|  |
| --- |
|  |

OPS-SAT Space to Space ICD

Approval

|  |  |
| --- | --- |
| Title OPS-SAT Space to Space ICD | |
| Issue Number | Revision Number |
| Author Cesar Coelho | Date 01/12/2016 |
| Approved By | Date of Approval |
|  |  |

Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Reason for change | Issue Nr. | Revision Number | Date |
|  |  |  |  |

Change Record

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

Distribution

|  |
| --- |
| Name/Organisational Unit |
|  |

Table of contents:

1 Introduction 4

2 MO Providers in OPS-SAT 5

3 Network, Data Link and Physical layers 6

4 List of services between SEPP and Nanomind 7

5 Provider: GMV’s OBSW 8

5.1 Service: Power 8

5.1.1 General 8

5.1.2 OPERATION: powerOffSBandTX 8

5.1.3 OPERATION: powerOnSBandTX 8

5.1.4 OPERATION: powerOffSBandRX 9

5.1.5 OPERATION: powerOnSBandRX 9

5.1.6 OPERATION: setPowerState 9

5.2 Service: ExperimentWD 10

5.2.1 General 10

5.2.2 OPERATION: alive 10

5.3 Service: GPS 11

5.3.1 General 11

5.3.2 OPERATION: getGPSData 11

5.4 Area data types: OPSSAT\_PF 12

5.4.1 ENUMERATION: PayloadDevice 12

6 Provider: NanoSat MO Supervisor 13

6.1 Service: AppsLauncher 13

6.1.1 General 13

6.1.2 High Level Requirements 13

6.1.3 Functional Requirements 13

6.1.4 COM usage 13

6.1.5 COM Event Service usage 14

6.1.6 COM Object Relationships 14

6.1.7 OPERATION: monitorExecution 15

6.1.8 OPERATION: runApp 16

6.1.9 OPERATION: killApp 16

6.1.10 OPERATION: stopApp 17

6.1.11 OPERATION: listApp 18

6.2 Service data types: AppsLauncher 18

6.2.1 Composite: AppDetails 18

7 Examples 20

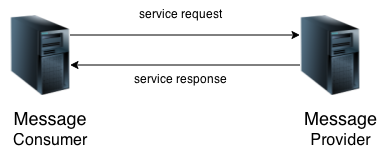
7.1 Starting an App from the Nanomind 20

7.2 Stopping an App from the Nanomind 21

# Introduction

This document contains is the Interface Control Document for OPS-SAT between the SEPP and the Nanomind, and vice-versa.

A service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online.



The CCSDS Mission Operations (MO) services are a set of standardized end-to-end services which are currently being defined by the Consultative Committee for Space Data Systems (CCSDS) and are intended to be used for mission operations of future space assets. The MO services are defined in a layered service-oriented architecture that allows them to be specified in an implementation and communication agnostic manner.

Following the spirit of CCSDS Mission Operations services, this ICD will specify the services using directly the content generated from the docxGenerator tool available on GitHub. This tool generates docx documents directly from the xml services specification.

The NanoSat MO Framework includes a high-level component named NanoSat MO Supervisor that allows having multiple apps sharing the same Platform services for the interaction with the peripherals on-board. Additionally, it brings software management capabilities such as: starting, stopping, installing, updating and uninstalling apps on-board of the spacecraft. For OPS-SAT, an Application has been developed using this component which will be denominated NanoSat MO Supervisor (NMS) on the pages that follow.

This ICD specifies which services are exposed from the Nanomind to the SEPP.

This ICD specifies which services are exposed from the SEPP to the Nanomind.

Services exposed by the Nanomind are consumed by the SEPP (specifically NMS).

Services exposed by the SEPP (specifically NMS) are consumed by the Nanomind.

