



esoc

European Space Operations Centre
Robert-Bosch-Strasse 5
D-64293 Darmstadt
Germany
T +49 (0)6151 900
F +49 (0)6151 90495
www.esa.int

DOCUMENT

NanoSat MO Framework - Quick Start Guide

Prepared by Cesar Coelho
Reference
Issue
Revision
Date of Issue
Status
Document Type TN
Distribution



APPROVAL

Title NanoSat MO Framework: Quick Start Guide	
Issue 1.5	Revision
Author César Coelho	Date
Approved by	Date

CHANGE LOG

Reason for change	Issue	Revision	Date

CHANGE RECORD

Issue	Revision		
Reason for change	Date	Pages	Paragraph(s)



Table of contents:

1	INTRODUCTION	4
2	REFERENCES	5
2.1	Referenced Documents	5
3	NANOSATELLITES AND OPS-SAT	6
4	MISSION OPERATIONS SERVICES	6
4.1	MO Architecture.....	6
5	APPS IN SPACE.....	7
6	NANOSAT MO FRAMEWORK: SOFTWARE DEVELOPMENT KIT (SDK).....	9
6.1	NMF Apps Lifecycle	10
6.2	CTT: Consumer Test Tool.....	11
6.3	Source Code examples	11
6.4	Playground environment	12
7	STARTING AN NMF APP DEMO	12
8	DEVELOPING AN APP	19



1 INTRODUCTION

This document provides a “Quick Start” about the main ideas and concepts about the NanoSat MO Framework.

Why NanoSat MO Framework?

Because it is a software framework for nanosatellites based on CCSDS Mission Operations services.

It introduces the concept of “apps” in space that can be started and stopped remotely. These apps can retrieve data from the nanosatellite platform through a set of well-defined MO services. Many possibilities for extensions are available due to its modular and flexible design approach which is not limited to the space segment but extends down to ground by providing all the building blocks for a complete and free end-to-end solution.

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.

By using the NMF, one can monitor and control apps without having to understand the complex details of the communication layer underneath. Additionally, controlling the nanosatellite platform is achieved through the usage of high-level interfaces, such as GPS, ADCS, Camera and others.

On Ground, the main advantage of using standardized interfaces is the possibility to reuse the same tools. For example, a Mission Control System implementing the standardized interfaces could be used to control different apps without any modification in the code and be reused across missions.

OPS-SAT is the first nanosatellite project in development by the European Space Agency and its goal is to demonstrate the arising capabilities in mission operations when spacecraft are capable of running faster on-board computers. It provides an in-orbit test-bed environment for the deployment of different experiments to test new protocols, new algorithms and new techniques in the software domain providing the perfect opportunity to test the reference implementation of the NanoSat MO Framework.

By using the NMF, it is also possible to develop portable apps which can then be reused on different platforms. For example, an experiment for OPS-SAT could be developed and tested in the developer’s local machine and then transferred to ESA facilities for a trial on a flatsat and then finally uploaded and installed in the nanosatellite.

The reference implementation of the NanoSat MO Framework was developed in Java, therefore it is necessary to have a Java Runtime Environment in order to run the software.

2 REFERENCES

2.1 Referenced Documents

Ref.	Title	Code	Issue	Date
[RD1]	NanoSat MO Framework: Achieving On-board Software Portability	AIAA 2016-2624		May 2016
[RD2]	CCSDS Mission Operations Services on OPS-SAT	IAA-B10-1301		April 2015
[RD3]	CCSDS Mission Operations Directory service: Paving the road for a new world of services' discoverability			April 2015
[RD4]	OPS-SAT Experiments' Software Management with the NanoSat MO Framework	AIAA 2016-5301		September 2016
[RD5]	NanoSat MO Framework: When OBSW turns into apps	IEEE AeroConf 10.0305		Mar 2017
[RD6]	OPS-SAT: Preparing for the Operations of ESA's First NanoSat	AIAA 2016-2490		May 2016



3 NANOSATELLITES AND OPS-SAT

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 of a space mission 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. [RD1]

ESA and its European industry partners generate every year many new and innovative ideas for advancing European space technology regarding mission operations but the majority of these innovations never make it to orbit. OPS-SAT emerged, providing a low cost in-orbit laboratory available for authorised experimenters to test, demonstrate and validate their development software experiments. OPS-SAT is the first CubeSat designed by ESA and is a safe experimental platform which shall fly in a LEO dawn-dusk orbit. OPS-SAT makes available a reconfigurable platform, at every layer from channel coding upwards, and it will be available for experimenters wishing to test and demonstrate new software and mission operation concepts. [RD2]

By determining the typical peripherals on-board of different nanosatellites, it is possible to find commonalities among them and subsequently generate generic interfaces that give access to the main functionalities of these devices. With this in mind, the NanoSat MO Framework contains a set of services which allow a developer to interact and have access to the platform peripherals. These are called: “Platform services”.

4 MISSION OPERATIONS SERVICES

CCSDS Mission Operations (MO) is a set of standard 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 it is intended to be used for mission operations of future space missions.

Please take a few minutes to watch the following introductory video:

<https://www.youtube.com/watch?v=XdGeaJE7yEk>

4.1 MO Architecture

The MO layered service framework, shown in the picture below, allows mission operation services to be specified in an implementation and communication agnostic manner. The core of the MO service framework is its Message Abstraction Layer (MAL) which ensures interoperability between mission operation services deployed on different framework implementations.

