esoc

|  |  |
| --- | --- |
| label_document |  |

|  |
| --- |
| NMF Mission: OPS-SAT - Software Design Document |



|  |  |  |  |
| --- | --- | --- | --- |
| Title NMF Mission: OPS-SAT – Software Design Document | | | |
| Issue 1 | | Revision 1.0 | |
| Author César Coelho | | Date 23/11/2017 | |
| Approved by | | Date | |
|  |  |  |



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Reason for change | | Issue | | Revision | | **Date** | |
|  |  |  |  |  |  |  |  |



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Issue | | Revision | | | | | |
| **Reason for change** | | **Date** | | **Pages** | | **Paragraph(s)** | |
|  |  |  |  |  |  |  |  |

Contents

[2. Introduction 4](#_Toc499542346)

[3. NMF Mission: OPS-SAT 5](#_Toc499542347)

[3.1. OPS-SAT System Deployment 6](#_Toc499542348)

[3.1.1. Experimental Platform 7](#_Toc499542349)

[3.1.2. Mission Control System 8](#_Toc499542350)

[3.1.3. Data Relay Server 8](#_Toc499542351)

[3.1.4. Experimenter Remote System 9](#_Toc499542352)

[3.2. OPS-SAT Scenario 10](#_Toc499542353)

[3.2.1. NanoSat MO Supervisor for OPS-SAT 11](#_Toc499542354)

[3.2.2. Ground MO Proxy for OPS-SAT 11](#_Toc499542355)

[3.2.3. EUD4MO 12](#_Toc499542356)

[3.2.4. External NMF Ground aplications 15](#_Toc499542357)

[3.3. Transport Binding 16](#_Toc499542358)

[3.4. Platform adapters integration with OPS-SAT 19](#_Toc499542359)

[3.4.1. Camera service adapter 19](#_Toc499542360)

[3.4.2. GPS service adapter 20](#_Toc499542361)

[3.4.3. Autonomous ADCS service adapter 21](#_Toc499542362)

[3.4.4. Other services’ adapters 22](#_Toc499542363)

[3.5. Customization of other services 23](#_Toc499542364)

# Introduction

The recent miniaturization of space components and electronics has allowed the design of smaller satellites which are considerably cheaper to build and launch than conventional satellites. This decrease in the total cost has boosted a new growing market for small satellites and, as the number of small satellites keeps increasing, there is a raising demand for reusable software across nanosatellites.

The NanoSat Mission Operations (MO) framework provides a standard on-board software framework that facilitates not only the monitoring and control of the nanosatellite, but also the interaction with its platforms and payload. This is achieved by using the MO services for Monitor and Control services included in the MO service suite and by defining a set of new Platform services, which also follow the MO services architecture.

The MO services are a set of standardized end-to-end services based on a service-oriented architecture which are currently being defined by the Consultative Committee for Space Data Systems (CCSDS) and they are intended to be used for mission operations of future space assets.

An NMF Mission is a concrete implementation of the NanoSat MO Framework for a specific mission. The NMF Mission development includes activities such as implementing the Platform services and the NanoSat MO Supervisor for the specific platform. If a custom or tailored transport is used for the mission, then the transport binding must be implemented and additionally, integrated with the Ground MO Proxy for protocol bridging.

An important part of an NMF Mission is to define the end-to-end scenario to describe how the system operates.

By reusing parts of an NMF implementation, it is possible to quickly develop a mission-specific service however an application connecting to this service would no longer be considered as an NMF App because it is no longer mission-agnostic.

In order to develop an NMF Mission, an NMF implementation is necessary.

**NMF Core Implementation**

**NMF Mission 1**

**NMF Mission 2**

**NMF Mission n**

**...**

**Figure 1:** NMF Missions developed on top of an NMF Implementation

The NMF Mission development is separate from the NMF Ground development and also from the NMF App development. This means that it is possible to have two different implementations of the NMF in different programming languages.

# NMF Mission: OPS-SAT

The “NMF Mission: OPS-SAT” is an implementation of the NanoSat MO Framework for ESA’s OPS-SAT mission. It uses a Java implementation with the Platform services using adapters connected to the APIs for the payload peripherals of OPS-SAT.

OPS-SAT is one of the first missions that intent to take advantage of the NanoSat MO Framework in order to manage its software on the Experimental Platform.

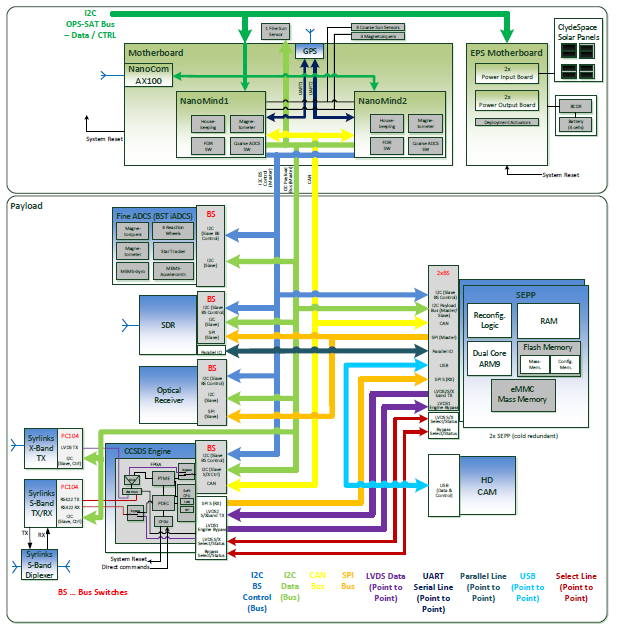
The next sections present the overall OPS-SAT system deployment, the scenario that was defined for OPS-SAT’s NMF Mission, the details of the selected transport binding for the communications to ground, the integration details of the Platform adapters with OPS-SAT’s payload peripherals, and finally, the implementation customizations that were done for other services.

The majority of the software development for the NMF Mission was done for the spacecraft side. On ground, only the Ground MO Proxy for OPS-SAT had to be developed as part of the NMF Mission.