# MO Providers in OPS-SAT

In OPS-SAT, the following MO providers (and respective APIDs) exist:

* GMV’s OBSW (APID: 10)
* NanoSat MO Supervisor (NMS) (APID: 100)
* File App (File Management service) (APID: 1111)

And possibly:

* Experiments (apid: [1024 - 1535])

Please notice that the CFDP APIDs are different from the File Management service APID. CFDP APID: 127 (CAN) & 2045 (SPI))

MO providers are software entities and not physical hardware devices therefore one must specify where the providers are running. The providers deployment takes place on the following hardware:

* GMV’s OBSW: on the Nanomind
* NanoSat MO Supervisor: on the SEPP
* File App: on the SEPP
* Experiments: on the SEPP

Please notice that the File App wraps the MO File Management services however these are not exposed to the Nanomind and instead exposed directly to ground. Thus, the File Management service definition does not belong in this ICD.

# Network, Data Link and Physical layers

This ICD is described at the service level and therefore the layer below also need to be explicitly defined in the document.

The Physical layer between the SEPP and the Nanomind is Controller Area Network (CAN bus) using the extended CAN Frame Format composed of 29 bits identifier and a maximum of 8 bytes data length.

To exchange messages on the CAN bus, an OPS-SAT-specific protocol has been defined. This protocol is CAN Fragmentation Protocol (CFP) and defines a set of rules to exchange messages between the different nodes and also provides fragmentation to allow the exchange of messages up to 256 bytes.

CFP is expecting to hold packets on the data field because the CCSDS Engine does routing based on the APIDs.

The services defined in this ICD are expecting to follow the CCSDS MAL Space Packet Protocol binding book with some changes on the encoding specifically done for OPS-SAT.

Changes on the encoding were done in order for SCOS to be able to talk with the Nanomind.

To avoid having different encodings on the Nanomind for different destinations, it was decided to follow one single one, in this case, the one of the more constrained connection.

# List of services between SEPP and Nanomind

Services are uniquely identified by their URI. When using the SPP implementation, these are expressed in the following form:

**malspp:APIDQualifier/APID/sourceId**

Below, one can find the list of services provided and their respective URIs.

GMV’s OBSW provider (on the Nanomind)

Services:

* + Power service: malspp:0/10/0
  + ExperimentWD service: malspp:0/10/0
  + GPS service: malspp: 0/10/0

NanoSat MO Supervisor provider (on the SEPP)

Services:

* + Apps Launcher: malspp:247/100/6

# Provider: GMV’s OBSW

## Service: Power

### General

The power service allows the user to control the power state of the on-board devices.

Table 5‑1: Power Service Operations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Identifier | Service Identifier | Area Number | Service Number | Area Version |
| OPSSAT\_PF | Power | 75 | 17 | 1 |
| Interaction Pattern | Operation Identifier | Operation Number | Support in Replay | Capability Set |
| SEND | [powerOffSBandTX](#_OPERATION_Power_powerOffSBandTX) | 1 | No | 1 |
| SEND | [powerOnSBandTX](#_OPERATION_Power_powerOnSBandTX) | 2 | No |
| SEND | [powerOffSBandRX](#_OPERATION_Power_powerOffSBandRX) | 3 | No |
| SEND | [powerOnSBandRX](#_OPERATION_Power_powerOnSBandRX) | 4 | No |
| SUBMIT | [setPowerState](#_OPERATION_Power_setPowerState) | 5 | No |

### OPERATION: powerOffSBandTX

#### General

The powerOffSBandTX operation powers off the S-Band transmitter.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | powerOffSBandTX | |
| Interaction Pattern | SEND | |
| Pattern Sequence | Message | Body Type |
| IN | SEND |  |

#### Structures

#### Errors

The operation cannot return any errors.

### OPERATION: powerOnSBandTX

#### General

The powerOnSBandTX operation powers on the S-Band transmitter.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | powerOnSBandTX | |
| Interaction Pattern | SEND | |
| Pattern Sequence | Message | Body Type |
| IN | SEND |  |

#### Structures

#### Errors

The operation cannot return any errors.