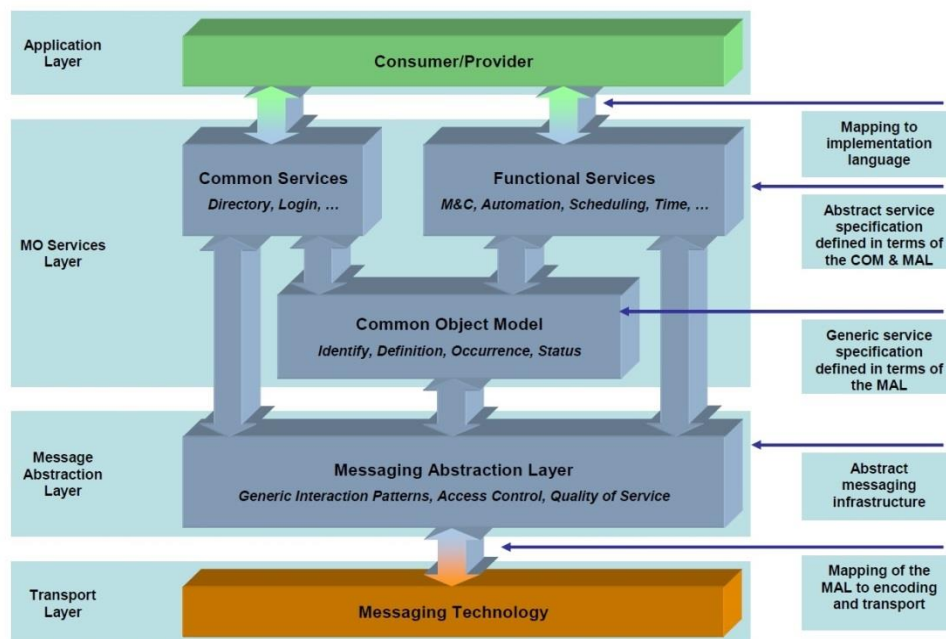


Figure 1: MO services Architecture diagram

The MO services are specified in compliance to a reference service model, using an abstract service description language, the MAL. This is similar to how Web Services are specified in terms of Web Services description language WSDL. For each concrete deployment, the abstract service interface must be bound to the selected software implementation and communication technology. The MAL layer provides in turn standardized interfaces in form of Application Programming Interfaces (API) towards both upper and lower layers. The Common Object Model (COM) provides a standard data object model for MO Services to utilise. Whereas the MAL provides the building blocks that can be used to define the operations of a MO service, the COM provides the building blocks for the specification of the data objects of a service. This builds upon the MAL to define a standard data model for an MO service.

To learn more about CCSDS Mission Operations services, use following guide:

https://github.com/esa/CCSDS_MO/wiki/CCSDS-Mission-Operations-services-for-Newbies

5 APPS IN SPACE

The classical on-board software (OBSW) is seen as an almost immutable software which is developed for a particular use where future updates would imply patching specific memory areas. Moreover, many times the OBSW is written for embedded devices which will never be updated again.

The sudden increase of commercial of-the-shelf (COTS) hardware for nanosatellites with great amounts of processing power are allowing space community to think and move towards spacecraft with system environments capable of running full Operating Systems. OPS-SAT is a good example of that, as it provides an experimental platform composed of a MityARM device with 1 GB of RAM, an ARM processor and a lightweight version of Linux.

The growing market of smartphones and tablets brought new ideas into software by providing quick development of software using well defined libraries from Android and iOS. Additionally, Android apps gave the community an extra degree of freedom by making the mobile apps portable entities of software, where the same app can run on the hardware of different vendors as long as both devices have the same underlying Android framework.

The NanoSat MO Framework intends to change the current view on OBSW by turning it into apps that can be easily developed, tested, deployed and updated at any time without causing any major problem to the spacecraft. Furthermore, it will be possible to use the exact same app on different spacecraft platforms as long as the NanoSat MO Framework is used. In principle, an app developed to OPS-SAT using the NanoSat MO Framework will be able to, for example, publish telemetry, receive telecommands or access the GPS device on a different nanosatellite without any change in the code. [RD5]

The framework includes a wide set of services. Therefore, they were assembled together into a generic model for extension. The components extending the generic model are named NMF Composites and they are specialized for a specific segment and particular purpose.

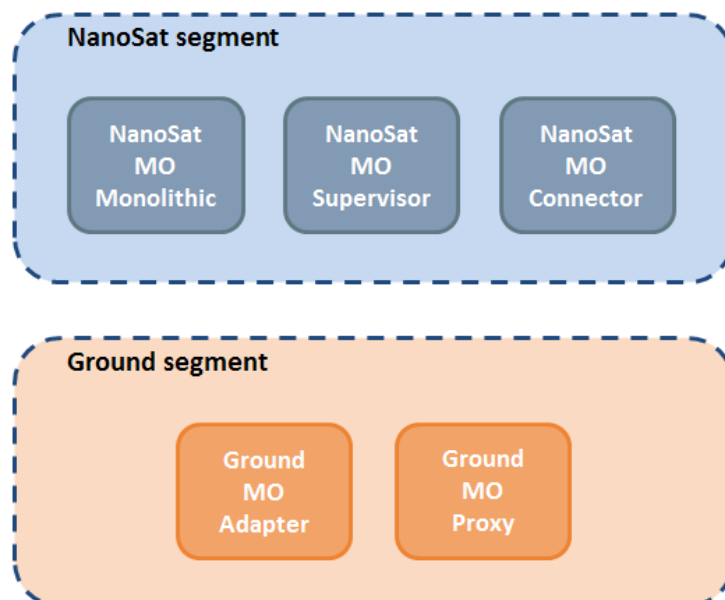


Figure 2: NMF Composites



The design of the NMF Composites is done in a modular and flexible manner which allows them to be reconfigured or adapted depending on the needs in the overall design of the system. This is similar to a Lego® type approach where the bricks can be recombined to form something different.

The objective of the NMF Composites is to provide prebuilt components that allow quick development of new software solutions that are interoperable in end-to-end scenarios.

6 NANOSAT MO FRAMEWORK: SOFTWARE DEVELOPMENT KIT (SDK)

The NanoSat MO Framework Software Development Kit (NMF SDK) is a set of development tools and software source code that facilitate the creation of applications with the NanoSat MO Framework. The NMF SDK is intended to be the starting point for a software developer willing to develop applications with the NMF.

It is composed of:

- Consumer Test Tool (CTT)
- Demos for NMF Ground applications development
- Demos of NMF Apps
- NMF Packager Assembler
- NMF Playground
- Documentation

In a traditional European mission, the separation between the stakeholders is commonly done in the Ground-Space link by defining an interface that is usually a tailored version of the ECSS Packet Utilization Standard (PUS). This creates a clear separation between the ground software development and the space software development. The latter is usually developed in a monolithic way covering all the functionalities of the spacecraft.

