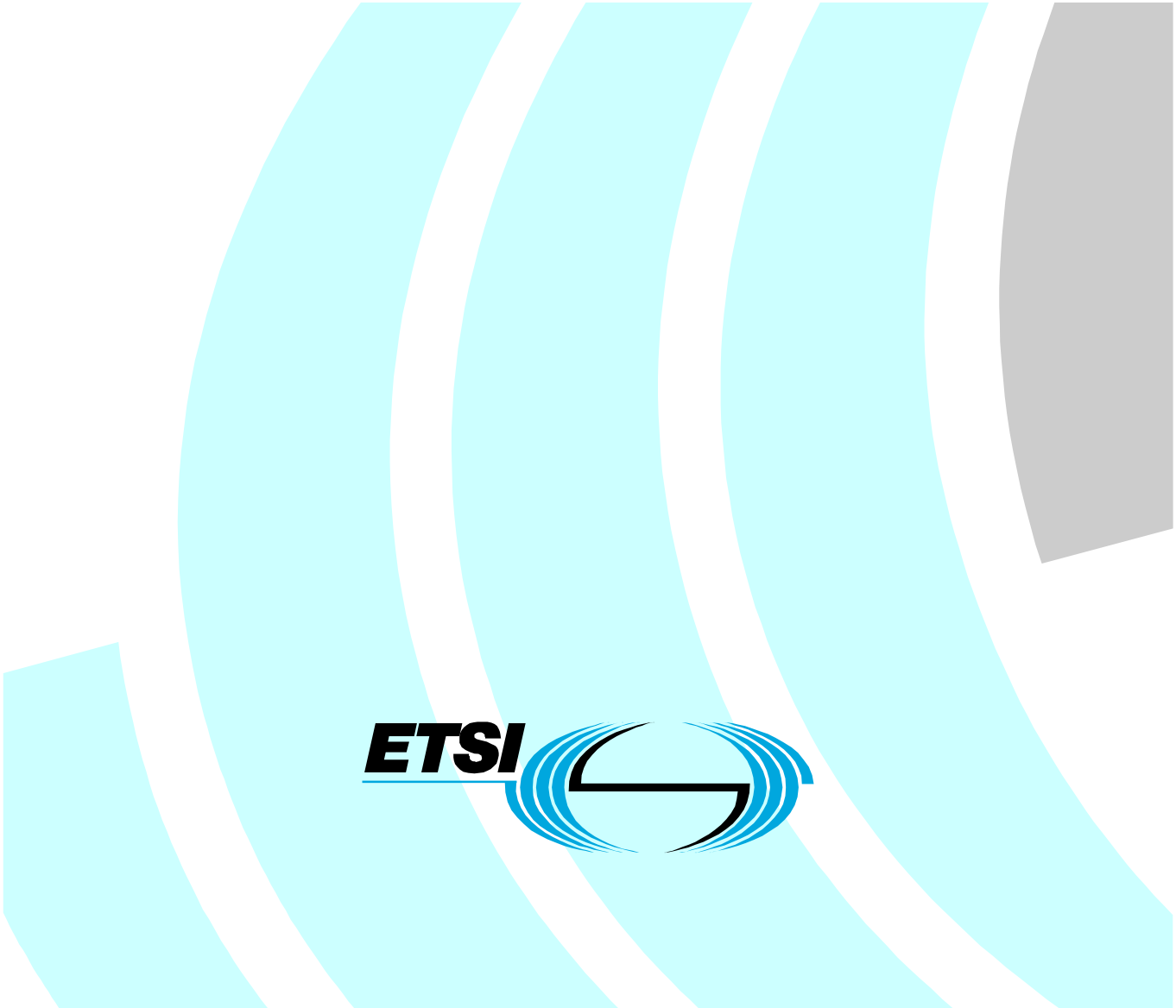
ETSI EN 301 927 V1.1.1 (2003-02)

European Standard (Telecommunications series)

Satellite Earth Stations and Systems (SES); European Co-operation for Space Standardization (ECSS); Satellite Software Data Handling Interfaces (SSDHI)



Reference
DEN/SES-000-ECSS-2

Keywords
data, interface, LAYER 7, satellite, service

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.org

Copyright Notification

Contents

Intellectual Property Rights 5
Foreword 5
Introduction 5
1 Scope 6
2 References 6
3 Definitions and abbreviations 6
3.1 Definitions 6
3.2 Abbreviations 8
4 Applicability 8
5 Recommendation guidelines 9
5.1 Objectives 9
5.2 Interface covering 9
5.3 Position of the standard proposed services with respect to other standards 10
6 Service specification 12
6.1 Introduction 12
6.2 Conventions 12
6.2.1 Service primitive naming 12
6.2.2 Parameter type and format abbreviations 13
6.3 Standard services 13
7 Services 16
7.1 Memory Management Service 16
7.2 Units power switching and resetting services 18
7.2.1 Device power switch-on service 18
7.2.2 Device power switch-off service 18
7.2.3 Device reset service 19
7.2.4 Device arming service 19
7.2.5 Function enabling service 20
7.3 Specific services 21
7.3.1 Platform Sensor and Actuator services 21
7.3.1.1 Specific sensors 21
7.3.1.1.1 Accelerometer service 21
7.3.1.1.2 Gyro service 22
7.3.1.1.3 Star tracker service 23
7.3.1.1.4 Earth sensor service 24
7.3.1.1.5 Sun Sensor service 26
7.3.1.1.6 Magnetometer service 26
7.3.1.2 Specific actuator 27
7.3.1.2.1 Thruster configuration service 27
7.3.1.2.2 Venting service 27
7.3.1.2.3 Apogee Boost Motor service 28
7.3.1.2.4 Thruster burst service 29
7.3.1.2.5 Plasmic Propulsion service 30
7.3.1.2.6 Wheel service 31
7.3.1.2.7 Pyrotechnic service 32
7.3.1.2.8 Magnetic-torquers service 34
7.3.1.3 Platform management 34
7.3.1.3.1 Battery reconditioning service 34
7.3.1.3.2 Power conditioning service 36
7.3.1.3.3 Stepper motor management service 37
7.3.2 Payload & TTC services 38
7.3.2.1 Synthesizers service 38

7.3.2.2 Power amplifiers service 39
7.3.2.3 Channel amplifier 40
7.3.2.4 Switching service 41
7.3.2.5 Digital Transparent Processor service 42
7.4 Simple devices services 45
7.4.1 Pulse command service 45
7.4.2 Temperature acquisition service 46
7.4.3 Pressure acquisition service 46
7.4.4 Current acquisition service 47
7.4.5 Voltage acquisition service 48
7.4.6 Frequency acquisition service 48

8 Design requirements 49

Annex A (informative): End to end example 50

Annex B (informative): Comments and recommendations for usage of the present document 51

B.1 Identifier type definition 51

B.2 Distinction between status and error parameters 51

B.3 Recommendations on services 52

B.4 SOIF services not kept in the present document 52

B.5 Services not defined in the present document 52

Annex C (informative): SOIF description 54

C.1 General 54

C.2 SOIF Command and Data Acquisition Service 55

C.3 Architectural Context 55

C.4 Service Operation - Service User Perspective 56

C.5 Service Operation - Service Providers Perspective 56

Annex D (informative): Parameter type code and format 57

D.1 Encoding formats of parameter types 57

D.2 Parameter type definitions 57
D.2.1 Boolean 57
D.2.2 Enumerated Parameter 57
D.2.3 Unsigned Integer 58
D.2.4 Signed Integer 58
D.2.5 Real 58
D.2.6 Bit-String 59
D.2.7 Octet-String 60

Annex E (informative): Bibliography 61

History 62

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

All published ETSI deliverables shall include information which directs the reader to the above source of information.

Foreword

This European Standard (Telecommunications series) has been produced by ETSI Technical Committee Satellite Earth Stations and Systems (SES).

National transposition dates	
Date of adoption of this EN:	21 February 2003
Date of latest announcement of this EN (doa):	31 May 2003
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	30 November 2003
Date of withdrawal of any conflicting National Standard (dow):	30 November 2003

Introduction

Shorter and shorter communications satellites development cycles and compatibility requested with off the shelf equipments requires interface standardization to be able to introduce a new equipment very late in satellite development cycle. The present document will allow the development of On Board Software (OBSW) independently of the choice of satellite equipment. It will be used to standardize the software specifications of the Interface between the application software and the communication services. It shall help the standardization of equipment interface.