The Ground MO Proxy allows other NMF Ground applications to connect to any NMF App as this is part of the portability concept of the NMF. On one hand it means that an NMF Ground application is able to connect to other future NMF Apps, and on the other hand it means that NMF Apps developed for OPS-SAT can be reused by future spacecraft that have an NMF Mission in place.

On the spacecraft side, OPS-SAT’s main bus is a Controller Area Network (CAN) bus which is capable of a 1Mbit/second bit rate and it is used for the exchange of messages between the CCSDS Engine, the Experimental Platform, and the Nanomind. This bus is represented in yellow in Figure 2.

It is also possible to visualize in Figure 2 the five payload devices in OPS-SAT: Camera, Optical Data Receiver, Software-defined Radio (SDR), FineADCS, and a GPS.

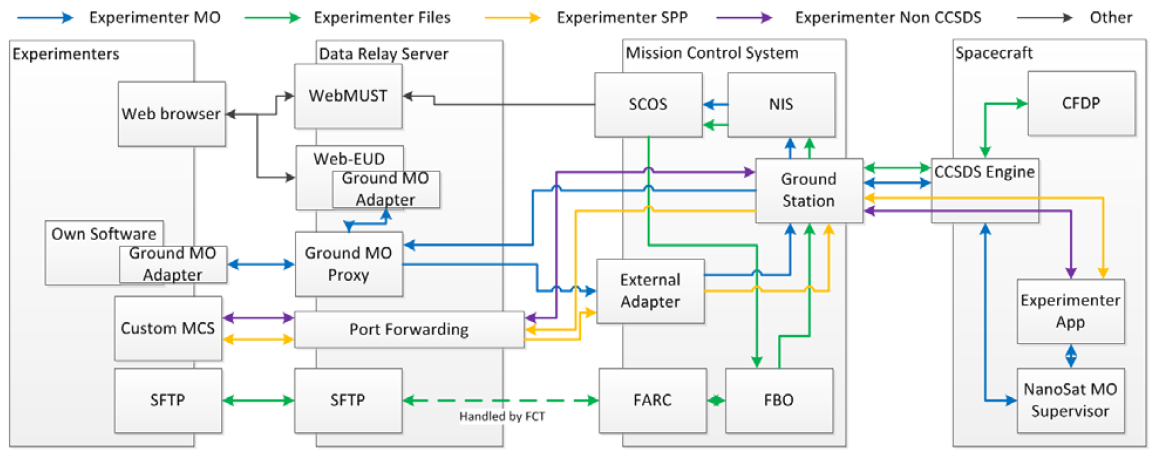


**Figure 2:** OPS-SAT spacecraft bus.

## OPS-SAT System Deployment

The OPS-SAT System Deployment is composed of 4 main parts. On the Experimental Platform there’s a mityARM device with 1 GB of RAM and an ARM processor with 900 MHz. On the OMCS and Data Relay Server, ESOC’s infrastructure is ready to support complex missions and includes many technical experts, a dedicated network, and its own virtual cloud. On the Remote experimenter side, the resources are an issue to be dealt by the experimenter.

The infrastructure itself is not so relevant because the resources are not so scarce therefore the focus of this section is on the applications running inside the nodes and not on the hardware.



**Figure 3:** System Deployment Overall view.

Figure 3 is an overall view of the system deployment however, it is a simplified view and does not represent all the details of the system. Some blocks were not represented in the Mission Control System node for two reasons. First, privacy concerns in ESOC’s Mission Control System and second, the details of the node are not very relevant for this document.

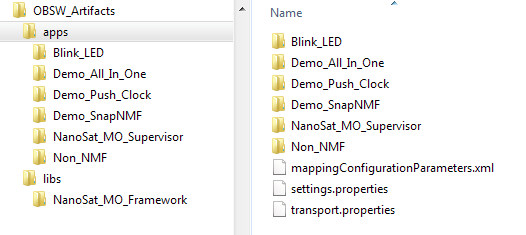
The next subsections present an overview of the software applications running on the different nodes.

### Experimental Platform

The Experimental Platform node is where the software experiments will run and they can be just simple applications, FPGA images, or complete OS images. A Standard OS image will be available containing default operational software: FileApp, and NanoSat MO Supervisor.

The FileApp application allows the transfer of files between ground and space using CFDP but this is out of scope for this document. The NanoSat MO Framework allows the management of the experiments and in case they are NMF Apps, to find them from ground.

The NMF implementation for OPS-SAT follows the Generic Deployment and includes a set of NMF Apps ready to be used for testing the software management mechanisms for the three different types of software experiment. An NMF Library for OPS-SAT was compiled and can be referenced by the NMF Apps running on-board.



**Figure 4:** NanoSat MO Framework deployment in OPS-SAT.

The folder’s structure presented in Figure 4 can be automatically generated from a maven project.

Some additional software was installed in order to run the NMF in OPS-SAT including the Java Runtime Environment version 8 for Linux for an ARM device, and the socketcand application which exposes a simple TCP/IP socket to exchange data with the CAN bus.

### Mission Control System

The Mission Control System node is composed by a set of applications that run on separate virtual machines. This includes NIS, OMCS, FARC, and others that are not represented for simplicity.

The Network Interface System (NIS) is the gateway between the Mission Control System and the Ground Station and provides means for TM/TC transfer via the CCSDS SLE specification as well as file transfers from ESA’s Telemetry and Telecommand System to the NIS.

The OPS-SAT Mission Control System (OMCS) is composed of 3 main chains, the File Transfer chain, the Monitor and Control chain, and the Experimenter’s chain. The File Transfer chain includes an application with an implementation of the CFDP standard, the Monitor and Control chain includes SCOS-2000 (Spacecraft Control and Operations System), and the Experimenter’s chain provides encoding and decoding of space packets for the applications running on the Data Relay Server.

No NMF-related application is running in the OPS-SAT Ground Data Systems node.

### Data Relay Server

The Data Relay Server node is composed by a set of applications that run on separate virtual machines. This includes EUD4MO, Ground MO Proxy, Port Forwarding, and SFTP.

EUD4MO and the Ground MO Proxy were developed using the NMF Composites and therefore they have dedicated subsections.

The Port Forwarding application allows forwarding data from/to the Experimenter’s chain of OPS-SAT’s Ground Data Systems to/from the Experimenter’s Remote System. The interface towards the Experimenter’s Remote System is TCP/IP and allows the exchange of space packets directly from/to the spacecraft. SFTP is also available in order to receive files that have been generated during the experiment’s execution in space.