Comparatively, when a mobile app is developed, it is never intended to cover all the possible functionalities of a normal phone but instead they are software entities dedicated to executing particular tasks in order to reach the user goals. Extending this idea to nanosatellites can potentially generate a new set of space software developers focused on particular satellite operations and functionalities, just as the Uber™ app and Snapchat™ app have very different purposes to the user.

Following the above line of thinking, one can split the problem domain in three parts: NMF Ground Development, NMF Mission Development, and NMF App Development.

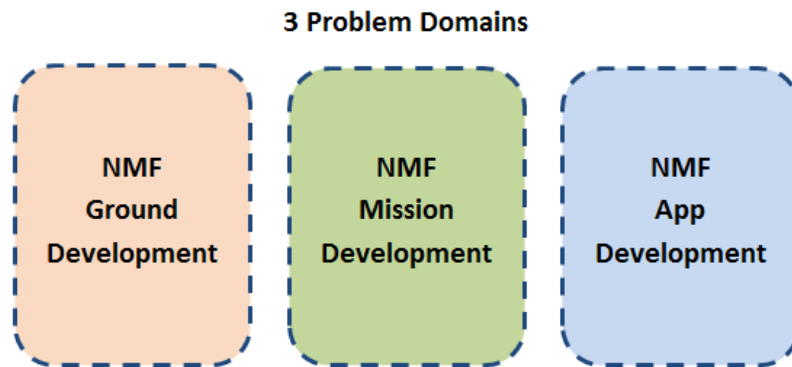


Figure 3: NanoSat MO Framework problem domains

6.1 NMF Apps Lifecycle

An NMF app can be developed by adding the correct dependency in the POM file of the project (in case maven is used) or by simply copying the source code of one of the many demo projects available in the SDK.

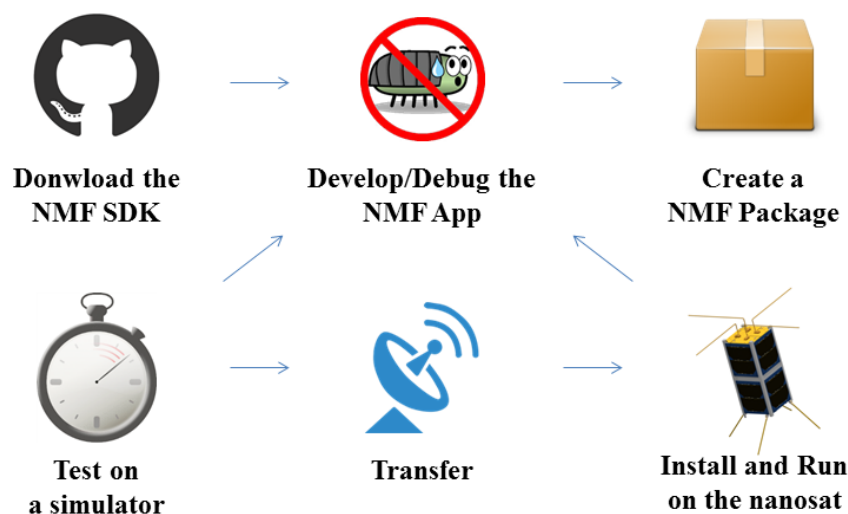


Figure 4: The lifecycle of an NMF App

An NMF app can be developed using the NMF SDK and then packaged into a NMF Package using the NMF Package Assembler. After, the app can be tested on a simulator which can be purely software-based or a Flatsat emulating the nanosatellite's hardware. Then transferred to the nanosatellite where it can be installed and started at any time. If a problem is found during the execution in the nanosatellite, it is possible to go back to the development phase and repeat the process.

6.2 CTT: Consumer Test Tool

The Consumer Test Tool (CTT) is an application to aid developers in the process of testing newly developed NMF Apps from a user-friendly interface. It includes a set of GUI panels that provides a Graphical User Interface (GUI) to interact with the services on a provider. CTT can also be used as a learning tool for MO services, or to connect to MO providers other than NMF providers.