The present document addresses ISO OSI application layer services [1].

Currently, no application layer services standard exists for the Data System of geostationary communication satellites. The aim of the present document is to respond to such requirements. Nevertheless, there are complementary agency standards existing or in progress (see bibliography).

Integration or adaptation of new services, due to technology evolution, will be implemented through the maintenance of the present document. The contents of the present document are subject to continuing work within TC-SES and may change following formal TC-SES approval. Should TC-SES modify the contents of the present document it will then be republished by ETSI with an identifying change of release date and an increase in version number as follows:

Version 1.m.n, where:

- m The second digit (m) is incremented for all other types of changes, i.e. technical enhancements, corrections, updates, etc.
- n The third digit (n) is incremented when editorial only changes have been incorporated in the specification.

1 Scope

The present document applies to Geostationary Communications Satellite architectures based on ISO OSI Reference Model or SOIF Reference Model, but could also be applied to other types of satellites.

The present document sets out the minimum definition, services and interfaces requirements of the satellite communication application layer.

The present document is complementary to Spacecraft Onboard Interfaces (SOIF).

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ISO/IEC 7498-1 (1994): "Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model".

[2] IEEE Standard 754-1985: "IEEE Standard for Binary Floating-Point Arithmetic".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Application Layer Service (ALS): entity which provides application software with a capability to operate devices, with a warranty of performance

application software: on-board software implementing the satellite control functions

NOTE: A service is implemented by hardware and/or software. Application layer services for the control of devices are defined in terms of:

- the functions provided by the service;
- a service access point;
- the management parameters.

autonomy: ability of a system to provide without external intervention, mission services on a given period of time and in a limited context: nominal or under anomaly conditions

NOTE: Three types of autonomy are defined:

- **reflex autonomy:** characterizes the capability to perform predefined actions under triggering events. This kind of autonomy, already in use, is performed through automation mechanisms, with predefined sequences of states and triggering events (failure detection, isolation and recovery, etc.).
- **function autonomy:** characterizes the capability of the satellite to perform automatic function on board, implementing on board closed loops to maintain function performances without ground intervention. Thermal regulation, batteries management, and attitude control are some examples of function autonomy.
- **decision autonomy:** characterizes the capability of the satellite to manage and update its schedule of activities or tasks, taking into account events (expected or not) occurring during the mission (Autonomous navigation, mission (re)scheduling, etc.).

Data Handling System (DHS): on-board system including computers and interface units hosting the application software and providing communications between on-board units and with the ground

devices: devices are the onboard components that onboard application software control in order to perform the operational objectives of the spacecraft

EXAMPLE: Sensors and actuators, Telemetry Command and Ranging (TCR) devices or other mission specific devices.

J2000: earth centred co-ordinate reference system with an epoch of January 1st 2000

primitive: a signal (e.g. a send request) which is passed across the service access point in order to use the service capabilities

NOTE: The primitive may have associated parameters.

Service Access Point (SAP): the interface provided to a service through which users access the capabilities of the service

NOTE: The service access points are defined in terms of a set of primitives.

spacelink interface: ground to satellite communication link for monitoring and control

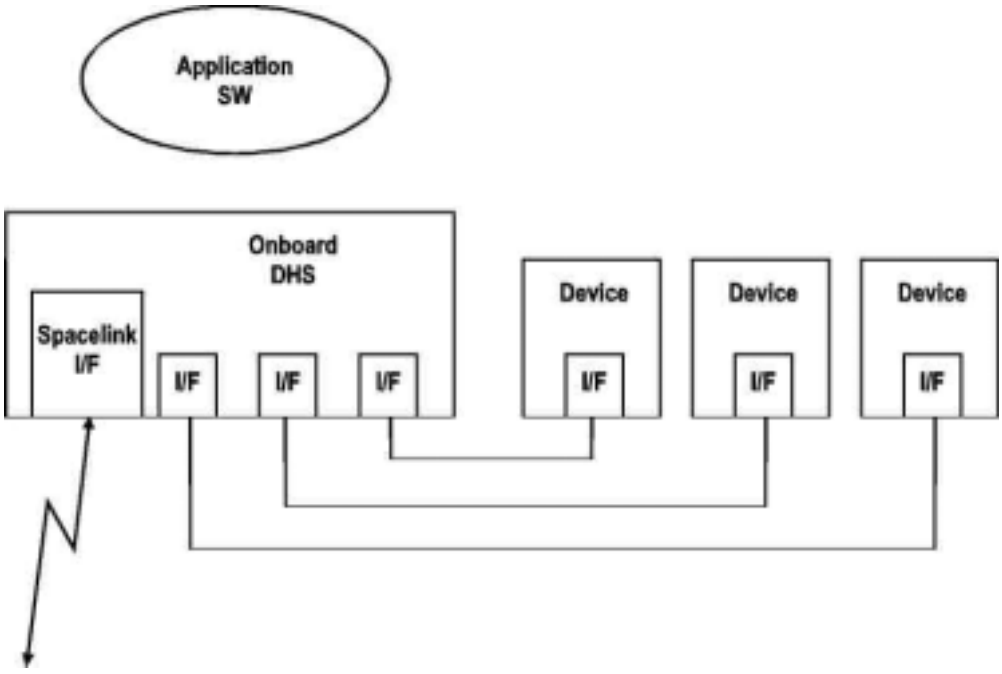


Figure 1: Example of Spacecraft System

Service: Application Layer Service

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ABM	Apogee Boost Motor
ACC	ACCelerometer
ALSS	Application Layer Services Standard
BSV	Bi Stable Valve
CAMP	Channel AMPlifier
DHS	Data Handling System
DTP	Digital Transparent Processor
EEPROM	Electrically Erasable Programmable Read Only Memory
EPC	Encapsulated Power Converter
ES	Earth Sensor
EUC	Engineering Unit Conversion
FDIR	Failure Detection Isolation and Recovery
GYRO	GYROscope
HW	HardWare
ICD	Interface Control Document
ISO	International Standard Organization
MSV	Mono Stable Valve
MT	Magneto Torquer
OBSW	On-Board SoftWare
OSI	Open System Interconnection
PCU	Power Conditioning Unit
PFC	Parameter Format Code
PP	Plasmic Propulsion
PTC	Parameter Type Code
PUS	Packet Utilization Standard
PYRO	PYROtechnic
RAM	Random Access Memory
ROM	Read Only Memory
RW	Reaction Wheel
SAP	Service Access Point
SDB	System Data Base
SM	Stepper Motor
SOIF	Spacecraft Onboard InterFaces
SS	Sun Sensor
SSDHI	Satellite Software Data Handling Interfaces
STR	Star TRacker
SW	SoftWare
TC	Tele Command
TCR	Telemetry, Command and Ranging
TOM	Thruster Orientation Mechanism
TWTA	Travelling Wave Tube Amplifier

4 Applicability

The present document can be applied to all classes of spacecraft including in particular communications satellites.

5 Recommendation guidelines

5.1 Objectives

The main recommendation drivers are:

- Development optimization:
  - Standardization data format.
  - Using common formats for key commands across different data buses.
  - Planning optimization by minimising development time.
  - Cost by reducing engineering process on specification and development of application layer services.
  - Minimising risk basing the development on validated requirements.
  - Increasing potential for flight equipment, software (SW) and test equipment reuse.
  - Overlapping of software development with hardware development:
    - It will be possible to optimize this process by applying the present document which provides requirements independent of any hardware configuration.
    - This concept allows to freeze application layer Interface Control Document (ICD) to guarantee hardware independence.
- Performance constraints:
  - Real time.
  - Memory size.
  - Load of onboard communication data bus.
  - Device Observability.
  - Device Commandability.
  - Number and type of units to interface.
  - Operations complexity.

5.2 Interface covering

Data communication on satellite requires electrical and data interfaces onboard. These interfaces include:

- Hardware interfaces:
  - Electrical interfaces (power, command, acquisition, pyrotechnic, etc.) to devices.
  - Onboard communication data buses.
- Software interfaces:
  - Software drivers controlling devices and communication on data buses.
  - Application software interfaces.