Ideally, these applications should be instantiated for every new experimenter but this mechanism is not defined at the time of writing.

### Experimenter Remote System

The Experimenter Remote System node is in the experimenter’s complete control and therefore it can be anything. Based on the interfaces exposed by the Data Relay Server node, A few possibilities are envisaged: a web browser, an MCS with the Ground MO Adapter, a dedicated MCS for the experiment using space packets, or a SFTP application.

A web browser can be used to access EUD4MO for monitoring and control of an experiment. This is the least effort solution for an experimenter as most of the Operating Systems today come already bundled with a web browser. The only necessary requirement to meet is to develop the experiment in form of an NMF App.

An MCS with the Ground MO Adapter can be connected to the Ground MO Proxy which would provide protocol bridging, COM Archive mirroring, queuing of actions, and other functionalities.

A dedicated MCS for the experiment using space packets can be connected to the Port Forwarding application for directly sending/receiving space packets to/from the spacecraft. And finally, a SFTP client application can be used for exchanging files with the Data Relay Server.

## OPS-SAT Scenario

The OPS-SAT scenario is a complex scenario created by the combination of different Generic Scenarios presented in chapter 3. On the NanoSat segment, it follows the Multiple NMF Apps scenario, and on the Ground segment, it follows the Multiple Consumers scenario combined with the Proxy scenario.

In OPS-SAT, an NMF App can be seen as an “experiment”. Experimenters can develop their NMF App using the NMF SDK, create the respective NMF Package and then this will be tested, transferred and installed by ESOC. When the satellite is ready, the “experiment” is started.

Figure 5 represents the OPS-SAT scenario and it only goes down to the MAL binding layer therefore does not represent the components that operate based on data of the layers below. The green line between Ground MO Proxy and both EUD4MO and the External System, represent a communication using a MAL-TCP/IP binding. The blue line represents a MAL-SPP binding however the two different segments have different mechanisms of exchanging the space packets. The different mechanisms are further explained in section 3.3.

On ground, an “experiment” can be monitored and controlled either using EUD4MO or by connecting a NMF Ground application to the Ground MO Proxy. EUD4MO is a solution developed by ESA that allows a user to use a simple web browser to monitor and control an NMF App.

There is also a raw interface to exchange space packets directly with the spacecraft however it bypasses the Ground MO Proxy and therefore all the advantages of using a Ground MO Proxy are lost. Thus, it will not be present in this section but one can find it in section 3.1.

**EUD4MO**

**Ground MO Adapter**

**Experiment A**

**Experiment B**

**NanoSat MO Connector**

**NanoSat MO Connector**

**External System**

**Ground MO Adapter**

**NanoSat MO Supervisor: OPS-SAT**

**Ground MO Proxy: OPS-SAT**

**Figure 5:** OPS-SAT scenario.

### NanoSat MO Supervisor for OPS-SAT

The NanoSat MO Supervisor application acts as a supervisor to be deployed on the NanoSat segment of OPS-SAT. It extends the generic NanoSat MO Supervisor of the Java implementation and therefore most of the default behavior of the component is already present.

It includes the Platform services implementation with the additional adapters for the OPS-SAT platform devices.

The Platform services can be consumed from both the MAL-SPP transport binding and the MAL-TCP/IP transport binding. The former is intended to be used between the NanoSat and the Ground segments, while the latter is intended to be used by IPC between NMF Apps and the NanoSat MO Supervisor.

A dedicated adapter for the monitoring and control was implemented. The getters and setters for the Parameter service were implemented for a few set of parameters and the correct dispatch of actions for certain method calls was also implemented.

Three parameters were defined with the following names and respective descriptions:

* CurrentPartition: "The Current partition where the OS is running."
* LinuxVersion: "The version of the software."
* CANDataRate: "The data rate on the can bus."

Three actions were defined with the following names and respective descriptions:

* GPS\_Sentence: "Injects the NMEA sentence identifier into the CAN bus."
* Reboot\_MityArm: "Reboots the mityArm."
* Clock.setTimeUsingDeltaMilliseconds: "Sets the clock using a diff between the on-board time and the desired time."

### Ground MO Proxy for OPS-SAT

The Ground MO Proxy application acts as a proxy that is deployed on the Ground segment for ESA’s OPS-SAT mission. It extends the generic Ground MO Proxy of the Java implementation and therefore most of the default behavior of the component is already present.

It is capable of acting as a protocol bridge, as a COM Archive mirror. As a result, multiple consumers can share the same ground-to-space connection to the spacecraft, and connect to independent NMF Apps simultaneously.

The protocol bridge includes on one side the dedicated MAL-SPP transport binding developed for OPS-SAT and on the other side includes the MAL-TCP/IP transport binding.

The COM Archive mirror on ground has a single instance of the COM Archive implementation. If multiple NMF Apps are running simultaneously on the NanoSat segment, then they will synchronize with a single instance of the COM Archive on the Ground segment.

If one assumes that one single experiment runs on a certain timeslot, then the Ground MO Proxy application for OPS-SAT should be deployed as a new instance whenever a new experiment is started.

Two NMF Apps can run simultaneously however the COM Archive data for both of them is stored on the same instance on ground. This might generate privacy concerns for experimenters that want to have private data.

### EUD4MO

EUD4MO is a web-based lightweight Mission Control System (MCS) for monitoring and control remote MO providers. In implementation terms, is uses EUD for displaying the information in a GUI, and it uses the Ground MO Adapter in order connect and exchange data with an MO provider. Although EUD4MO is presented in this chapter, it is not limited to OPS-SAT and can be used by other future missions.