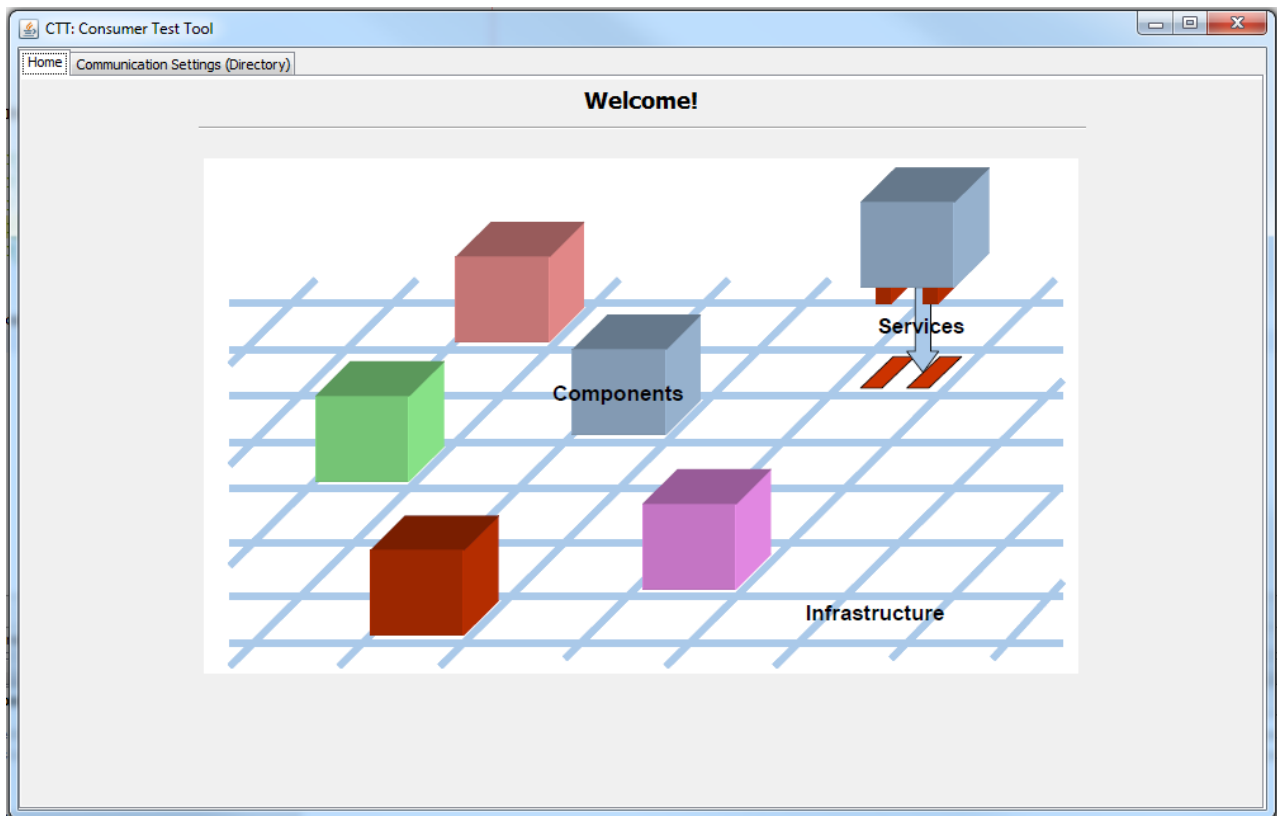


Figure 5: Splash screen of CTT

6.3 Source Code examples

The NMF SDK's source code examples are available in the "src" folder of the NMF SDK. This includes demos of NMF Apps, demos of NMF Ground applications, a demo of a Monolithic Provider, and some other projects.

The NMF App demos included are the following:

- Hello World
- PushClock
- 10 seconds Alert
- 5 steps Action

- GPS Data
- All in One
- SnapNMF
- Serialized object

There are also NMF Ground application demos:

- Ground Zero
- Ground with Directory
- Ground Set and Command
- Facebook

6.4 Playground environment

The Playground is a multi-purpose environment for experimenting with NMF-related software and it includes a set of NMF Apps ready to be used.

The Playground environment can be used as a learning tool because the NMF Apps are already compiled and can just be started by running the script `runAppWin.bat` in Windows or `runAppLin.sh` in Linux. This allows a newcomer to get familiar with the NMF ideas without actually doing any code.

During the development of an NMF Ground application, usually it is necessary to test the application by connecting it to a provider side. The NMF Apps available in the Playground can be used for this purpose and save time to a ground developer.

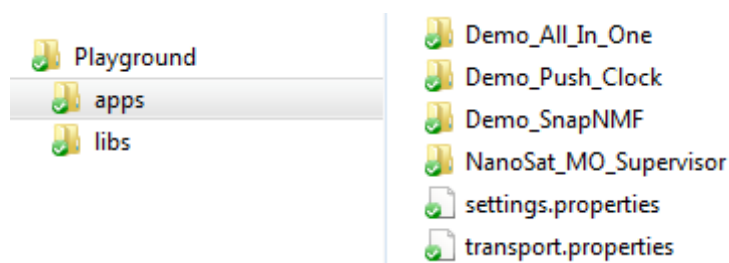


Figure 6: Playground environment

7 STARTING AN NMF APP DEMO

As presented before, a set of ready to be used NMF Apps is available on the Playground environment. All of them include a `runAppWin.bat` and a `runAppLin.sh` executable files in order to start the applications.

Given the service-oriented nature of the MO services, there are always two endpoints, a provider and a consumer. Now one just needs to connect the consumer to the NMF App provider.



Figure 7: Basic Consumer-Provider interaction

The NMF Apps can be started using two different approaches:

1. Directly from the executable file in the folder
2. Using the NanoSat MO Supervisor

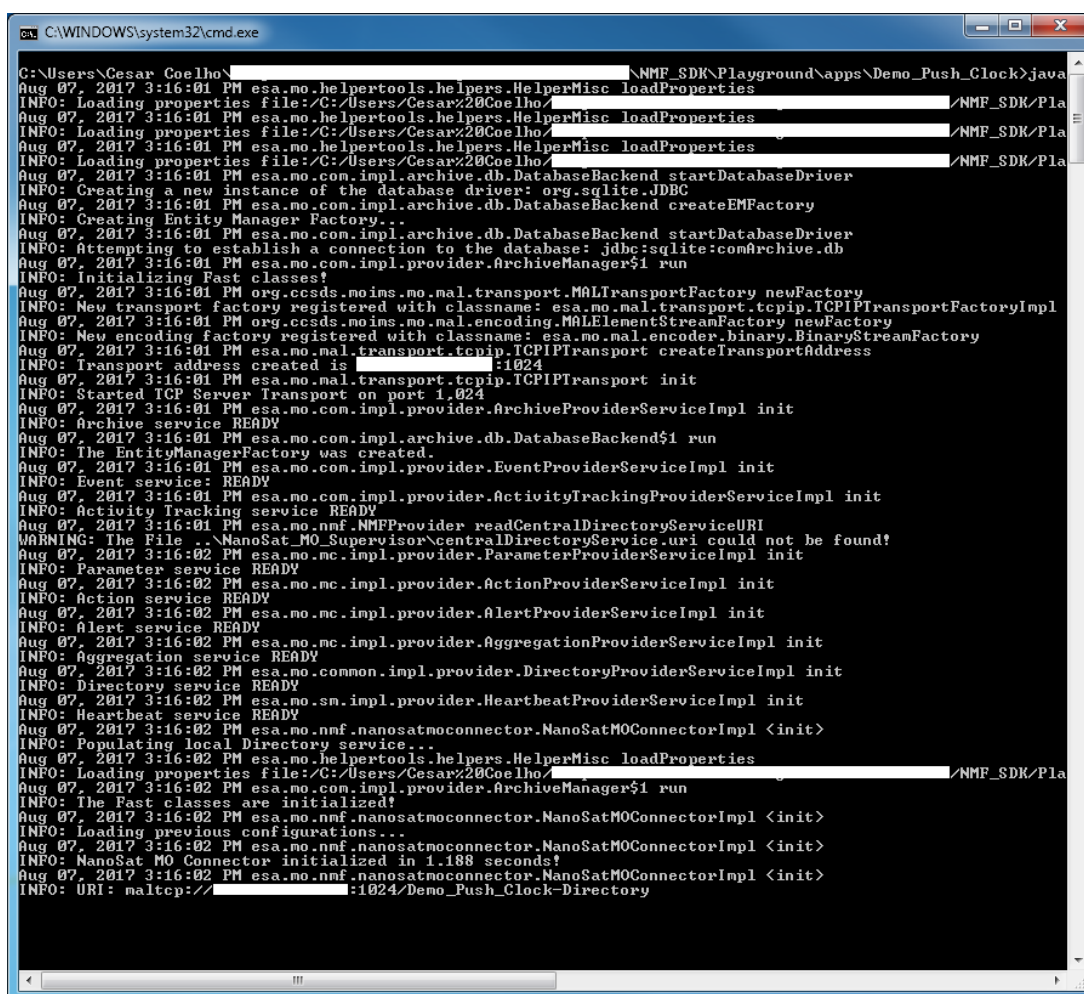
Let's begin with the first approach: Directly from the executable file