The application layer is the highest layer of the reference OSI [1] and SOIF (see bibliography) models (see figure 2). In the OSI Reference Model, the application software is not included in the reference model application layer, since this software is the user of the communications services provided by the OSI Reference Model.

The present document provides recommendations to interface application services between themselves and to other OSI/SOIF layers. In particular, it defines the services to interface devices with application software independent of lower layer levels (particularly the electrical interface).

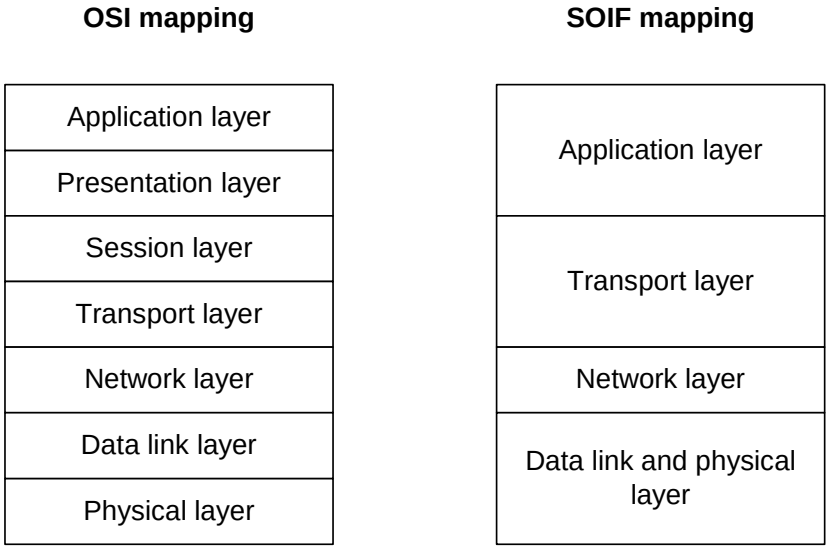


Figure 2: OSI and SOIF layers mapping

5.3 Position of the standard proposed services with respect to other standards

The present document implies the provision of certain underlying services, that could be provided by SOIF (see annex C):

- Command and Data Acquisition service;
- Time service;
- File transfer service.

The present document does not address in particular:

- Task management;
- On-board scheduling;
- Satellite System FDIR.

User Application Layer definition:

This layer contains satellite application services like AOCS management, Control and Data management and FDIR management, unit management, etc. This layer shall be independent of the underneath layers in term of equipment manufacturer and transmission protocol/media. The goal is to provide user application reuse capability.

User application services implementation can call (see figure 3):

- Other user application layer services;
- Application Interface services;
- SOIF or non SOIF application layer services (see annex C);
- Any other lower layer services with exposed Service Access Point.

Application Interface Services definition:

These services provide satellite function or equipment oriented services with an interface independent of the equipment manufacturer or of the transmission protocol/physical link.

The service implementation requires the equipment database in order to make the link between the standardized interface and the equipment function.

Although the service implementation is equipment dependent, the objective is to use common services provided by e.g. the SOIF application layer.

These services can call:

- Other application interface services;
- SOIF or non SOIF application layer services;
- Any other lower layer services with exposed Service Access Point.

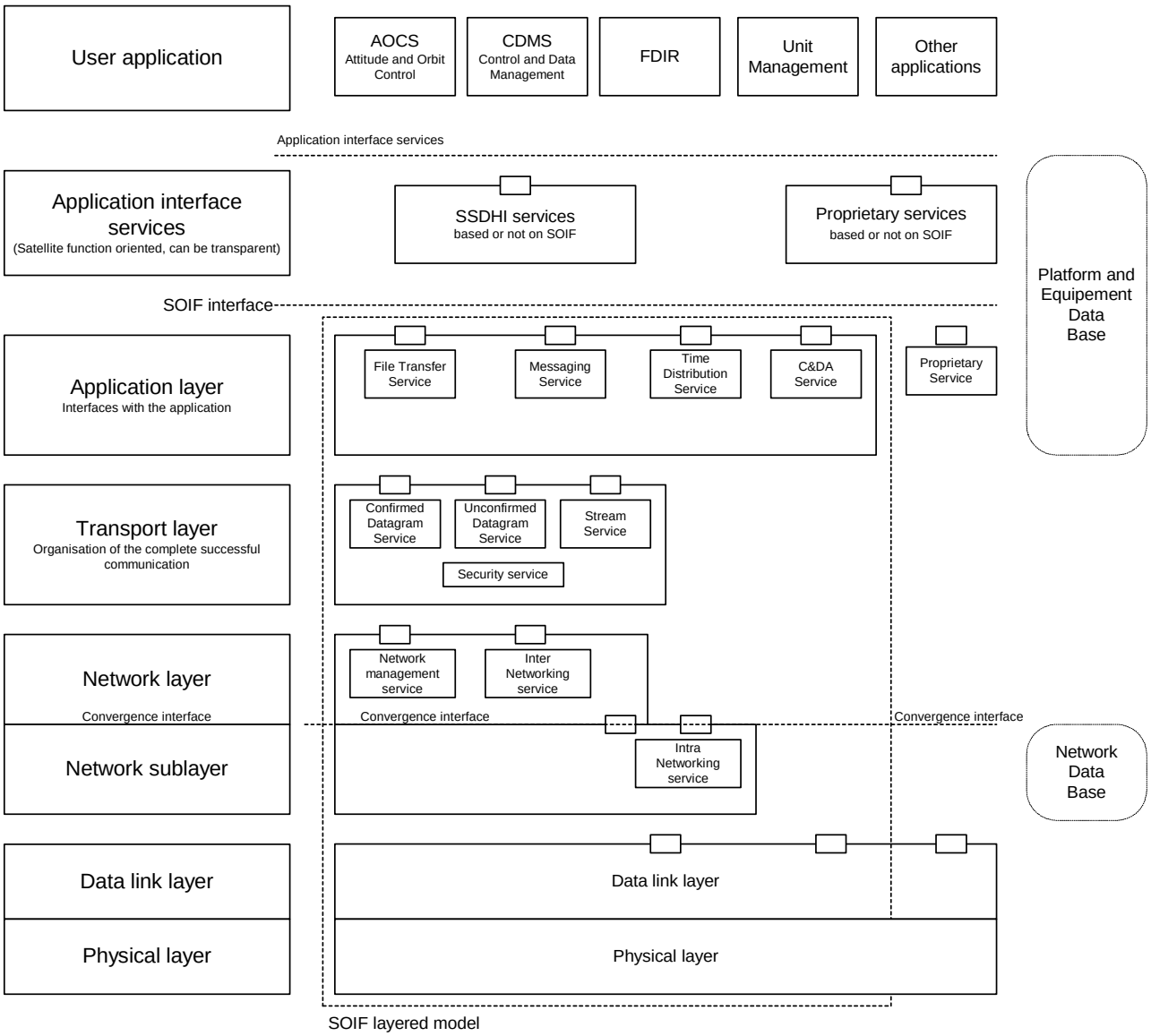


Figure 3: SSDHI position with respect to. other services

6 Service specification

6.1 Introduction

This service specification covers all onboard satellite applications.

Generic services are provided according to onboard satellite complexity increase: memory management, etc.

Specific services cover Platform and Payload devices. These services are equipment functionality dependent. They are defined according to the state of the art.

Evolution of these services will be done according to the integration of new equipment.

Simple device services are used for low level command (Pulse) or low level acquisition (Pressure, Temperature, Current, Voltage, Frequency, etc.).

For raw data acquisition it is recommended to use directly Command and Acquisition services, or eventually File Transfer provided by SOIF, or proprietary services.

Units redundancy concepts and management are performed in the User Application Layer, and thus it is not addressed in the present document.

6.2 Conventions

6.2.1 Service primitive naming

Inside a service one or more primitive can be defined.

These primitives provide data exchange between the application software and the service.

The following naming convention suffixes are used according to the transaction direction:

- XXXXX_RQ: Request primitive issued by the service user to the service provider.
- XXXXX_IND: Indication primitive issued by the service provider to the service user. This primitive may be associated to a previous request primitive.