**Dummy App**

**NanoSat MO Connector**

**EUD4MO**

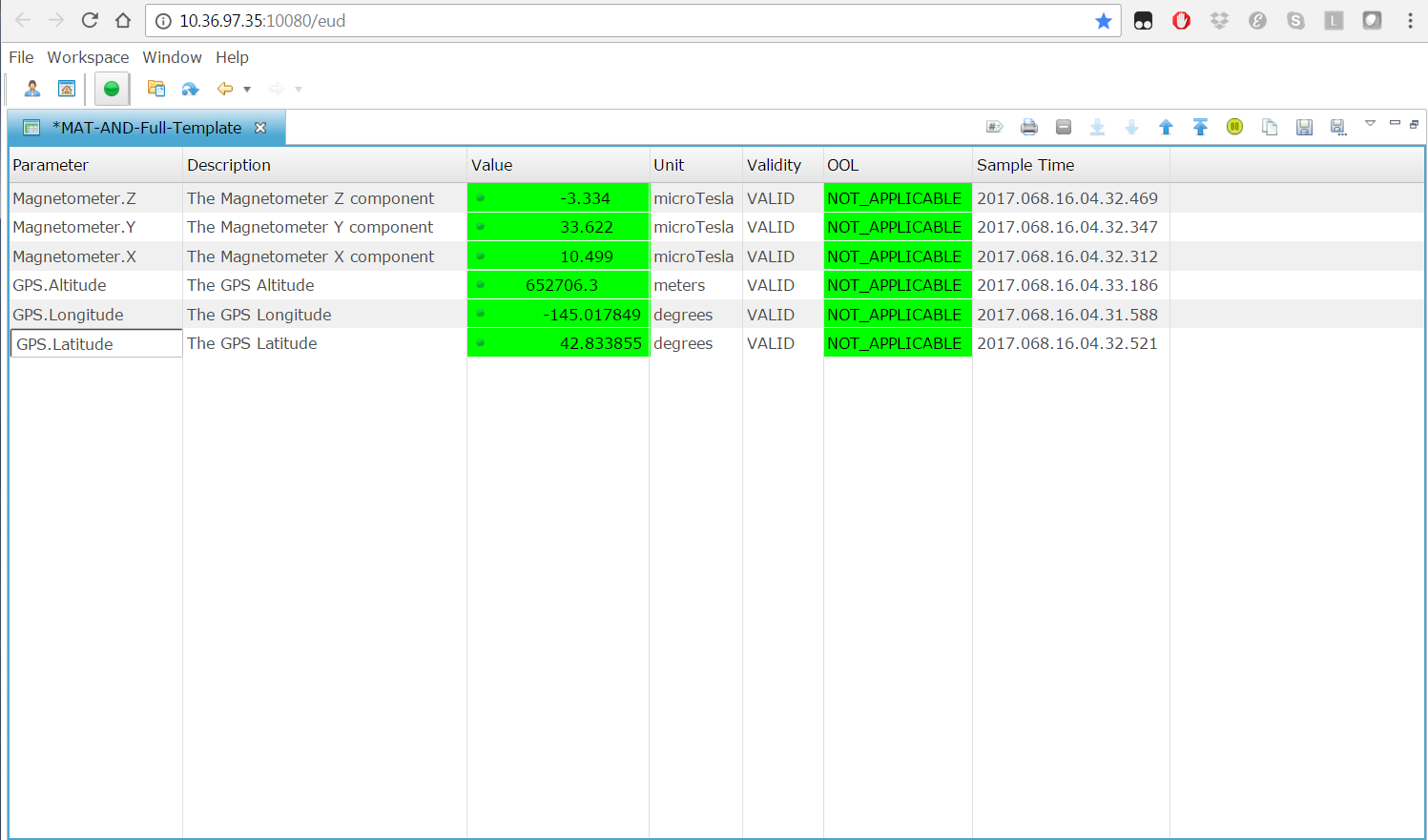
**Ground MO Adapter**

**Web Browser**



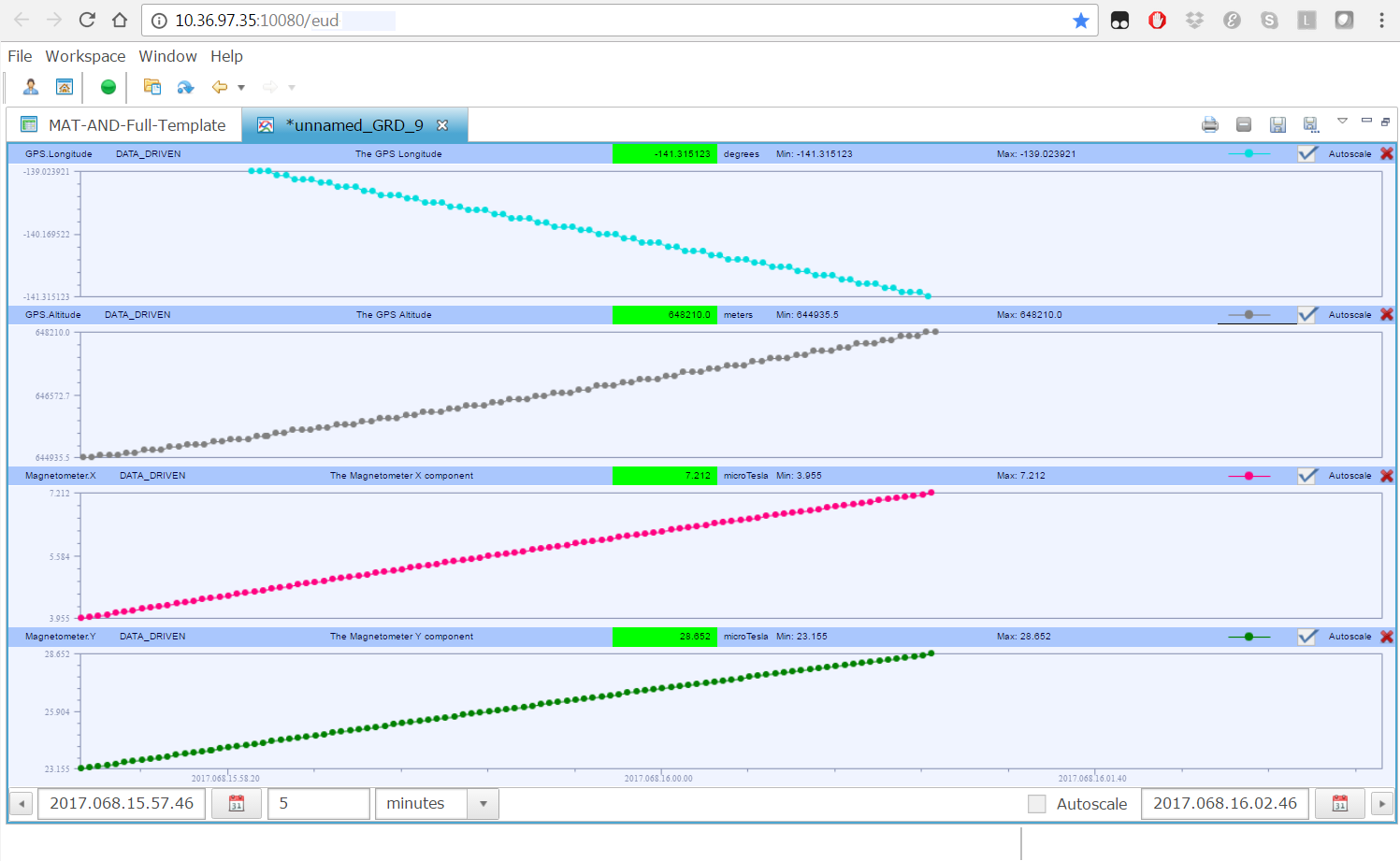
**Figure 6:** Example of EUD4MO connected to a Dummy App and accessed from a web browser.