### OPERATION: powerOffSBandRX

#### General

The powerOffSBandRX operation powers off the S-Band receiver.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | powerOffSBandRX | |
| Interaction Pattern | SEND | |
| Pattern Sequence | Message | Body Type |
| IN | SEND |  |

#### Structures

#### Errors

The operation cannot return any errors.

### OPERATION: powerOnSBandRX

#### General

The powerOnSBandRX operation powers on the S-Band receiver.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | powerOnSBandRX | |
| Interaction Pattern | SEND | |
| Pattern Sequence | Message | Body Type |
| IN | SEND |  |

#### Structures

#### Errors

The operation cannot return any errors.

### OPERATION: setPowerState

#### General

The setPowerState sets the power state of a selected devices.

Only ground segment has full control on the power state changing (APID filtering is performed).

|  |  |  |
| --- | --- | --- |
| Operation Identifier | setPowerState | |
| Interaction Pattern | SUBMIT | |
| Pattern Sequence | Message | Body Type |
| IN | SUBMIT | devices : (List<[PayloadDevice](#_DATATYPE_PayloadDevice)>)  powerStates : (List<MAL::Boolean>) |

#### Structures

1. The 'devices' field shall contain the list of devices for which the power state change should be applied.
2. The 'powerStates' field shall contain the list of MAL::Boolean objects that corresponds to the elements in the device list. If the MAL::Boolean object on the list contains the value of 'TRUE' then the corresponding device should be powered on; otherwise it should be powered off.

#### Errors

The operation may return the following error:

##### ERROR: UNKNOWN

1. One or more of the devices specified in the list do not exist.
2. The indexes of the error values shall be contained in the extra information field.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | List<MAL::UInteger> |

## Service: ExperimentWD

### General

The experiment watchdog service is provided by the OBC and shall be used by the experiments signal that they are still running.

Table 5‑2: ExperimentWD Service Operations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Identifier | Service Identifier | Area Number | Service Number | Area Version |
| OPSSAT\_PF | ExperimentWD | 75 | 20 | 1 |
| Interaction Pattern | Operation Identifier | Operation Number | Support in Replay | Capability Set |
| SEND | [alive](#_OPERATION_ExperimentWD_alive) | 1 | No | 1 |

### OPERATION: alive

#### General

The alive operation should be called every TBD seconds by an experiment to OBC that it is still running.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | alive | |
| Interaction Pattern | SEND | |
| Pattern Sequence | Message | Body Type |
| IN | SEND |  |

#### Structures

The operation has no arguments.

#### Errors

The operation cannot return any errors.

## Service: GPS

### General

The GPS service allows the user to obtain data samples from the GPS device.

Table 5‑3: GPS Service Operations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |
| PROGRESS | [getGPSData](#_OPERATION_GPS_getGPSData) | 1 | No | 1 |

### OPERATION: getGPSData

#### General

The getGPSData operation allows a consumer to receive requested data from the GPS device (for example, GPGSA data frame).

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

#### Structures

1. The ‘command’ field shall contain a request encoded as a string to be sent to the GPS receiver.
2. The ‘data’ field, in each update or final response message, holds the GPS data frame encoded as a string of ASCII characters.

#### Errors

The operation does not return any errors.

## Area data types: OPSSAT\_PF

### ENUMERATION: PayloadDevice

PayloadDevice is an enumeration representing the possible payload devices that could be serviced.

|  |  |  |
| --- | --- | --- |
| Name | PayloadDevice | |
| Short Form Part | 2 | |
| Enumeration Value | Numerical Value | Comment |
| CCSDSEngine | 4 | CCSDS Engine |
| XBandTRX | 11 | X-Band Transmitter/Receiver |
| SBandTRX | 12 | S-Band Transmitter/Receiver |
| FineADCS | 13 | Fine ADCS |
| SDR | 14 | Software Defined Radio |
| SEPP1 | 15 | Processing Platform 1 |
| SEPP2 | 16 | Processing Platform 2 |
| HDCamera | 17 | HD Camera |
| OpticalRX | 18 | Optical Receiver |

# Provider: NanoSat MO Supervisor

## Service: AppsLauncher

### General

The Apps Launcher service provides the ability to monitor the execution, run, stop, kill and list applications software on-board of a spacecraft. The apps can be organized in categories.