The following naming convention prefixes can be used for a better primitive understanding:

- SET: Prefix for a configuration primitive: equipment configuration setting, command execution, device switch on or off.
- GET: Prefix for an acquisition primitive: data acquisition, status acquisition.

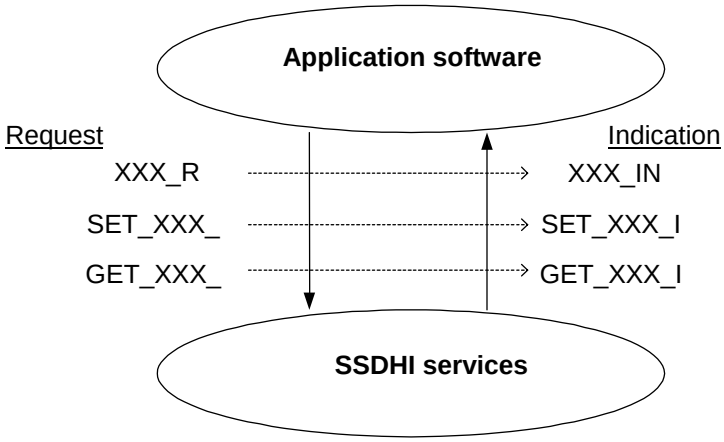


Figure 4: Primitives naming

6.2.2 Parameter type and format abbreviations

Parameter types are those defined in the PUS Packet Utilization Standard (see bibliography). A summary is given in annex D.

Within the specification of a service, the following abbreviations are used.

Table 1: Parameter type

Parameter type	Abbreviation
Boolean parameter	'Boolean'
Enumerated parameter	'Enumerated'
Integer parameter Unsigned Integer Signed Integer	 'Unsigned Integer' 'Signed Integer'
Real Parameter	'Real'
Bitstring Parameter Fixed length bit string Variable length bit string	 'Fixed BitString' 'Variable BitString'
OctetString Parameter Fixed length octet string Variable length octet string	 'Fixed OctetString' 'Variable OctetString'

For each parameter type (PTC) the present document specifies the parameter type (PTC) and format (PFC).

6.3 Standard services

The standard services listed in table 2 are described in the following clauses.

Table 2: SSDHI standard services

Service name/Primitive name	Request and Indication primitives
Memory Management Service	
Load memory using base plus offset	LOAD_MEMORY_BO_RQ LOAD_MEMORY_BO_IND
Dump memory using base plus offset	DUMP_MEMORY_BO_RQ DUMP_MEMORY_BO_IND
Check memory using base plus offset	CHECK_MEMORY_BO_RQ CHECK_MEMORY_BO_IND
Device power switch on service	
Device power switch ON	SET_DEVICE_POWER_ON_RQ SET_DEVICE_POWER_ON_IND
Device power switch off service	
Device power switch OFF	SET_DEVICE_POWER_OFF_RQ SET_DEVICE_POWER_OFF_IND
Device reset service	
Device reset	SET_DEVICE_RESET_RQ SET_DEVICE_RESET_IND
Device arming service	
Device arming	SET_DEVICE_ARMING_RQ SET_DEVICE_ARMING_IND
Device disarming	SET_DEVICE_DISARMING_RQ SET_DEVICE_DISARMING_IND
Function enabling service	
Device function enabling	SET_DEVICE_FUNCTION_ENABLE_RQ SET_DEVICE_FUNCTION_ENABLE_IND
Device function disabling	SET_DEVICE_FUNCTION_DISABLE_RQ SET_DEVICE_FUNCTION_DISABLE_IND
Accelerometer service	
Incremental speed acquisition	GET_ACC_INC_RQ GET_ACC_INC_IND

Service name/Primitive name	Request and Indication primitives
Gyroscope service	
Incremental delta angle acquisition	GET_GYRO_ANG_RQ GET_GYRO_ANG_IND
Star Tracker service	
Quaternion acquisition	GET_STR_QUAT_RQ GET_STR_QUAT_IND
Earth sensor service	
Attitude data acquisition	GET_ES_DATA_RQ GET_ES_DATA_IND
Acquisition mode selection	SET_ES_MODE_RQ SET_ES_MODE_IND
Array inhibition	SET_ES_ARRAY_RQ SET_ES_ARRAY_IND
Sun sensor service	
Sun sensor data acquisition	GET_SUN_SENSOR_DATA_RQ GET_SUN_SENSOR_DATA_IND
Magnetometer service	
Magnetometer data acquisition	GET_MAGNETOMETER_RQ GET_MAGNETOMETER_IND
Thruster configuration service	
Thruster BSV valve setting	SET_THRUSTER_BSV_VALVE_RQ SET_THRUSTER_BSV_VALVE_IND
Venting service	
Start venting	SET_VENTING_START_RQ SET_VENTING_START_IND
Stop venting	SET_VENTING_STOP_RQ SET_VENTING_STOP_IND
Apogee boost motor service	
ABM switch On	SET_ABM_ON_RQ SET_ABM_ON_IND
ABM switch Off	SET_ABM_OFF_RQ SET_ABM_OFF_IND
Thruster burst service	
Thruster MSV command	SET_THRUSTER_MSV_VALVE_RQ SET_THRUSTER_MSV_VALVE_IND
Plasmic propulsion service	
Plasmic propulsion start	SET_PP_START_RQ SET_PP_START_IND
Plasmic propulsion stop	SET_PP_STOP_RQ SET_PP_STOP_IND
Wheel service	
Speed or torque command	SET_RW_RQ SET_RW_IND
Speed acquisition	GET_RW_RQ GET_RW_IND
Pyrotechnic service	
Pyro selection	SET_PYRO_SELECT_RQ SET_PYRO_SELECT_IND
Pyro deselecting	SET_PYRO_DESELECT_RQ SET_PYRO_DESELECT_IND
Pyro firing	SET_PYRO_FIRING_RQ SET_PYRO_FIRING_IND
Magnetic torquers service	
Torquer current command	SET_MT_CURRENT_RQ SET_MT_CURRENT_IND
Battery reconditioning service	
Battery start reconditioning	SET_BATT_RECOND_START_RQ SET_BATT_RECOND_START_IND
Battery stop reconditioning	SET_BATT_RECOND_STOP_RQ SET_BATT_RECOND_STOP_IND
Battery reconditioning load change	SET_BATT_RECOND_LOAD_RQ SET_BATT_RECOND_LOAD_IND

Service name/Primitive name	Request and Indication primitives
Power conditioning service	
Charge current intensity setting	SET_PCU_CHARGE_INT_RQ SET_PCU_CHARGE_INT_IND
Stepper motor management service	
Stepper motor motion command	SET_SM_MOTION_RQ SET_SM_MOTION_IND
Stepper motor reference position setting	SET_SM_REFERENCE_RQ SET_SM_REFERENCE_IND
Stepper motor position acquisition	GET_SM_SENSOR_RQ GET_SM_SENSOR_IND
Synthesizer service	
Frequency setting	SET_FREQUENCY_RQ SET_FREQUENCY_IND
Power amplifier service	
TWTA autorestart setting	SET_TWTA_AUTORESTART_RQ SET_TWTA_AUTORESTART_IND
TWTA status acquisition	GET_TWTA_STATUS_RQ GET_TWTA_STATUS_IND
Channel amplifier service	
CAMP configuration setting	SET_CAMP_CONFIG_RQ SET_CAMP_CONFIG_IND
Switching service	
Switch configuration setting	SET_SWITCH_CONFIG_RQ SET_SWITCH_CONFIG_IND
Switch configuration acquisition	GET_SWITCH_CONFIG_RQ GET_SWITCH_CONFIG_IND
DTP service	
Links setting ON	SET_DTP_LINK_ON_RQ SET_DTP_LINK_ON_RQ
Links setting OFF	SET_DTP_LINK_OFF_RQ SET_DTP_LINK_OFF_RQ
DTP link configuration setting	SET_DTP_CONFIG_RQ SET_DTP_CONFIG_IND
DTP link power acquisition	GET_DTP_POWER_RQ GET_DTP_POWER_IND
DTP link configuration acquisition	GET_DTP_CONFIG_RQ GET_DTP_CONFIG_IND
Pulse command service	
Pulse command	SET_PULSE_RQ SET_PULSE_IND
Temperature acquisition service	
Temperature acquisition	GET_TEMPERATURE_RQ GET_TEMPERATURE_IND
Pressure acquisition service	
Pressure acquisition	GET_PRESSURE_RQ GET_PRESSURE_IND
Current acquisition service	
Current acquisition	GET_CURRENT_RQ GET_CURRENT_IND
Voltage acquisition service	
Voltage acquisition	GET_VOLTAGE_RQ GET_VOLTAGE_IND
Frequency acquisition service	
Frequency acquisition	GET_FREQUENCY_RQ GET_FREQUENCY_IND

7 Services

7.1 Memory Management Service

This service is based on PUS Memory Management Service (see Bibliography).