ESA’s EGOS User Desktop (EUD) provides a framework for M&C User Interfaces for all types of ground segment systems, including ground station backend systems and mission control systems. This not only allows exploiting synergies in the UI development but also facilitates a common look and feel across all ground segment systems. It uses the Eclipse RAP framework which makes it possible to run EUD as web application as is, with only minor adjustments to the code base.



**Figure 7:** EUD4MO Parameter values display.

Figure 7is a screenshot of a view from EUD4MO that allows the visualization of parameter values in a matrix display. It is also possible to represent the received parameter values in a graphical display as the one presented in Figure 8.



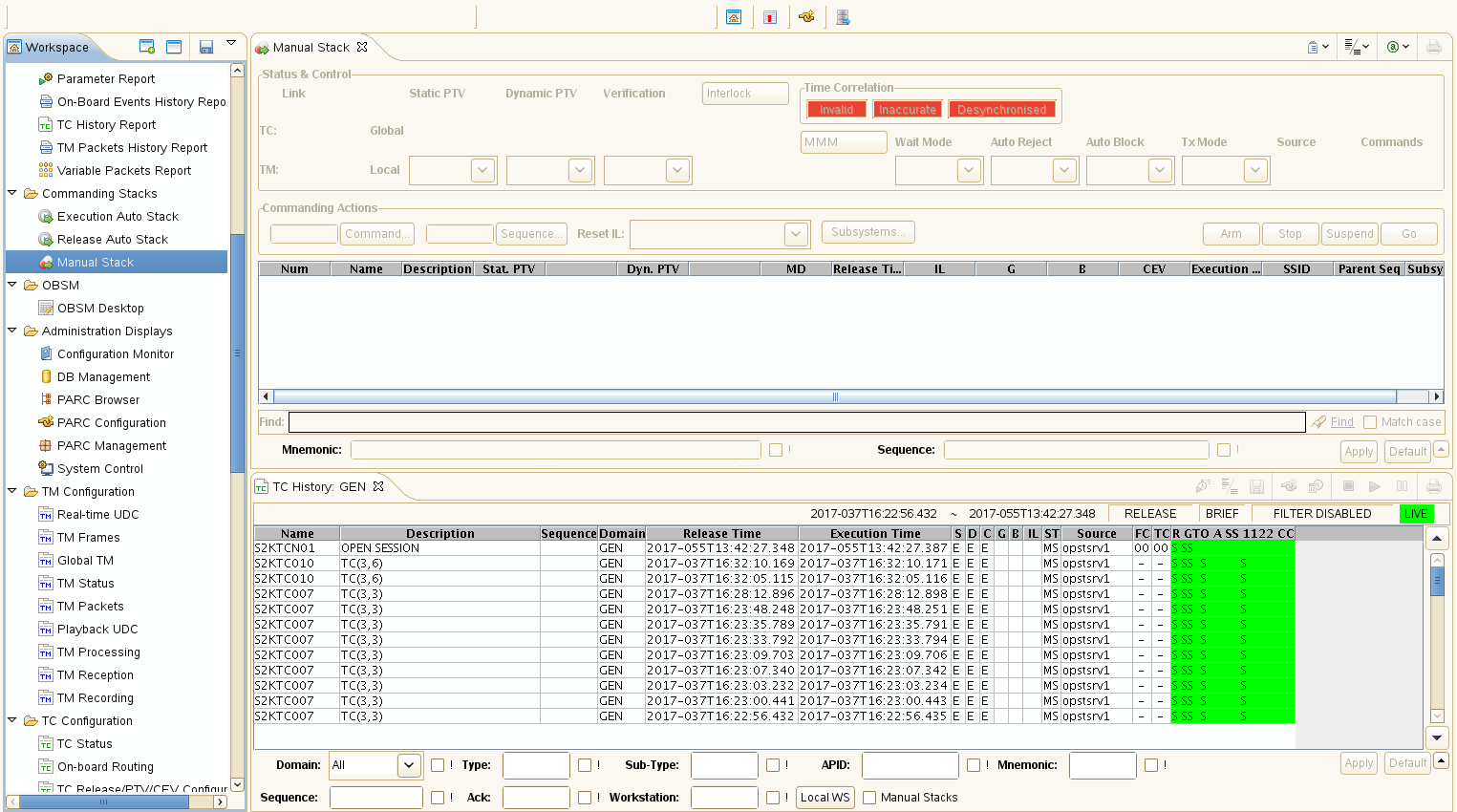
**Figure 8:** EUD4MO Parameter values plot display.

EUD4MO can also display Alert events in form of a list of messages.