The service is independent from any particular Operating System or platform.

Table 6‑: AppsLauncher Service Operations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Identifier | Service Identifier | Area Number | Service Number | Area Version |
| SoftwareManagement | AppsLauncher | 7 | 5 | 1 |
| Interaction Pattern | Operation Identifier | Operation Number | Support in Replay | Capability Set |
| PUBLISH-SUBSCRIBE | [monitorExecution](#_OPERATION_AppsLauncher_monitorExecution) | 1 | No | 1 |
| SUBMIT | [runApp](#_OPERATION_AppsLauncher_runApp) | 2 | No | 2 |
| SUBMIT | [killApp](#_OPERATION_AppsLauncher_killApp) | 3 | No |
| PROGRESS | [stopApp](#_OPERATION_AppsLauncher_stopApp) | 4 | No | 3 |
| REQUEST | [listApp](#_OPERATION_AppsLauncher_listApp) | 5 | Yes | 4 |

### High Level Requirements

1. The Apps Launcher service shall provide:
   1. the capability for periodic monitoring of the applications output;
   2. the capability for running and killing applications;
   3. the capability for stopping applications;
   4. the capability for listing the object instance identifiers for the available apps.

### Functional Requirements

### COM usage

1. An App COM object represents an on-board application. The COM object body shall hold the details of the application.
   1. The App COM object source link should point to the package from where the app was installed.

Table 6‑: AppsLauncher Service Object Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object Name | Object Number | Object Body Type | Related points to | Source points to |
| App | 1 | [AppDetails](#_DATATYPE_AppDetails) |  | PackageManagement::2 |

### COM Event Service usage

1. A StopApp COM event represents a request to stop a certain application. The COM event object body shall hold the name of the provider to be stopped.
   1. The StopApp COM event related link shall point to the App COM object.
   2. The StopApp COM event source link should point to the object that caused it to be created, most likely a COM OperationActivity object.
   3. The StopApp COM event shall be generated by the stopApp operation.
2. A Stopping COM event represents an acknowledgement that the application is stopping its execution.
   1. The Stopping COM event source link shall point to the StopApp COM event that requested the application to stop or to null if there was no request.
   2. The Stopping COM event shall be generated by the application when it is going to stop its execution.
3. A Stopped COM event represents an acknowledgement that the application is going to completely stop its execution.
   1. The Stopping COM event source link shall point to the StopApp COM event that requested the application to stop or to null if there was no request.
   2. The Stopping COM event shall be generated by the application when it is completely stopping its execution.

Table 6‑: AppsLauncher Service Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Event Name | Object Number | Object Body Type | Related points to | Source points to |
| StopApp | 2 | MAL::Identifier | 1 | COM::ActivityTracking::6 |
| Stopping | 3 | No body |  | 2 |
| Stopped | 4 | No body |  | 2 |

### COM Object Relationships

The Figure below shows the COM object and event relationships for this service:

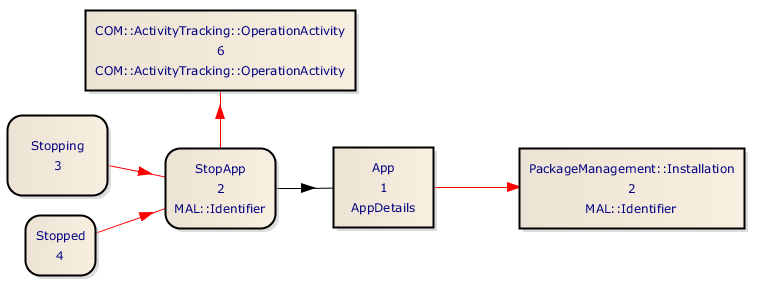


Figure 6‑: AppsLauncher Service COM object and event relationships

### OPERATION: monitorExecution

#### General

The monitorExecution operation allows a consumer to subscribe for the application execution output stream.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | monitorExecution | |
| Interaction Pattern | PUBLISH-SUBSCRIBE | |
| Pattern Sequence | Message | Body Type |
| OUT | PUBLISH/NOTIFY | outputStream : (MAL::String) |