One can start the “PushClock” demo app by running the shortcut on the following path:
(Windows machine) Playground\apps\Demo_Push_Clock\runAppWin.bat

OR:

(Linux machine) Playground/apps/Demo_Push_Clock/runAppLin.sh

A similar window should pop up:



```

C:\WINDOWS\system32\cmd.exe
C:\Users\Cesar Coelho\... \NMF_SDK\Playground\apps\Demo_Push_Clock>java
Aug 07, 2017 3:16:01 PM esa.mo.helpertools.helpers.HelperMisc loadProperties
INFO: Loading properties file:/C:/Users/Cesar/20Coelho/.../NMF_SDK/Pla
Aug 07, 2017 3:16:01 PM esa.mo.helpertools.helpers.HelperMisc loadProperties
INFO: Loading properties file:/C:/Users/Cesar/20Coelho/.../NMF_SDK/Pla
Aug 07, 2017 3:16:01 PM esa.mo.helpertools.helpers.HelperMisc loadProperties
INFO: Loading properties file:/C:/Users/Cesar/20Coelho/.../NMF_SDK/Pla
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.archive.db.DatabaseBackend startDatabaseDriver
INFO: Creating a new instance of the database driver: org.sqlite.JDBC
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.archive.db.DatabaseBackend createEMFactory
INFO: Creating Entity Manager Factory...
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.archive.db.DatabaseBackend startDatabaseDriver
INFO: Attempting to establish a connection to the database: jdbc:sqlite:comArchive.db
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.provider.ArchiveManager$1 run
INFO: Initializing Fast classes!
Aug 07, 2017 3:16:01 PM org.ccsds.moims.mo.mal.transport.MALTransportFactory newFactory
INFO: New transport factory registered with classname: esa.mo.mal.transport.tcpip.TCPIPTransportFactoryImpl
Aug 07, 2017 3:16:01 PM org.ccsds.moims.mo.mal.encoding.MALElementStreamFactory newFactory
INFO: New encoding factory registered with classname: esa.mo.mal.encoder.binary.BinaryStreamFactory
Aug 07, 2017 3:16:01 PM esa.mo.mal.transport.tcpip.TCPIPTransport createTransportAddress
INFO: Transport address created is :1024
Aug 07, 2017 3:16:01 PM esa.mo.mal.transport.tcpip.TCPIPTransport init
INFO: Started TCP Server Transport on port 1,024
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.provider.ArchiveProviderServiceImpl init
INFO: Archive service READY
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.archive.db.DatabaseBackend$1 run
INFO: The EntityManagerFactory was created.
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.provider.EventProviderServiceImpl init
INFO: Event service: READY
Aug 07, 2017 3:16:01 PM esa.mo.com.impl.provider.ActivityTrackingProviderServiceImpl init
INFO: Activity Tracking service READY
Aug 07, 2017 3:16:01 PM esa.mo.nmf.NMFProvider readCentralDirectoryServiceURI
WARNING: The File ..\NanoSat_MO_Supervisor\centralDirectoryService.uri could not be found!
Aug 07, 2017 3:16:02 PM esa.mo.mc.impl.provider.ParameterProviderServiceImpl init
INFO: Parameter service READY
Aug 07, 2017 3:16:02 PM esa.mo.mc.impl.provider.ActionProviderServiceImpl init
INFO: Action service READY
Aug 07, 2017 3:16:02 PM esa.mo.mc.impl.provider.AlertProviderServiceImpl init
INFO: Alert service READY
Aug 07, 2017 3:16:02 PM esa.mo.mc.impl.provider.AggregationProviderServiceImpl init
INFO: Aggregation service READY
Aug 07, 2017 3:16:02 PM esa.mo.common.impl.provider.DirectoryProviderServiceImpl init
INFO: Directory service READY
Aug 07, 2017 3:16:02 PM esa.mo.sm.impl.provider.HeartbeatProviderServiceImpl init
INFO: Heartbeat service READY
Aug 07, 2017 3:16:02 PM esa.mo.nmf.nanosatmoconnector.NanoSatMOConnectorImpl <init>
INFO: Populating local Directory service...
Aug 07, 2017 3:16:02 PM esa.mo.helpertools.helpers.HelperMisc loadProperties
INFO: Loading properties file:/C:/Users/Cesar/20Coelho/.../NMF_SDK/Pla
Aug 07, 2017 3:16:02 PM esa.mo.com.impl.provider.ArchiveManager$1 run
INFO: The Fast classes are initialized!
Aug 07, 2017 3:16:02 PM esa.mo.nmf.nanosatmoconnector.NanoSatMOConnectorImpl <init>
INFO: Loading previous configurations...
Aug 07, 2017 3:16:02 PM esa.mo.nmf.nanosatmoconnector.NanoSatMOConnectorImpl <init>
INFO: NanoSat MO Connector initialized in 1.188 seconds!
Aug 07, 2017 3:16:02 PM esa.mo.nmf.nanosatmoconnector.NanoSatMOConnectorImpl <init>
INFO: URI: maltcp://:1024/Demo_Push_Clock-Directory
  
```

Figure 8: PushClock demo displaying the initialization of the NanoSat MO Framework

The NMF App is now running!