This service shall be use for onboard memory block management (RAM, ROM, EEPROM, etc.). This service shall provide the capability for loading, dumping and checking the contents of contiguous memory blocks.

The PUS non contiguous memory block management is not defined in this service.

The following primitives are defined:

- Load memory using base plus offset;
- Dump memory using base plus offset;
- Check memory using base plus offset.

Load memory using base plus offset:

This primitive shall be used to load data into a memory block defined by means of a base reference plus offset.

LOAD_MEMORY_BO_RQ(Device_Id, Base, Offset, Data)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)
The device_Id identifies the destination memory block according to satellite architecture.

Base = Base reference, Type = Unsigned Integer (PFC = 14)
This base shall be used as the zero reference for the offset.

Offset = Offset from the base reference, Type = Signed Integer (PFC = 14)

Data = Data block to load, Type = Variable OctetString (PFC = 0)

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- calculate the checksum (ISO standard 16 bits checksum cf annex 3 PUS)
- load the data block according to the transmitted base and offset
- read the loaded block and compare the checksum with the previous one

LOAD_MEMORY_BO_IND (Device_Id, Checksum, Error)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Checksum = Checksum status
1 = Checksum error or loading fail
0 = Checksum OK

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Dump memory using base plus offset:

This primitive shall be used to dump data from a memory block defined by means of a base reference plus offset.

DUMP_MEMORY_BO_RQ(Device_Id, Base, Offset, Length)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)
The device_Id identifies the source memory block according to satellite architecture.

Base = Base reference, Type = Unsigned Integer (PFC = 14)
This base shall be used as the zero reference for the offset

Offset = Offset from the base reference, Type = Signed Integer (PFC = 14)

Length = Number of octets to be dumped, Type = Unsigned Integer (PFC = 14)

On receipt of this primitive, the service shall send commands to an underlying communication service to dump the memory block.

DUMP_MEMORY_BO_IND (Device_Id, Base, Offset, Data, Error)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Base = Base reference, Type = Unsigned Integer (PFC = 14)
This base shall be used as the zero reference for the offset

Offset = Offset from the base reference, Type = Signed Integer (PFC = 14)

Data = Dumped data block to load, Type = Variable OctetString (PFC = 0)

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Check memory using base plus offset:

This primitive shall be used to check a memory block defined by a base, an offset and a length.

CHECK_MEMORY_BO_RQ(Device_Id, Base, Offset, Length)

Device_Id = Device logical Identifier, Type = Type = Unsigned Integer (PFC = 12)
The device_Id identifies the source memory block according to satellite architecture.

Base = Base reference, Type = Unsigned Integer (PFC = 14)
This base shall be used as the zero reference for the offset

Offset = Offset from the base reference, Type = Signed Integer (PFC = 14)

Length = Number of octets to be dumped, Type = Unsigned Integer (PFC = 14)

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- read the memory block
- calculate the checksum (ISO standard 16 bit checksum cf annex 3 of PUS)
- report the calculated checksum

CHECK_MEMORY_BO_IND (Device_Id, Checksum, Error)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Checksum = Memory block calculated checksum, Type = Unsigned Integer (PFC = 12)

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.2 Units power switching and resetting services

7.2.1 Device power switch-on service

This service shall be used for all the device switch-ons. It concerns simple devices without any configuration to apply up to large configurable equipment.

The service operations are the following:

- Power ON the device;
- Initialize the device if required;
- Get global status flag.

SET_DEVICE_POWER_ON_RQ (Device_Id)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication layer to switch ON and initialize the device.

This service shall manage specific timing constraints linked to device switch ON procedure.

SET_DEVICE_POWER_ON_IND (Device_Id, Status, Error)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Device logical status:, Type = Enumerated Parameter (PFC = 2)
00 = OFF: the device is power OFF
01 = Initialization in progress
10 = Initialized: operational mode or standby mode
11 = Failed: internal check during power switch on has not passed

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.2.2 Device power switch-off service

This service shall be used for all the device switch-offs. It concerns simple devices without any configuration to apply up to large configurable equipment.

The service operations are the following:

- Power OFF the device;
- Get global status flag.

SET_DEVICE_POWER_OFF_RQ (Device_Id)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to switch OFF the device.

This switch Off shall not be conditioned by any previous answer or internal state.

Only device switch OFF shall be performed whatever the device internal state. The switch Off command shall not be conditioned by any telemetry.

This service shall manage specific timing constraints linked to device switch OFF procedure.

SET_DEVICE_POWER_OFF_IND (Device_Id, Status, Error)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Device logical status, Type = Enumerated Parameter (PFC = 1)
0 = OFF
1 = ON

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.2.3 Device reset service

This service shall be used for device reset. This service can only be used with devices implementing an internal hardware reset.

The service operations are the following:

- Reset the device;
- Get global status flag.

SET_DEVICE_RESET_RQ (Device_Id)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to reset the device.

Only device reset shall be performed without any device configuration. The reset commands shall not be conditioned by any telemetry.

This service shall manage specific timing constraints linked to device Reset

SET_DEVICE_RESET_IND (Device_Id, Status, Error)

Device_Id = Device logical address, Type = Unsigned Integer (PFC = 12)

Status = Device logical status, Type = Enumerated Parameter (PFC = 2)
00 = Not applicable
01 = Initialization in progress
10 = Initialized: device is in operational mode or in standby mode
11 = Failed: internal check has not been passed during power switch

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.2.4 Device arming service

This service shall be used for device arming and disarming. This service can only be used with devices supporting an arming/disarming capacity.

Arming (disarming) can be done by relay selection, status position or any action ensuring a command enabling (disabling).

The following primitives are defined in this service:

- Arm the device;
- Disarm the device.

Device arming primitive:

SET_DEVICE_ARMING_RQ (Device_Id)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to arm the device.

SET_DEVICE_ARMING_IND (Device_Id, Status, Error)

Device_Id = Device logical address, Type = Unsigned Integer (PFC = 12)

Status = Device logical status, Type = Enumerated Parameter (PFC = 1)
1 = Device not armed
0 = Device armed

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Device disarming primitive:

SET_DEVICE_DISARMING_RQ (Device_Id)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to disarm the device.

SET_DEVICE_DISARMING_IND (Device_Id, Status, Error)

Device_Id = Device logical address, Type = Unsigned Integer (PFC = 12)

Status = Device logical status, Type = Enumerated Parameter (PFC = 1)
1 = Device not disarmed
0 = Device disarmed

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.2.5 Function enabling service

This service shall be used for function enabling or disabling. This service can only be used with devices supporting a function.

The following primitives are defined in this service:

- Enable device function;
- Disable device function.

Device function enabling primitive:

SET_DEVICE_FUNCTION_ENABLE_RQ (Device_Id, Function_Id)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Function_Id = Function logical Identifier, Type = Unsigned Integer (PFC = 4)

On receipt of this primitive, the service shall send commands to an underlying communication service to enable the device function.

SET_DEVICE_FUNCTION_ENABLE _IND (Device_Id, Function_Id, Status, Error)

Device_Id = Device logical address, Type = Unsigned Integer (PFC = 12)

Function_Id = Function logical Identifier, Type = Unsigned Integer (PFC = 4)

Status = Device logical status, Type = Enumerated Parameter (PFC = 1)
1 = Device function disabled
0 = Device function enabled

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Device function disabling primitive:

SET_DEVICE_FUNCTION_DISABLE_RQ (Device_Id, Function_Id)

Device_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Function_Id = Function logical Identifier, Type = Unsigned Integer (PFC = 4)

On receipt of this primitive, the service shall send commands to an underlying communication service to disable the device function.