#### Structures

1. The outputStream field shall contain a stream of characters corresponding to the output stream of the application.
2. The MAL EntityKey.firstSubKey shall contain the App name.
3. The MAL EntityKey.secondSubKey shall contain the AppDetails object instance identifier.
4. The MAL EntityKey.thirdSubKey shall be NULL.
5. The MAL EntityKey.fourthSubKey shall be NULL.
6. The timestamp of the update shall be the on-board time when the update was published.
7. The publish message shall include the ObjectId of the source link of the update.
8. If no source link is needed then the ObjectId shall be replaced with a NULL.

#### Errors

The operation does not return any errors.

### OPERATION: runApp

#### General

The runApp operation allows a consumer to run an application on the provider.

An object instance identifier is returned for further monitoring of the application execution in the monitorExecution operation.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | runApp | |
| Interaction Pattern | SUBMIT | |
| Pattern Sequence | Message | Body Type |
| IN | SUBMIT | appInstIds : (List<MAL::Long>) |

#### Structures

1. The appInstIds field contains the list of apps to run.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

1. One or more of the requested apps to run is unknown.
2. A list of the indexes of the error values shall be contained in the extra information field.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | List<MAL::UInteger> |

##### ERROR: INVALID

1. One or more of the requested apps is already running.
2. A list of the indexes of the error values shall be contained in the extra information field.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID | Defined in COM | List<MAL::UInteger> |

##### ERROR: INTERNAL

1. The process of the app could not be started.
2. A list of the indexes of the error values shall be contained in the extra information field.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INTERNAL | Defined in MAL | List<MAL::UInteger> |

### OPERATION: killApp

#### General

The killApp operation allows a consumer to kill the execution of an application on the provider in case an application becomes unresponsive.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | killApp | |
| Interaction Pattern | SUBMIT | |
| Pattern Sequence | Message | Body Type |
| IN | SUBMIT | appInstIds : (List<MAL::Long>) |

#### Structures

1. The appInstIds field contains the list of apps to be killed.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

1. One or more of the requested apps to be killed is unknown.
2. A list of the indexes of the error values shall be contained in the extra information field.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | List<MAL::UInteger> |

##### ERROR: INVALID

1. One or more of the requested apps is not running.
2. A list of the indexes of the error values shall be contained in the extra information field.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID | Defined in COM | List<MAL::UInteger> |

### OPERATION: stopApp

#### General

The stopApp operation allows a consumer to stop the execution of an application on the provider.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | stopApp | |
| Interaction Pattern | PROGRESS | |
| Pattern Sequence | Message | Body Type |
| IN | PROGRESS | appInstIds : (List<MAL::Long>) |
| OUT | ACK |  |
| OUT | UPDATE | appClosing : (MAL::Long) |
| OUT | RESPONSE |  |

#### Structures

1. The appInstIds field contains the list of apps to stop.
2. The appClosing field shall contain the object instance identifier of an app. This update shall be sent after the app acknowledges the reception of the command to stop.

#### Errors

The operation does not return any errors.