One can start the Consumer Test Tool by running the shortcut on the following path:

(Windows machine) CTT\runCTT.bat

or:

(Linux machine) CTT/runCTT.sh

Go to the “Communication Settings (Directory)” tab to set the provider URI, insert the provider URI and fetch the information. Then, press “Connect to Selected Provider” to establish the connection to the NMF App:

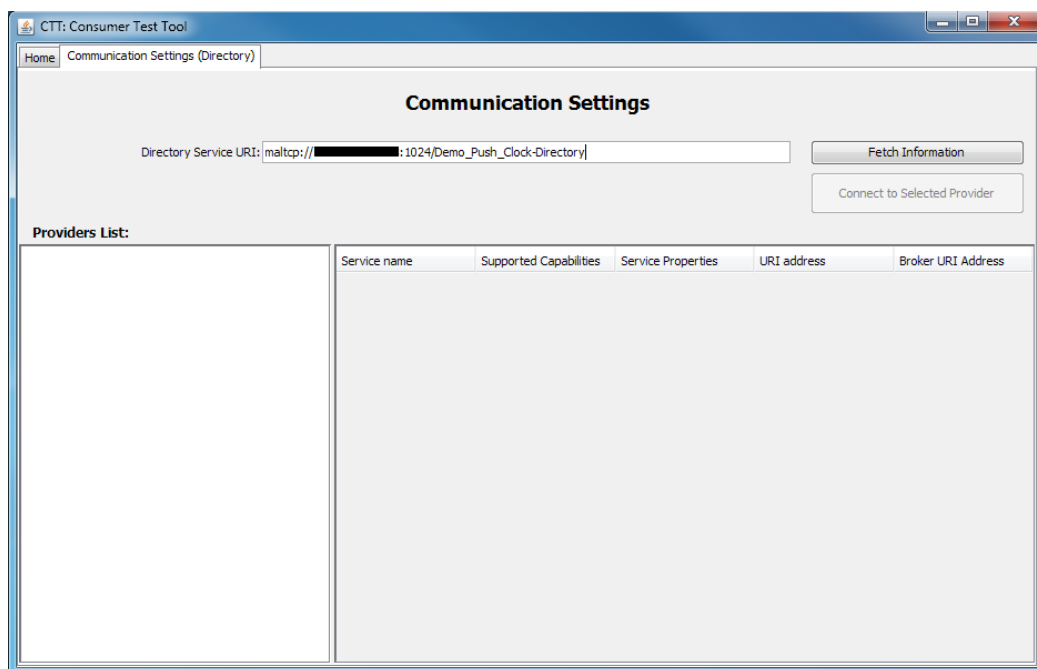
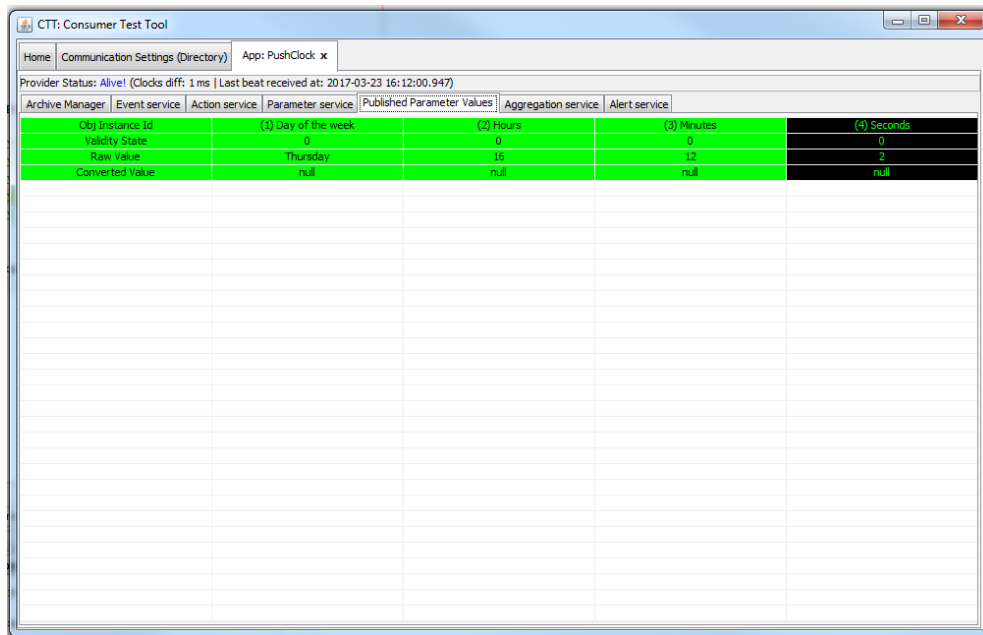


Figure 9: Consumer Test Tool - Communication Settings tab

The CTT will connect to all the services and open a new tab for the connected provider:



The screenshot shows the 'CTT: Consumer Test Tool' window. The 'Published Parameter Values' tab is active, displaying a table with the following data:

Obj Instance Id	(1) Day of the week	(2) Hours	(3) Minutes	(4) Seconds
Validity State	0	0	0	0
Raw Value	Thursday	16	12	2
Converted Value	null	null	null	null

Figure 10: Consumer Test Tool connected to the PushClock demo

One can now interact with the PushClock demo.

It is possible to visualize that on the Published Parameter Values tab, the Day of the week, Hours, Minutes and Seconds are being published.

The 3 first parameters are only published every minute and so they might not appear right after connecting to the app.

Let's now try the second approach: Using the NanoSat MO Supervisor

In order to start multiple applications remotely in the same nanosatellite, a supervisor application was created which is responsible for starting, stopping, and killing them.