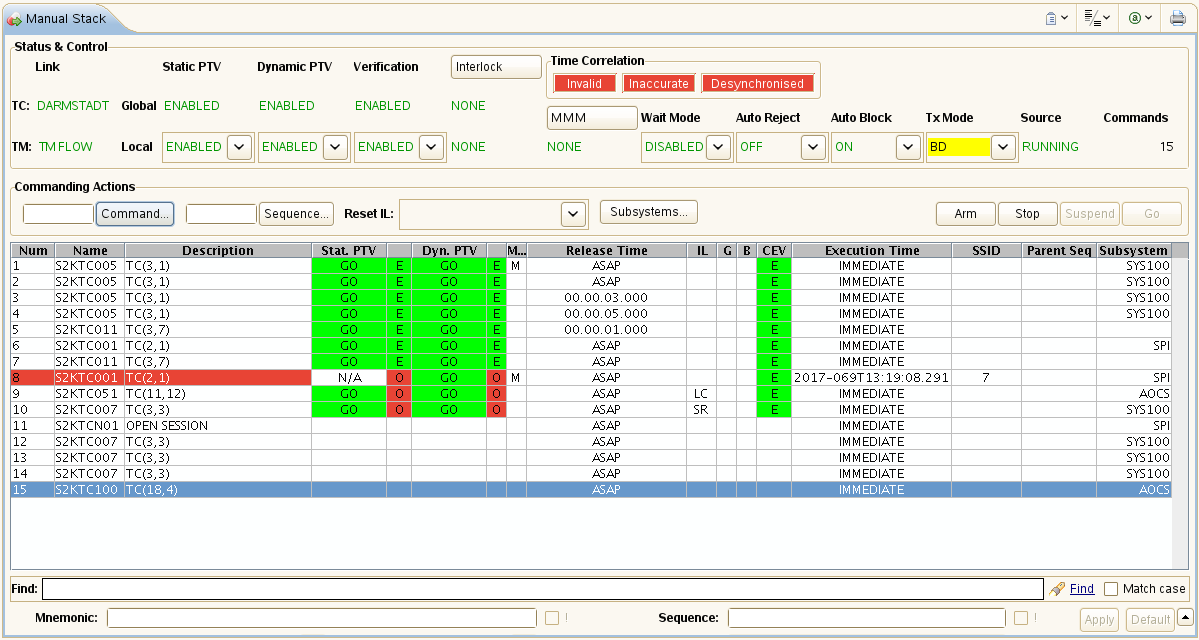


**Figure 9:** EUD4MO Alerts display.

It includes a Manual Stack in order to create stacks of actions which can be invoked using EUD4MO. Figure 10 and Figure 11 present a screenshot of the Manual Stack display.



**Figure 10:** EUD4MO Manual Stack display.



**Figure 11:** EUD4MO Manual Stack display in more detail.

### External NMF Ground aplications

External NMF Ground applications can connect to the Ground MO Proxy in order to reach the spacecraft without being hosted in ESOC’s premises. This allows external ground systems developed by experimenters to be deployed anywhere around the world and still be able to reach the spacecraft using a TCP/IP connection between the Ground MO Proxy and the external system. All the advantages of the Ground MO Proxy will then be available.

**External System**

**Ground MO Adapter**

**Ground MO Proxy: OPS-SAT**

**Figure 12:** EUD4MO Manual Stack display in more detail.

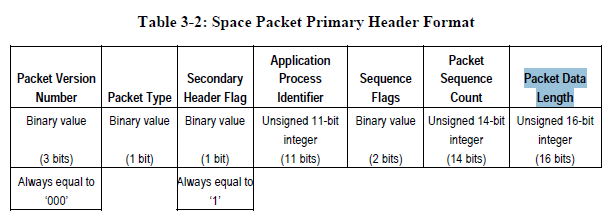
## Transport Binding

An NMF Mission has to specify the transport bindings used between the applications. As mentioned in chapter 3, the MAL-TCP/IP Transport Binding was selected to be part of NMF Generic Model and therefore it is a default transport available in all NMF Composites. For OPS-SAT, the default transport binding is not sufficient because the ground-to-space link is done using space packets.

The MAL-TCP/IP Transport Binding is used for IPC between the NMF Apps and the NanoSat MO Supervisor. Additionally, it is used on ground between the Ground MO Proxy and other NMF Ground applications. In Figure 6 these connections are represented in green.

The MAL-SPP Transport Binding and the Binary Encoding are used by the NMF Apps and also by the NanoSat MO Supervisor in order to exchange data with consumers on the ground-to-space link. On ground, it is used by the Ground MO Proxy where its protocol bridge allows other NMF Ground applications to connect to the spacecraft using to the default MAL-TCP/IP transport binding.

In OPS-SAT, the MAL-SPP Transport Binding does not completely follow the CCSDS standard. There are 2 octets being appended at the end of the Space Packet containing the CRC checksum of the message. Consequently, the Packet Data Length of the Space Packet Primary Header is incremented by 2.



**Figure 13:** Space Packet Primary Header.

Additionally, some customizations on the Binary Encoding had to be done for ESA’s OPS-SAT mission because third-party software could not operate with the default encoding. The Identifier, String, and URI MAL Data Types do not follow the Binary Encoding standard prescriptions. The following requirement is not met:

**5.21.3.** The field ‘String Length’ shall be encoded as a UInteger (see 5.18).

In the standard, a UInteger MAL Data Type is encoded as an unsigned 32-bit Integer according to requirement:

**5.18.2.** If the MCP VARINT\_SUPPORTED is FALSE, then a MAL::UInteger shall be encoded as an Unsigned 32-bit Integer (see 5.25).

However, for OPS-SAT the UInteger MAL Data Type present in the ‘String Length’ field is being encoded as an unsigned 16-bit integer.



**Figure 14:** Structure of the Identifier, String, and URI MAL Data Types after being encoded as Binary.

The Binary Encoding allows configuring the Mapping Configuration Parameters (MCPs) in order to optimize the amount of data that is being exchanged by the transport. An out-of-band agreement was defined by all parties involved in the OPS-SAT project that specify the MCPs in order to save bandwidth.

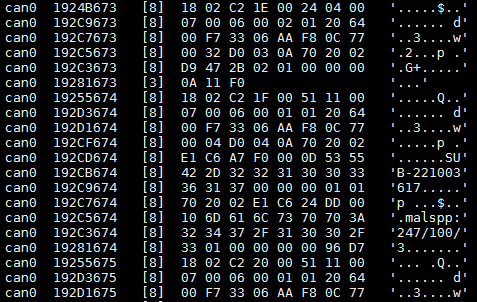
The implementation of the MAL-SPP Transport Binding used by the NanoSat MO Framework was developed by DLR and it includes the additional changes mentioned above for OPS-SAT. The implementation allows MCPs to be defined per Application Process Identifier (APID) however a default configuration was defined in order to avoid the hassle of having to define the MCPs to every single NMF App.