### OPERATION: listApp

#### General

The listApp operation allows a consumer to request the object instance identifiers of the Apps objects and running status for an app name or for a certain app category.

|  |  |  |
| --- | --- | --- |
| Operation Identifier | listApp | |
| Interaction Pattern | REQUEST | |
| Pattern Sequence | Message | Body Type |
| IN | REQUEST | appNames : (List<MAL::Identifier>)  category : (MAL::Identifier) |
| OUT | RESPONSE | appInstIds : (List<MAL::Long>)  running : (List<MAL::Boolean>) |

#### Structures

1. The appNames field contains a list of application names.
2. The category field contains the category identifier to filter on.
3. The appInstIds field contains a list of apps.
4. The running field contains a list of boolean values with the information about thte running status of requested apps.
5. The returned lists shall maintain the same order as the submitted list unless the wildcard value was included in the appNames field request.

#### Errors

The operation may return the following error:

##### ERROR: UNKNOWN

1. One or more of the requested apps to run is unknown.
2. A list of the indexes of the error values shall be contained in the extra information field.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | List<MAL::UInteger> |

## Service data types: AppsLauncher

### Composite: AppDetails

The AppDetails structure holds the details of an instance of an app.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | AppDetails | | |
| Extends | MAL::Composite | | |
| Short Form Part | 1 | | |
| Field | Type | Nullable | Comment |
| name | MAL::Identifier | No | The name of the app. Must not be empty or wildcard value. |
| description | MAL::String | No | The description of the app. |
| version | MAL::String | No | The version of the app. |
| category | MAL::Identifier | No | The category of the app. |
| runAtStartup | MAL::Boolean | No | Controls whether the app runs at startup. |
| running | MAL::Boolean | No | The current running state of the app. |
| extraInfo | MAL::String | Yes | Additional information that might be implementation-specific. |
| copyright | MAL::String | Yes | The copyright of the app. |
| iconPath | MAL::String | Yes | The icon location path of the app. It can be either to a remote link or to a local file. |

# Examples

## Starting an App from the Nanomind

The example below is a CAN message exchanged on the bus to start an App:

can0 1085674F [8] 18 64 C0 11 00 59 01 00 '.d...Y..'

can0 108D474F [8] 07 00 05 00 02 01 20 02 '...... .'

can0 108D274F [8] 00 F7 30 21 76 D9 96 4E '..0!v..N'

can0 108D074F [8] 00 12 D2 07 06 6F 62 41 '.....obA'

can0 108CE74F [8] 9E 2B 02 0C 00 00 00 03 '.+......'

can0 108CC74F [8] 01 00 03 65 73 61 01 00 '...esa..'

can0 108CA74F [8] 07 4F 50 53 2D 53 41 54 '.OPS-SAT'

can0 108C874F [8] 01 00 15 4E 61 6E 6F 53 '...NanoS'

can0 108C674F [8] 61 74 5F 4D 4F 5F 53 75 'at\_MO\_Su'

can0 108C474F [8] 70 65 72 76 69 73 6F 72 'pervisor'

can0 108C274F [8] 01 00 00 00 01 01 **00 00** '........'

can0 1088074F [8] **00 00 00 00 04 0C** **09 EC** '........'

can0 19249E1C [8] 18 02 C0 28 00 1F 02 00 '...(....'

can0 192C7E1C [8] 07 00 05 00 02 01 20 64 '...... d'

can0 192C5E1C [8] 00 F7 30 21 76 D9 96 4E '..0!v..N'

can0 192C3E1C [8] 00 12 D0 06 07 6F 62 41 '.....obA'

can0 19281E1C [6] 9E 0D 4F DF 23 3B '..O.#;'

It can be seen that the consumer&provider exchanged 2 messages:

1st : From the Consumer to the Provider (SUBMIT)

2nd : An ACK from the Provider to the consumer (ACK)

This is to fulfil the SUBMIT interaction pattern defined by the MAL.

On the example above, an app with APID=1036 (in hex: 04 0C) was used. The octets in red on the message above must be set to the APID value of the App to be launched.

There are 8 bytes to select because it corresponds to a Long MAL Data Type which is the object instance identifier on the Apps Launcher service. For OPS-SAT, the object instance identifier matches the APID of the application.