SET_DEVICE_FUNCTION_DISABLE _IND (Device_Id, Function_Id, Status, Error)

Device_Id = Device logical address, Type = Unsigned Integer (PFC = 12)

Function_Id = Function logical Identifier, Type = Unsigned Integer (PFC = 4)

Status = Device logical status, Type = Enumerated Parameter (PFC = 1)
1 = Device function enabled
0 = Device function disabled

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3 Specific services

7.3.1 Platform Sensor and Actuator services

7.3.1.1 Specific sensors

7.3.1.1.1 Accelerometer service

This service shall be used for one axis Accelerometer devices. It provides cumulate speed increment for an elapsed number of time frames.

The service operations are the following:

- Get timer value: number of elapsed time frames;
- Get speed increment for the considered axis;
- Get global status flag.

GET_ACC_INC_RQ (GYRO_Id)

ACC_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to acquire Accelerometer data:

- Timer value: elapsed time frame between two acquisitions;
- Incremental speed increment for the accelerometer axis;
- Flags for Global status flag report.

GET_ACC_INC_IND (ACC_Id, ACC_Timer, ACC_incr_speed, ACC_Status, Error)

ACC_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

ACC_Timer = ACC elapsed time frame (between two acquisition), Type = Unsigned Integer (PFC=12)
Time frame value is defined for each equipment.

ACC_incr_speed = Incremental speed increment on one axis, Type = Signed Integer (PFC=12)
Speed increment is expressed in mm/s. LSB = 1 mm/s, limits are -32768mm/s and +32767mm/s

ACC_Status = Accelerometer status flag, Type = Enumerated (PFC = 1)
1 = ACC error
0 = no error

This status flag shall report all ACC internal flags which have an impact on data acquisition values.
This status flag shall not report data consistency check.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.1.2 Gyro service

This service shall be used for 3 rotation axis Gyrometer devices. It provides cumulate angles for an elapsed number of time frames.

The service operations are the following:

- Get timer value: number of elapsed time frames;
- Get Incremental delta angle vector for x, y and z axis;
- Get global status flag.

GET_GYRO_ANG_RQ (GYRO_Id)

GYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to acquire GYRO data:

- Timer value: number of elapsed time frame between two acquisitions;
- Incremental delta angle for x, y and z axis;
- Flags for Global status flag report.

GET_GYRO_ANG_IND (GYRO_Id, GYRO_Timer, GYRO_incr_ang[3], GYRO_Status, Error)

GYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

GYRO_Timer = GYRO elapsed time frame (between two acquisition). Type = Unsigned Integer (PFC=12)
Time frame value is defined for each equipment.

GYRO_incr_ang [3] = Incremental delta angle for x, y, z axis, Type = Signed Integer (PFC=12)
Angle value is expressed in µradian. LSB = 1 µradian, limits are -32768 µ rad and +32767 µrad

GYRO_Status = GYRO status flag, Type = Enumerated (PFC = 1)
1 = GYRO error
0 = no error

This status flag shall report all GYRO internal flags which have an impact on data acquisition values.
This status flag shall not report data consistency check.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.1.3 Star tracker service

This service shall be used for Star Tracker devices.

The service operations are as follows:

- Start a new acquisition;
- Compute quaternion data related to previous acquisition;
- Get global status flag.

GET_STR_QUAT_RQ (STR_Id)

STR_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to start Star Tracker next acquisition and to acquire Star Tracker previous data.

GET_STR_QUAT_IND (STR_Id, STR_Mode, Quaternion, Quality_index_xy, Quality_index_z, Status, Error)

STR_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

STR_Mode = STR current state, Type = Enumerated (PFC = 1)
1 = Tracking state
0 = Acquisition state, quaternion value is not OK

Quaternion = Computed quaternion, Type = Real (PFC = 1)
The quaternion is relative to J2000 earth identification mark

Quality_index_xy = Measured quality index on x and y axis, Type = Unsigned Integer
Quality index LSB = 1°/3600s

Quality_index_z = Measured quality index on z axis, Type = Unsigned Integer
Quality index LSB = 1°/3600s

STR_Status = STR status flag, Type = Enumerated (PFC = 1)
1 = STR error
0 = no error

This status flag shall report all STR internal flags which have an impact on data acquisition values.
This status flag shall not report data consistency check.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.1.4 Earth sensor service

This service shall be used for 2 axis attitude Earth Sensor devices.

The following primitives are defined in this service:

- Data acquisition primitive;
- Acquisition mode selection primitives;
- Array inhibition primitives.

ES attitude data acquisition primitives:

These primitives shall be used for ES 2 axis attitude data acquisition.

The primitive operations are the following:

- Roll and pitch value acquisition;
- Attitude measurement status acquisition.

GET_ES_DATA_RQ (ES_Id)

ES_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to acquire ES data (roll and pitch) and measurement status acquisition.

GET_ES_DATA_IND (ES_Id, Roll, Pitch, ES_Meas_status, Error)

ES_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Roll = value for Roll axis, Type = Signed Integer (PFC=12)
Angle value is expressed in u radian. LSB = 50u radian

Pitch = value for Pitch axis, Type = Signed Integer (PFC=12)
Angle value is expressed in u radian. LSB = 50u radian

ES_Meas_status = ES status flag, Type = Enumerated (PFC = 1)
1 = ES error
0 = no error

This status flag shall report all ES internal flag which have an impact on data acquisition values.

This status flag shall be ES internal flags without any data consistency check performed.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

ES acquisition mode selection primitives:

This primitive shall be used for ES fine or coarse acquisition mode setting.

SET_ES_MODE_RQ (ES_Id, Mode)

ES_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Mode = ES mode, Type = Enumerated (PFC = 1)
1 = Fine acquisition mode
0 = Coarse acquisition mode

On receipt of this primitive the service shall send commands to an underlying communication service to set ES in fine or coarse acquisition mode.

SET_ES_MODE_IND (ES_Id, Mode, Error)

ES_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Mode = ES mode, Type = Enumerated (PFC = 1)
1 = Fine acquisition mode
0 = Coarse acquisition mode

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

ES array inhibition primitive:

This primitive shall be used for ES array inhibition or activation.

SET_ES_ARRAY_RQ (ES_Id, Array_inhib)

ES_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Array_inhib = 4 arrays inhibition command parameter, Type = Fixed BitString (PFC = 4)
1XXX = Array N°1 inhibition
0XXX = Array N°1 activation
X1XX = Array N°2 inhibition
X0XX = Array N°2 activation
XX1X = Array N°3 inhibition
XX0X = Array N°3 activation
XXX1 = Array N°4 inhibition
XXX0 = Array N°4 activation

On receipt of this primitive the service shall send commands to an underlying communication service to inhibit or active selected Earth sensor array.

SET_ES_ARRAY_IND (ES_Id, Array_inhib, Error)

ES_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Array_inhib = 4 arrays inhibition command parameter, Type = Fixed BitString (PFC = 4)
1XXX = Array N°1 inhibited
0XXX = Array N°1 used
X1XX = Array N°2 inhibited
X0XX = Array N°2 used
XX1X = Array N°3 inhibited
XX0X = Array N°3 used
XXX1 = Array N°4 inhibited
XXX0 = Array N°4 used

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.1.5 Sun Sensor service

This service shall be used for Sun Sensor devices

The service operations are the following:

- Get Sun vector on 3 axis;
- Get Sun Presence flag;
- Get global status;
- Start a new acquisition.

GET_SUN_SENSOR_DATA_RQ (SS_Id)

SS_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to acquire SS data.

This service shall start a new acquisition after previous data telemetry.

GET_SUN_SENSOR_DATA_IND (SS_Id, SS_sun_vector[3], Sun Presence, SS_status, Error)

SS_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

SS_sun_vector[3] = Measured sun vector on the 3 axis, Format = Signed Integer (PFC=12)
Reference axis is Sun Sensor based, vector values are normalized between -1 and +1.
Vector LSB = 2^-15

Sun Presence = Sun Presence Flag, Type = Boolean
1 = Sun Presence effective
0 = No Sun Presence

SS_Status = SS status flag, Type = Enumerated (PFC = 1)
1 = SS error
0 = no error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.1.6 Magnetometer service

This service shall be used for Magnetometer devices.