In OPS-SAT every experimenter has a dedicated APID that is pre-allocated by ESA. If an experimenter decides to develop an NMF App, then it will be using the APID that was assigned to the experiment. By default, all NMF Apps parse a configuration properties file (provider.properties) in order to set the APID.

After encoding the messages into space packets, they can be sent and received via different mechanisms on the layers below. On the spacecraft bus, the space packets are exchanged between the units using CAN Fragmentation Protocol (CFP), a dedicated protocol specifically defined for OPS-SAT that runs on top of the CAN bus. On ground, the space packets are exchanged between the machines using TCP/IP.

The CAN Frames have a maximum size of 8 bytes on the Data field and therefore messages longer than that have to be fragmented. CFP supports fragmentation of messages up to 32 segments which extends the limitation to 256 bytes messages. Additionally, CFP provides the possibility of having retransmission of messages that might have been lost, and finally, it also defines a source and destination fields in order to select the nodes that are sending and receiving the messages.

Under high load it was observed that the CAN bus was having problems. With Linux kernel version 3.14 the mityARM would stall entirely. After updating to 3.16, the CAN bus was able to send messages without stalling however it was observed that some of the messages were being corrupted in two of the bytes of the Data field. The problem was found to be on the Linux Kernel driver of the CAN bus and a solution was found online. The patch was provided to the manufacturers of the device and informed of the problem.



**Figure 15:** Sniff of CAN messages on the bus.

Although the space packets are exchanged between the machines on ground using TCP/IP, reliable transmission of data to the spacecraft is not guaranteed. It is important to mention this because it might induce an experimenter into wrong assumptions.

## Platform adapters integration with OPS-SAT

ESA’s OPS-SAT mission uses the Java implementation which includes the implementation of all the Platform services. Figure 16 represents how the Platform services take advantage of the adapter pattern in order to support different units.

**Platform service implementation**

**Adapter**

**MO service interface**

**Real Unit**

**Figure 16:** Generic Platform service implementation and its respective adapter to the real unit.

This section presents in more detail the integration details of the Platform adapters with the OPS-SAT peripherals. There are 5 payload peripherals in OPS-SAT that can be used by the experimenters: Camera, Optical Data Receiver, SDR, FineADCS, and GPS.

### Camera service adapter

The Camera service adapter for OPS-SAT is connected to the Camera payload device via a USB virtual serial port. OPS-SAT’s Camera is a ST200 from Hyperion Technologies and it has a power consumption of 650 mW in nominal mode, a mass of 42 grams, and an update rate of 5 Hz.

Upon initialization, the adapter checks if there is a serial port device camera and if so, it attempts to connect to it. If the device is not available, the Camera service is still started however if for example, a consumer tries to take a picture, then the service will return an error.

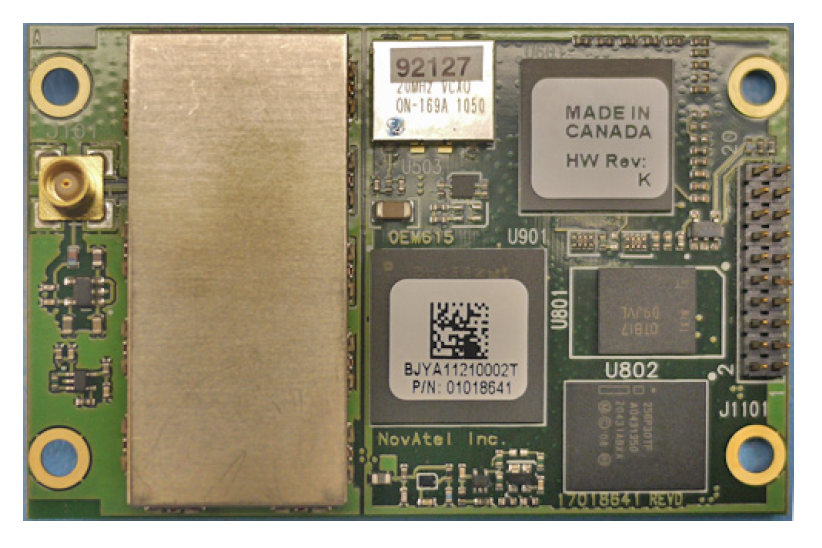
When the camera is functional and a consumer takes a picture, the adapter will double-check if the device is connected and then, send the instructions to configure it and take the picture. Finally, it copies the content of the image that is stored as an external memory device into a Picture object and sends it back to the consumer.



**Figure 17:** ST200 from Hyperion Technologies.

### GPS service adapter

The GPS service adapter for OPS-SAT is connected to the GPS payload device via an MO interface exposed by the Nanomind device on the CAN bus. OPS-SAT’s GPS is part of the OEM615 Family from NovAtel and it has low power consumption, dual frequency (L1, L2, and L2C for GPS and GLONASS), and supports multi-constellations (E1 for Galileo and B1 for BeiDou).



**Figure 18:** OEM615 from NovAtel.

The Nanomind is connected to the GPS Unit via a UART connection that allows requesting information from a set of commands. Then, the Nanomind device exposes a GPS service for interacting with the GPS unit. This service can be consumed from ground or by the Experimental Platform. The setup can be visualized on Figure 19.

**Nanomind**



**Experimental Platform**

**CAN Bus**

**UART**

**Figure 19:** GPS access from the Experimental Platform.

Although the GPS service from the Nanomind has the same name as the GPS service from the Platform services, they are completely different. The GPS service from the Nanomind includes one single operation as it can be seen in Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Identifier | Service Identifier | Area Number | Service Number | Area Version |
| OPSSAT\_PF | GPS | 75 | 21 | 1 |
| Interaction Pattern | Operation Identifier | Operation Number | Support in Replay | Capability Set |
| INVOKE | [getGPSData](#_OPERATION_GPS_getGPSData) | 1 | No | 1 |

**Table 1:** GPS service exposed by the Nanomind.

The getGPSData operation allows a consumer to receive requested data from the GPS device (for example, GPGSA data frame). The ‘command’ field must contain a request encoded as an ASCII string to be sent to the GPS receiver. Then, the returned ‘data’ field holds the GPS data frame encoded as a string of ASCII characters.