One can start the “NanoSat MO Supervisor” application by running the shortcut on the following path:

(Windows machine) Playground\apps\NanoSat_MO_Supervisor\runSupervisor.bat

or:

(Linux machine) Playground/apps/NanoSat_MO_Supervisor/runSupervisor.sh

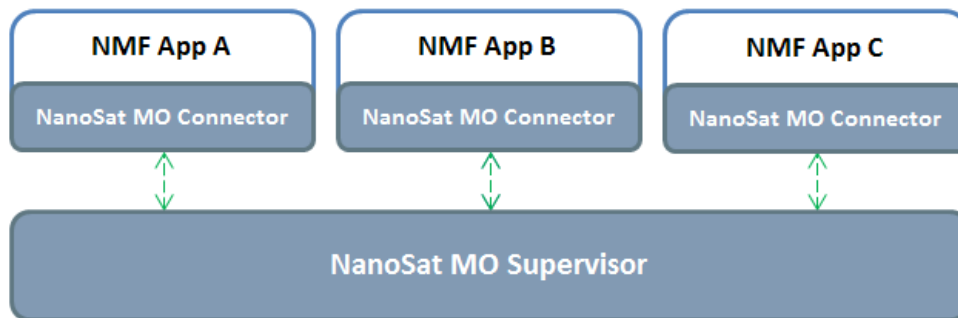


Figure 11: NanoSat MO Supervisor can initialize NMF Apps

After starting the NanoSat MO Supervisor, one can connect to it using CTT. Starting an application from CTT is easy, go to “Apps Launcher” tab, select the application to be started and press the “runApp” button.

Let’s select the All in One demo and start it.

A confirmation message stating that the NanoSat MO Connector was initialized should be presented on the application’s output. During the initialization, the application registered on the Central Directory service of the NanoSat MO Supervisor which means that all the connections for its services can be found on it.

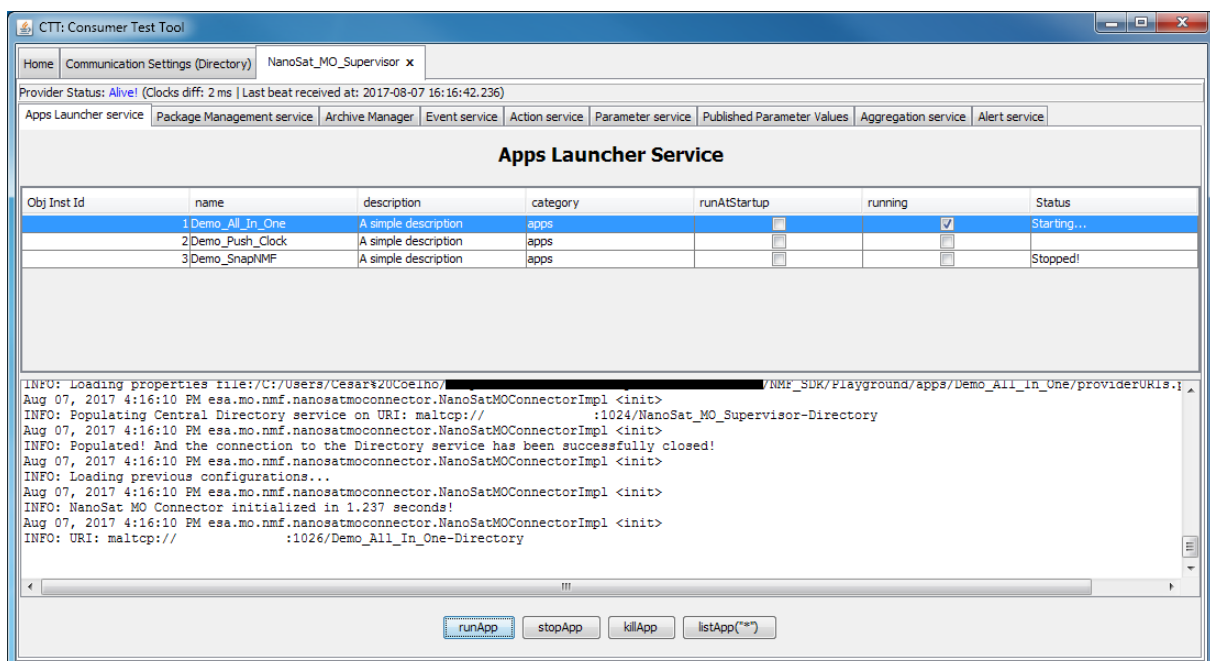


Figure 12: Apps Launcher service from the NanoSat MO Supervisor

One can now connect to the started application by going to the “Communication Settings” tab and fetch the Central Directory service once again. Select the application to connect to and press “Connected to Selected Provider”.