The service operations are the following:

- Get static magnetic field on 3 axis;
- Get global status flag.

GET_MAGNETOMETER_RQ(MAG_Id)

MAG_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to acquired and compute magnetic field on 3 axis.

GET_MAGNETOMETER_IND(MAG_Id, Field[3], MAG_Status)

MAG_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Field[3] = Static magnetic field on x, y and z axis, Type = Signed Integer (PFC = 12)
Magnetic field value LSB = 4 nano Tesla

MAG_Status = MAG status flag, Type = Enumerated (PFC = 1)
1 = MAG error
0 = no error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2 Specific actuator

7.3.1.2.1 Thruster configuration service

This service shall be used for thruster devices. This service shall configure n thruster bi stable or latch valves.

The service operations are the following:

- Configure n thruster Bi Stable or latch Valves;
- Configure the associated electronic interfaces.

SET_THRUSTER_BSV_VALVE_RQ (BSV_Id_list[n], BSV_cmd_list[n])

BSV_Id_list[n] = n Device logical Identifier list, Type = Unsigned Integer (PFC = 12)

BSV_cmd_list[n]) = n Bi Stable Valve command list to apply, Type = Enumerated (PFC = 1)
1 = Open valve
0 = Close valve

On receipt of this primitive the service shall send commands to an underlying communication service to select and configure the BSV list.

SET_THRUSTER_BSV_VALVE_IND (BSV_Id_list[n], BSV_state_list[n], Error)

BSV_Id_list[n] = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

BSV_state_list[n] = Bi Stable Valve state: OPEN or CLOSE, Type = Enumerated (PFC = 1)
1 = Open state
0 = Close state

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2.2 Venting service

This service shall be used for thruster venting. This service shall perform venting for a list of n thruster.

Venting is performed by opening valves until the reception of a stop venting.

The stop venting is applied to all the thruster

Primitive operations for start venting:

- Open valves of the n thruster list.

Primitive operations for stop venting:

- Close all the thruster valves.

Start venting primitive:

SET_VENTING_START_RQ (Thruster_Id_list[n])

Thruster_Id_list[n] = n Device logical identifier list, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to start venting operation for a list of n thruster.

SET_VENTING_START_IND (Thruster_Id_list[n], Thruster_venting_status[n], Error)

Thruster_Id_list[n] = n Device logical identifier list, Type = Unsigned Integer (PFC = 12)

Thruster_venting_status[n]= n thruster list, Type = Enumerated (PFC = 1)
1 = Thruster venting not initialized
0 = Thruster venting in progress

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Stop venting primitive:

SET_VENTING_STOP_RQ (Thruster_Id_list[n])

Thruster_Id_list[n] = n Device logical identifier list, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to stop venting operation for all the thruster, whatever their state.

SET_VENTING_STOP_IND (Thruster_Id_list[n], Thruster_venting_status[n], Error)

Thruster_Id_list[n] = n Device logical identifier list, Type = Unsigned Integer (PFC = 12)

Thruster_venting_status[n]= n thruster list, Type = Enumerated (PFC = 1)
1 = Thruster venting in progress
0 = No thruster venting in progress

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2.3 Apogee Boost Motor service

This service shall be used for Apogee Boost Motor devices. This service shall use both nominal and redundant command interfaces.

The following primitives are defined in this service:

- ABM start boost;
- ABM stop boost.

ABM switch On primitives:

This primitive shall be used for ABM configuration and start boost.

SET_ABM_ON_RQ (ABM_Id)

ABM_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to configure and switch ON the ABM.

SET_ABM_ON_IND (ABM_Id, Status, Error)

ABM_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = ABM error
0 = no error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

ABM switch OFF primitives:

This primitive shall be used for ABM stop boost.

SET_ABM_OFF_RQ (ABM_Id)

ABM_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to switch OFF the ABM.

SET_ABM_ OFF _IND (ABM_Id, Status, Error)

ABM_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = ABM error
0 = no error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2.4 Thruster burst service

Operations:

- Configure Mono Stable Valves time impulsion for n thruster;
- Set n thruster activity for the next AOCS period.

The AOCS period is user defined, meanwhile, the thruster activity will be configured independently for each 1/100 of the period.

The maximum number of simultaneous thruster commands is user defined.

This primitive shall be sent for each AOCS period.

SET_THRUSTER_MSV_VALVE_RQ (MSV_Id_list[n], MSV_cmd_list[n])

MSV_Id_list[n] = n Device logical Identifier list, Type = Unsigned Integer (PFC = 12)

MSV_cmd_list[n] = n Mono Stable Valve activity in next AOCS period, Type = Enumerated (PFC = 100)
1 = MSV active during the 1/100 considered AOCS period
0 = MSV not active during the 1/100 considered AOCS period

On receipt of this primitive the service shall send commands to an underlying communication service to configure MSV activity for the next AOCS period.

SET_THRUSTER_MSV_VALVE_IND (MSV_Id_list[n], MSV_history_list[n], Status_list[n], Error)

MSV_Id_list[n] = n Device logical Identifier list, Type = Unsigned Integer (PFC = 12)

MSV_history_list[n] = n Mono Stable Valve history during the previous AOCS period, Type = enumerated (PFC = 100)

Status_list[n] = n thruster status flag list, Type = Enumerated (PFC = 1)
1 = Thruster error
0 = no error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2.5 Plasmic Propulsion service

This service shall be used for plasmic propulsion thruster devices. This service implements redundancy at plasmic thruster level for the cathode and the monostable valve.

The following primitives are defined in this service:

- plasmic propulsion configuration and start boost;
- plasmic propulsion stop boost.

PP thruster venting is performed by the generic venting service.

The TOM (Thruster Orientation Mechanism) command is performed by SM (Stepper Motor) service.

Plasmic propulsion start primitives:

This primitive shall be used for PP thruster configuration and start boost.

The service operations are the following:

- Select plasmic thruster;
- Select thruster cathode A or B;
- Select thruster valve redundancy N or R;
- Set and check heater and warm up current;
- Start boost and check discharge current.

SET_PP_START_RQ (PP_thruster_Id, PP_cathode, PP_valve)

PP_thruster_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

PP_cathode = PPU cathode A or B selection, Type = Enumerated (PFC = 1)
1 = Cathode A selection
0 = Cathode B selection

PP_valve = Thruster valve redundancy selection, Type = Enumerated (PFC = 1)
1 = Nominal valve selection
0 = Redundant valve selection

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- Select plasmic thruster;
- Select thruster cathode A or B;
- Select thruster valve redundancy N or R;
- Set and check heater and warm up current;
- Start boost and check discharge current.

SET_PP_START_IND (PP_thruster_Id, Status, Error)

PP_thruster_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = PP failed, propulsion not performed
0 = PP propulsion in progress

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Plasmic propulsion stop primitives:

This primitive shall be used for plasmic propulsion stop boost.

SET_PP_STOP_RQ (PP_thruster_Id, PP_cathode, PP_valve)

PP_thruster_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- Stop boost;
- Deselect thruster cathode A or B and N or R valve.

SET_PP_STOP_IND (PP_thruster_Id, Status, Error)

PP_thruster_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = PP failed, stop not performed
0 = PP stopped

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2.6 Wheel service

This service shall be used for reaction wheel devices.

The service operations are the following:

- Set speed or set torque;
- Get wheel speed;
- Get global status.

RW has two distinct control principles: speed control or torque control.

The present document defines the two capabilities using a generic service but distinguishing command types.

Set speed or set torque will be set using the following service:

Speed or torque command primitive:

SET_RW_RQ (RW_Id, Cmd_type, RW_cmd)

RW_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Cmd_Type = Reaction Wheel command mode, Type = BOOLEAN
1 = speed mode
0 = torque mode

RW_cmd = Reaction wheel speed or torque command mode, Type = Signed Integer (PFC=12)
For speed mode, LSB = 1 rotation per minute
For torque mode, LSB = 0.5Nm/2^15

On receipt of this primitive, Speed or Torque value range shall be checked.
(in each service having input parameter, this parameter shall be systematically checked in the acceptable range)

In this service, the command check is not recommended because of RW inertia and AOCS loop period. (to avoid memory saturation)

SET_RW_SPEED_IND (RW_Id, Cmd_Status, Error)

RW_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Cmd_status = Command status, Type = Enumerated (PFC = 1)
1 = Command not sent
0 = no error, command sent.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Speed acquisition primitives:

GET_RW_SPEED_RQ (RW_Id, Speed_value, RW_Status, Error)

RW_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to acquire Reaction Wheel speed value.