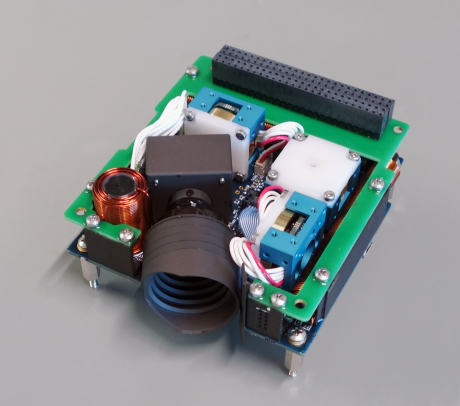
|  |  |  |
| --- | --- | --- |
| Operation Identifier | getGPSData | |
| Interaction Pattern | INVOKE | |
| Pattern Sequence | Message | Body Type |
| IN | INVOKE | command : (MAL::Blob) |
| OUT | ACK |  |
| OUT | RESPONSE | data : (MAL::Blob) |

**Table 2:** GPS service getGPSData operation.

### Autonomous ADCS service adapter

The Autonomous ADCS service adapter for OPS-SAT is connected to the ADCS payload device via I²C. OPS-SAT’s ADCS is an iADCS-100 from Berlin Space Technologies and consists of a star tracker, gyro module, reaction wheels, and magnetorquers, with the possibility to integrate external sensors such as sun sensors.

It incorporates the ADCS algorithms from the LEOS platform and allows full ADCS functionality including nadir pointing as well as autonomous target acquisition and tracking.



**Figure 20:** iADCS-100 from Berlin Space Technologies.

### Other services’ adapters

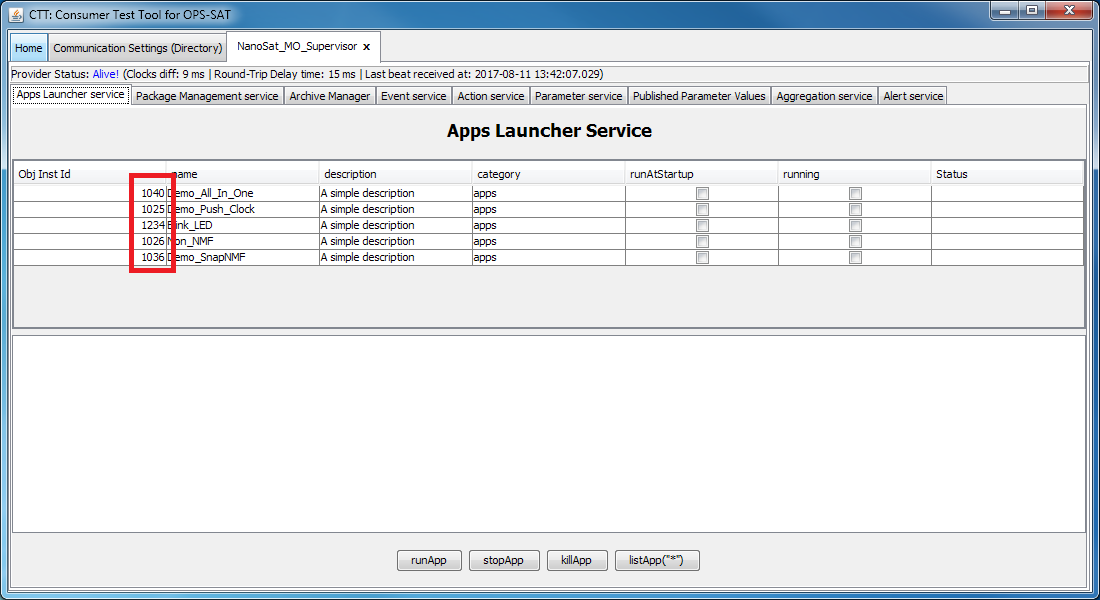
The Software-defined Radio service adapter and the Optical Data Receiver service adapter were not implemented because the units were not available during this research.

## Customization of other services

Some services from NMF’s Java implementation had to be customized to cope with ESA’s OPS-SAT mission. This involves a requirement for the management of applications based on the APID for the Apps Launcher service and an optimization to reduce the amount of data transmitted on the space-to-ground link by the Directory service upon requesting a lookup.

The Apps Launcher service needs to be able to start, stop, and kill the applications from their respective APID that was assigned by the OPS-SAT’s flight control team. The service uniquely identifies applications by an object instance identifier and therefore the service had to be tweaked to parse the APID value available in the provider.properties file and define the application’s object instance identifier to be the same as the APID value.

In OPS-SAT there are three types of experiments: applications, OS images, and FPGA images. Originally, there were different mechanism for starting and stopping each type however these have all been simplified and now they can be started from a script that is executed by the Apps Launcher service. The Apps Launcher executes the script on the runAppLin.sh file after receiving the start operation command for that application.



**Figure 21:** Apps Launcher service displaying applications with their respective object instance identifier assigned per APID.

The Directory service holds a list of providers and their respective service connection details. The NanoSat MO Supervisor includes a Central Directory service that exposes all the existing providers on the NanoSat segment. When an NMF App or the NanoSat MO Supervisor is started in OPS-SAT, its services are initialized and become operational for two different transports: TCP/IP, and SPP over CFP.

The TCP/IP transport is used for IPC and therefore this information is not relevant to a ground consumer.

In order to reduce the amount of exchanged data, a small hack was introduced in the lookupProvider operation. A ServiceFilter object is passed as an argument to the operation and inside that object there’s the field sessionName that can be abused. The hack consists of checking the sessionName field and if there’s a string with the value “s2g”, then the lookupProvider operation only returns the SPP connection details.

The “NMF Package” concept was theorized during the research process and a prototype was implemented that allows the installation, upgrade, and uninstallation of packages for demonstration. The Package Management service for OPS-SAT uses this prototype.