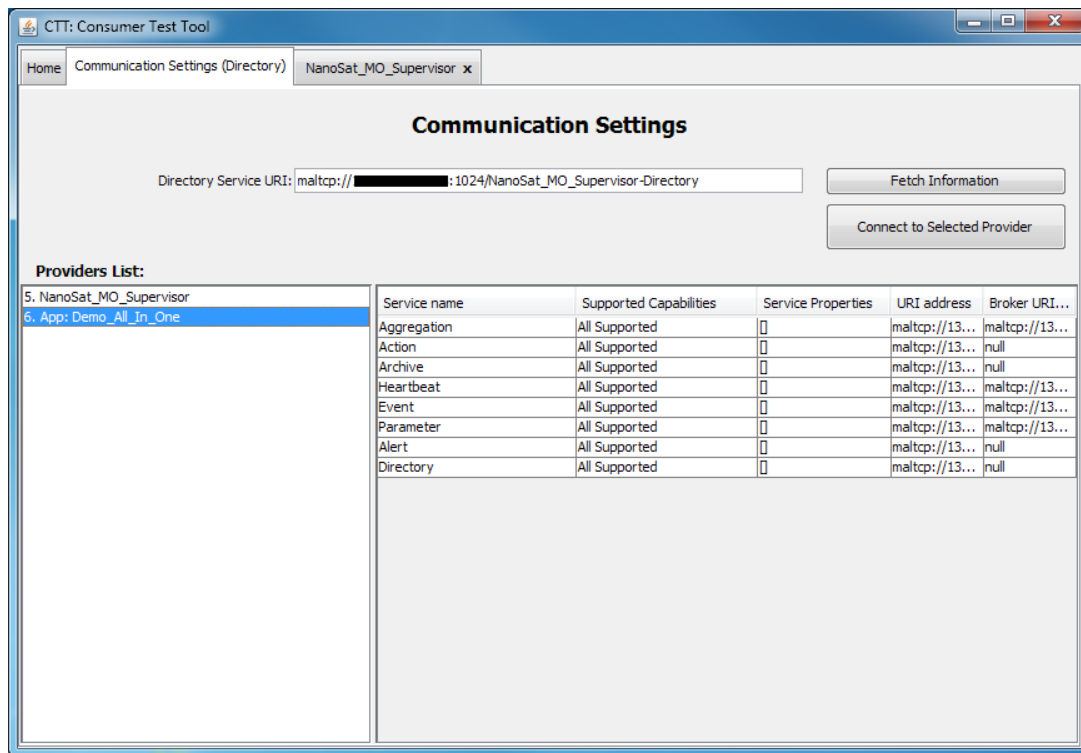


Figure 13: Central Directory with two MO providers

The All in One demo allows a consumer to try all the available features which includes all the M&C services.

The Parameter service is already configured with some parameters such as GPS.Latitude, GPS.Longitude and GPS.Altitude which can be seen in the Parameter service tab. Go to the “Parameter service” tab, select one of the parameters and press `getValue()` in order to receive the value of the selected parameter on demand.

To receive periodic updates of parameters, one must update the `updateInterval` field of the parameter definition to the desired rate and then enable the generation. If the user press `enableGeneration()`, the service will start to periodically send the parameter value for the selected parameter. The reception can be visualized in the “Published Parameter Values” tab.

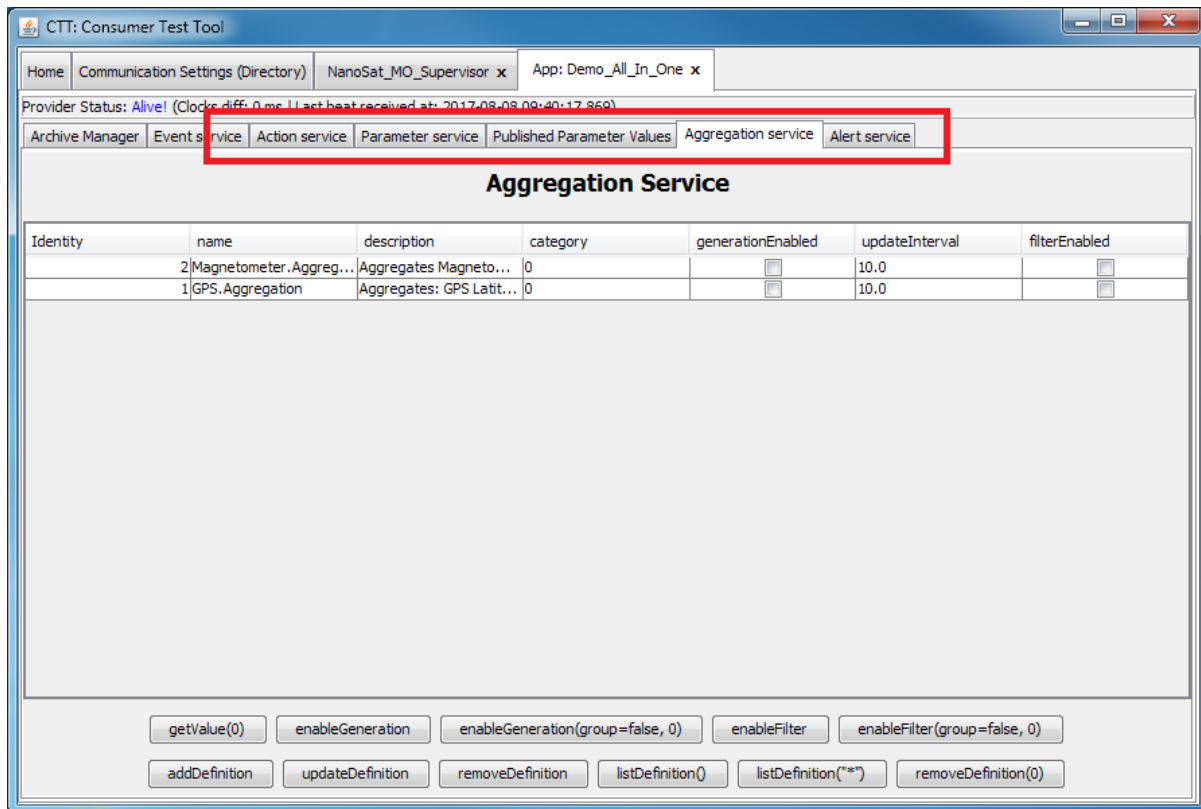


Figure 14: CTT connected to the All in One demo

The Aggregation service allows the user to group parameters together and retrieve the values simultaneously. As an example, an aggregation can be defined by going to the “Aggregation service” tab and pressing “addDefinition”. Then, a filled-in template will pop up that can be edited as the user wishes. It is important to edit the “parameterSets” field and inside, edit the first entry of the list which contains a “parameters” field that must be edited to include the the objIds of the parameter definitions available in the “Parameter service” tab.

The “Alert service” tab allows the management of Alert definitions (for example, enabling/disabling the generation). In this demo, one single Alert definition was defined and a corresponding Alert is generated every ten seconds. In MO, Alerts are defined as COM Events and therefore their reception can be visualized in the “Event service” tab.

The “Action service” tab includes a set of Actions that were defined by the application. This includes one that demonstrates the reporting of five stages for a particular action and three others that allow interacting with the simulated ADCS. The latter are connected to the Platform services on the provider side and use the AutonomousADCS service.

For further investigation, one can check directly the code of this NMF App which is available in the “src” folder of the SDK.



8 DEVELOPING APPLICATIONS

Developing an NMF App or an NMF Ground application that can be used by different nanosatellites platforms is possible by using the NanoSat MO Framework.

The development processes are described in the “docs” folder:

- Development Guide for NMF Apps
- Development Guide for NMF Ground applications

Please notice that:

The specification for the MO services are available in the “docs/MO_services” folder.

The javadocs of the NanoSat MO Framework are available in the “docs/Javadocs” folder.