overall spacecraft mass and mass distribution may be maintained. Given a starting time and a finite interval, MASSEVO uses telemetered burn time totals for the thrusters in combination with the same fuel flow rate model used in CALIB to determine the total fuel consumption over the interval and updates the stored spacecraft mass estimate to reflect the loss of fuel mass. It also uses the telemetered burn times to generate an estimate of the actual delta-v achieved during the manoeuvre.

OBPU: On Board Propagator Update

This module is in charge of computing the required information for computing the inputs used by the on board propagator.

COMMANDO: $\Omega\ \omega t_f m_f$

This module is used to generate the specific command inputs that are needed for operational procedures. Typically, it will include maneuver parameters, on-board propagator updates and other specific inputs, depending on satellite platform.

8.8. LEO AND CONSTELLATION STATION KEEPING

Focus provides the necessary programs to perform station-keeping for LEO and and constellation missions.

Ground track control: TARGETDEF, ORBANA, EOCONTROL

These modules allow the operator to keep a ground-track and local time orbit control. This control is mainly used for Sun-synchronous orbits, typically dedicated to Earth observation missions.

TARGETDEF: the purpose of this program is to generate the reference orbit, ground-track and Nodal Sequence file which will fulfil the user constraints on the definition of the orbit.

ORBANA: The main purpose of this program is to compute the ground-track and nodal separation of the satellite with respect to the reference.

EOCONTROL: The main purpose of this program is to compute the manoeuvres to maintain the current orbit within the limits defined by the user on the reference ground track, which will guarantee that the spacecraft does not leave the orbital tube define for the mission; the control strategy will then be based on a ground track control in all latitudes, with a configurable dead-band, and with the aim of minimizing the number of manoeuvres.

SVTARGET: target acquisition.

The Target Acquisition component computes manoeuvres to position the satellite.

MANCO: manoeuvre computation

MANCO is in charge of station keeping manoeuvres computation. It is responsible for computing the needed ΔV and epoch of the manoeuvres to keep the satellite inside several user-defined windows for all the relevant orbital parameters (SMA, eccentricity, inclination…). It is able to handle windows for different in-plane and out-of-plane parameters in the same execution, optimizing the computed manoeuvres to reach the desired targets.

ORBIN: orbit insertion

ORBIN is in charge of insert the satellite into its slot for starting the station keeping manoeuvres computation. It can be also used for de-orbiting. It is responsible for computing the needed ΔV and epoch of the manoeuvres to insert the satellite inside several user-defined windows for all the relevant orbital parameters (SMA, eccentricity, inclination…). It is able to handle windows for different in-plane and out-of-plane parameters in the same execution, optimizing the computed manoeuvres to reach the desired targets.

RELOC: relocation

RELOC is in charge of relocate the satellite from one slot to another one inside the same plane. It is responsible for computing the needed ΔV and epoch of the manoeuvres to perform an in-plane relocation.

8.9. GEO STATION KEEPING

Focus also provides the necessary programs to perform the station-keeping in a GEO slot, the relocation from one slot to another and the initialization of a collocation strategy with other GEO satellites.

SOLONG: East-West Station-Keeping Maneuver Planning

SOLONG calculates one, two, or three burn longitude maneuvers for longitude station keeping with eccentricity control. SOLONG uses the ephemeris file and calculates the execution time and size of the tangential burn according to the options specified in the input file. If multiple maneuvers are planned, then the first and second maneuvers will be separated by odd multiples of 12 hours. SOLONG includes the capability of supporting longitude maneuver planning for collocated satellites.

Two station keeping strategies are available: Sun-pointing perigee and Minimum eccentricity, and follow the control requirements (longitude box, eccentricity control circle), which are defined in the Station Keeping Database. Additional strategies can be implemented. End of life maneuver can also be computed by SOLONG for the final de-orbiting maneuvers to move the satellite to the "graveyard" orbit.

INCLON: Inclination Station-Keeping Maneuver Planning

INCLON calculates a North or South maneuver thrust for inclination station keeping. INCLON uses the ephemeris file and calculates the execution time and size of the North/South thrust according to the options specified by the user, including the capability of supporting inclination maneuver planning for collocated satellites. This capability is accomplished by allowing the user to define the center of an offset inclination control circle (IX, IY) to be used in the different North/South station keeping strategies supported by INCLON.

There are currently three modes supported for computing the North/South maneuver: Target Pole, Pole Motion Direction, and Inclination Symmetry. Additional strategies can be implemented.

IDEC: Inclination and Eccentricity control

IDEC computes the N/S continuous maneuvers required to control the S/C inclination and eccentricity (if required). The main objective of IDEC is to correct the inclination secular drift to keep the satellite latitude inside the control box. Therefore, based on the user inputs, it computes the required planned maneuvers (delta-V and midpoint epochs) to be calibrated afterwards by a maneuver calibration program (ISKM). The tool can also compute E/W maneuvers to control the S/C longitude.

Note that this module is only provided in case of ionic propulsion to satisfy a simultaneous inclination and eccentricity control.

COLLOC: Collocation Monitoring

COLLOC is the collocation strategy and proximity monitoring and maneuver sensitivity assessment function. The process performs a proximity check for all satellites within a collocation group for the purposes of monitoring the inter-satellite separations in terms of distance and angles and for the purpose of warning against the risk of too close an approach that might cause radio signal interference or a collision.