The octets in green correspond to the CRC of the SPP and must be recalculated depending on the content of the message according to GMV’s OBSW ICD.

## Stopping an App from the Nanomind

The example below is a CAN message exchanged on the bus to stop an App:

can0 10856C17 [8] 18 64 C0 12 00 59 08 00 '.d...Y..'

can0 108D4C17 [8] 07 00 05 00 04 01 20 02 '...... .'

can0 108D2C17 [8] 00 F7 30 21 8D F4 56 2C '..0!..V,'

can0 108D0C17 [8] 00 13 D2 07 06 6F 62 47 '.....obG'

can0 108CEC17 [8] BC A0 C4 9C 00 00 00 03 '........'

can0 108CCC17 [8] 01 00 03 65 73 61 01 00 '...esa..'

can0 108CAC17 [8] 07 4F 50 53 2D 53 41 54 '.OPS-SAT'

can0 108C8C17 [8] 01 00 15 4E 61 6E 6F 53 '...NanoS'

can0 108C6C17 [8] 61 74 5F 4D 4F 5F 53 75 'at\_MO\_Su'

can0 108C4C17 [8] 70 65 72 76 69 73 6F 72 'pervisor'

can0 108C2C17 [8] 01 00 00 00 01 01 **00 00** '........'

can0 10880C17 [8] **00 00 00 00 04 0C** **82 E2** '........'

can0 192482BE [8] 18 02 C0 4A 00 1F 09 00 '...J....'

can0 192C62BE [8] 07 00 05 00 04 01 20 64 '...... d'

can0 192C42BE [8] 00 F7 30 21 8D F4 56 2C '..0!..V,'

can0 192C22BE [8] 00 13 D0 06 07 6F 62 47 '.....obG'

can0 192802BE [6] BC 7E F9 DB BF 52 '.~...R'

On the example above, an app with APID=1036 (in hex: 04 0C) was used. The octets in red on the message above must be set to the APID value of the App to be launched.

There are 8 bytes to select because it corresponds to a Long MAL Data Type which is the object instance identifier on the Apps Launcher service. For OPS-SAT, the object instance identifier matches the APID of the application.

The octets in green correspond to the CRC of the SPP and must be recalculated depending on the content of the message according to GMV’s OBSW ICD.

## Killing an App from the Nanomind

The example below is a CAN message exchanged on the bus to kill an App:

can0 108564CE [8] 18 64 C0 18 00 59 01 00 '.d...Y..'

can0 108D44CE [8] 07 00 05 00 03 01 20 02 '...... .'

can0 108D24CE [8] 00 F7 30 8E 27 75 59 07 '..0.'uY.'

can0 108D04CE [8] 00 19 D2 07 06 6F 7E 15 '.....o~.'

can0 108CE4CE [8] 10 4B 02 0C 00 00 00 03 '.K......'

can0 108CC4CE [8] 01 00 03 65 73 61 01 00 '...esa..'

can0 108CA4CE [8] 07 4F 50 53 2D 53 41 54 '.OPS-SAT'

can0 108C84CE [8] 01 00 15 4E 61 6E 6F 53 '...NanoS'

can0 108C64CE [8] 61 74 5F 4D 4F 5F 53 75 'at\_MO\_Su'

can0 108C44CE [8] 70 65 72 76 69 73 6F 72 'pervisor'

can0 108C24CE [8] 01 00 00 00 01 01 **00 00** '........'

can0 108804CE [8] **00 00 00 00 04 0C** **05 23** '.......#'

On the example above, an app with APID=1036 (in hex: 04 0C) was used. The octets in red on the message above must be set to the APID value of the App to be killed.

There are 8 bytes to select because it corresponds to a Long MAL Data Type which is the object instance identifier on the Apps Launcher service. For OPS-SAT, the object instance identifier matches the APID of the application.

The octets in green correspond to the CRC of the SPP and must be recalculated depending on the content of the message according to GMV’s OBSW ICD.