COLLOC reads the appropriate orbit files and prints the inter-satellite separations for all combinations of pairs of satellites with a warning message if they become too close (when distance falls below specified threshold values). Both text and graphical output is produced including:

- Inter-satellite distance for all combinations of pairs of satellites
- Inter-satellite angular separations for all combinations of pairs of satellites

In addition, the process can assess the impact on the inter-satellite separation (distance and angle) of a planned maneuver, either under-performing and/or over-performing by factors specified by the user.

In case the minimum distance threshold is violated a collision avoidance maneuver will be computed in the following way:

- Eccentricity maneuver will be applied to accomplish the condition that the delta-e – delta-i angle is 0. This is done in such a way that the East/West maneuver will achieve an eccentricity target that will be the minimum that avoid a collision. Delta-e and delta-i values are obtained by the differences between the centers of eccentricity and inclination control circles.
- The longitude drift will be kept within bounds by proposing double maneuvers with exactly the same delta-V in opposite directions.
- Once the collision avoidance maneuver has been applied internally, a check if the satellite is kept in the longitude control window is performed. The number of days for longitude window control is taken from the COLLOC main input file. If the maneuvers cause the satellite to drift out of the window, then COLLOC will modify the longitude drift by re-planning the collision avoidance maneuvers, maintaining the eccentricity target.

INICOL: Collocation Initialization and Station Acquisition Support

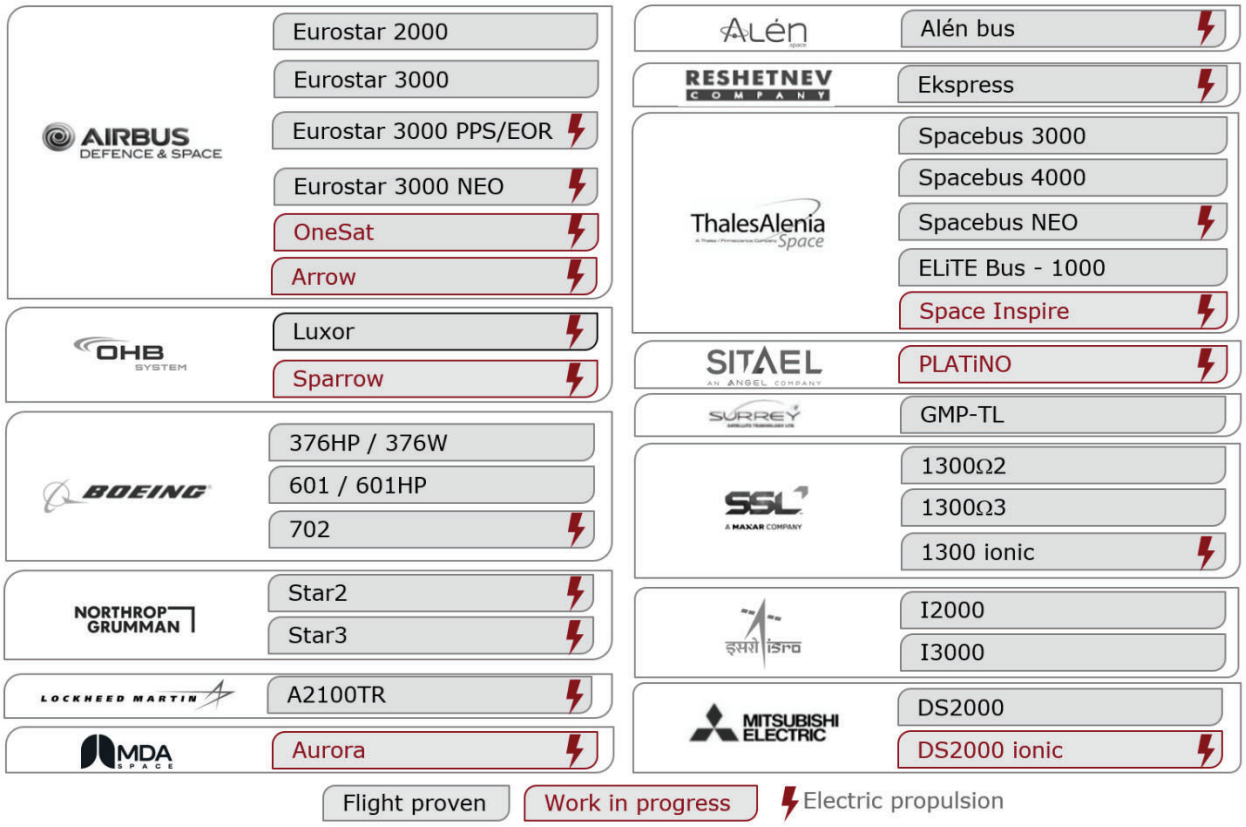
INICOL is the collocation strategy initialization function of ***Focus***. Three different scenarios are supported:

- New satellite addition to an existing collocation group.
- Satellite removal from an existing collocation group (relocation or deorbiting)
- Satellite returns to the group from a non-nominal situation.

The INICOL program calculates the sequence of maneuvers to insert a satellite in its on-station window, collocated with other satellites or alone. INICOL computes the maneuvers to reach the on-station window, avoiding other satellite windows in the path. INICOL also initializes the first cycle of station keeping maneuver (East/West and North/South). Finally, there is an option to update the operational maneuver file with the maneuvers proposed by INICOL.

9. FOCUSSUITE SUPPORTED PLATFORMS

Table 9–1: *Focussuite* supported platforms (March 2026)



END OF DOCUMENT