GET_RW_SPEED_IND (RW_Id, Speed_value, RW_Status, Error)

This service returns the following report:

RW_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Speed_value = Reaction Wheel filtered speed value, type = Signed Integer (PFC = 12)
Speed value LSB = 1 rotation per minute

RW_status = Reaction wheel internal status flag, Type = Enumerated (PFC = 1)
1 = RW internal error
0 = no error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2.7 Pyrotechnic service

This service shall be used for pyrotechnic devices.

This service shall command pyrotechnic devices and acquire pyrotechnic device status by both nominal and redundant interfaces.

The following primitives are defined in this service:

- pyrotechnic device selection;
- pyrotechnic device deselecting;
- pyrotechnic device firing.

Pyro selection primitives:

This service shall perform the pyrotechnic device selection.

SET_PYRO_SELECT_RQ (PYRO_Id)

PYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication to select a pyrotechnic device. This selection shall be done both by nominal and redundant interfaces.

SET_PYRO_SELECT_IND (PYRO_Id, Nominal_Status, Redundant_Status, Error)

PYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Nominal_Status = Pyrotechnic device arming status by nominal interface, Type = Enumerated (PFC = 1)
1 = Not selected
0 = Selected, Armed

Redundant_Status = Pyrotechnic device arming status by redundant interface, Type = Enumerated (PFC = 1)
1 = Not selected
0 = Selected, Armed

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Pyro deselecting primitives:

This primitive shall be used for pyrotechnic device deselection.

SET_PYRO_DESELECT_RQ (PYRO_Id)

PYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication to deselect a pyrotechnic device. This deselecting shall be done both by nominal and redundant interfaces.

SET_PYRO_DESELECT_IND (PYRO_Id, Nominal_Status, Redundant_Status, Error)

PYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Nominal_Status = Pyrotechnic device arming status by nominal interface, Type = Enumerated (PFC = 1)
1 = Not selected
0 = Selected, Armed

Redundant_Status = Pyrotechnic device arming status by redundant interface, Type = Enumerated (PFC = 1)
1 = Not selected
0 = Selected, Armed

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Pyro firing primitives:

This primitive shall be used for pyrotechnic device firing.

SET_PYRO_FIRING_RQ (PYRO_Id)

PYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication to fire a pyrotechnic device. This fire command shall be done both by nominal and redundant interfaces.

SET_PYRO_FIRING_IND (PYRO_Id, Nominal_Status, Redundant_Status, Error)

PYRO_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Nominal_Status = Pyrotechnic device arming status by nominal interface, Type = Enumerated (PFC = 1)
1 = Not selected
0 = Selected, Fired

Redundant_Status = Pyrotechnic device arming status by redundant interface, Type = Enumerated (PFC = 1)
1 = Not selected
0 = Selected, Fired

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.2.8 Magnetic-torquers service

This service shall be used for Magnetic Torquer devices. This service shall be used for each Magneto Torquer set on the 3 axis.

The service operations are the following:

- Set current to apply on the magnetic torquer;
- Get global status flag.

SET_MT_CURRENT_RQ (MT_Id, Current).

MT_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Current = MT current to apply, Type = Unsigned Integer (PFC = 4)
Current value LSB = 10mA

On receipt of this primitive, the service shall send commands to an underlying communication service to apply the current value on the Magnetic Torquer.

SET_MT_CURRENT_IND (MT_Id, Status, Error)

MT_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = MT status flag, Type = Enumerated (PFC = 1)
1 = ES error
0 = no error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.3 Platform management

7.3.1.3.1 Battery reconditioning service

This service shall be used for battery reconditioning. It performs the selected battery start or stop discharge with a selected electronic chain with a defined load.

Battery reconditioning service application field is Nickel accumulators according to memory effects.

The following primitives are defined in this service:

- Battery start reconditioning;
- Battery stop reconditioning;
- Battery reconditioning load change.

Battery start reconditioning primitives:

This primitive shall be used for battery start reconditioning.

The primitive operations are the following:

- Select the battery management electronic;
- Select the battery to discharge;
- Select load to apply and start discharge.

SET_BATT_RECOND_START_RQ (BM_Id, BATT_Id, BM_LOAD)

BM_Id = Battery Management logical Identifier, Type = Unsigned Integer (PFC = 12)

BATT_Id = Battery logical Identifier, Type = Unsigned Integer (PFC = 12)

BM_LOAD = Discharge load to apply:, Type = Enumerated Parameter (PFC = 2)
00 = No load
01 = Low rate
10 = Medium rate
11 = High rate

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- select the battery to discharge;
- select the battery management electronic to use;
- select the discharge rate to apply;
- start the battery discharge.

SET_BATT_RECOND_START_IND (BM_Id, BATT_Id, BM_LOAD, Error)

BM_Id = Selected Battery Management logical Identifier, Type = Unsigned Integer (PFC = 12)

BATT_Id = Selected Battery logical Identifier, Type = Unsigned Integer (PFC = 12)

BM_LOAD = Applied discharge load:, Type = Enumerated Parameter (PFC = 2)
00 = No load
01 = Low rate
10 = Medium rate
11 = High rate

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Battery stop reconditioning primitive:

This primitive shall be used for battery discharge stop.

SET_BATT_RECOND_STOP_RQ (BM_Id)

BM_Id = Battery Management logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- deselect discharge switch for all the batteries;
- deselect all the discharge loads (set to no load).

SET_BATT_RECOND_STOP_IND (BM_Id, BATT_status, BM_LOAD, Error)

BM_Id = Selected Battery Management logical Identifier, Type = Unsigned Integer (PFC = 12)

BATT_Status = Battery selection status, Type = Enumerated (PFC = 1)
1 = Battery selected
0 = No battery selected

BM_LOAD = Applied discharge load:, Type = Enumerated Parameter (PFC = 2)
00 = No load (Expected value)
01 = Low rate
10 = Medium rate
11 = High rate

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Battery reconditioning load change primitive:

This primitive shall be used for battery discharge load change.

SET_BATT_RECOND_LOAD_RQ (BM_Id, BM_LOAD).

BM_Id = Selected Battery Management logical Identifier, Type = Unsigned Integer (PFC = 12)

BM_LOAD = Discharge load to apply:, Type = Enumerated Parameter (PFC = 2)
00 = No load
01 = Low rate
10 = Medium rate
11 = High rate

On receipt of this primitive, the service shall send commands to an underlying communication service to change the current discharge load used by a Battery management device.

SET_BATT_RECOND_LOAD_IND (BM_Id, BM_LOAD, Error)

BM_Id = Selected Battery Management logical Identifier, Type = Unsigned Integer (PFC = 12)

BM_LOAD = Applied discharge load:, Type = Enumerated Parameter (PFC = 2)
00 = No load
01 = Low rate
10 = Medium rate
11 = High rate

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.3.2 Power conditioning service

This service shall be used for power conditioning management.

The service operations are the following:

- Set maximum battery charge current intensity.

SET_PCU_CHARGE_INT_RQ (BCR_Id, Intensity)

BCR_Id = Battery Charge Regulator Identifier, Type = Unsigned Integer (PFC = 12)

Intensity = Maximum charge intensity = Unsigned Integer (PFC = 12)
Intensity value LSB = 0.1A

SET_PCU_CHARGE_INT_IND (BCR_Id, Status, Error)

BCR_Id = Battery Charge Regulator Identifier, Type = Unsigned Integer (PFC = 12)

Status = Device status, Type = Enumerated (PFC = 1)
1 = Failed
0 = OK

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.1.3.3 Stepper motor management service

This service shall be used for stepper motor device management. Stepper motors can be used for antenna orientation, plasma thruster orientation, solar array orientation or other applications.

Stepper motor position is acquired by two types of sensor:

- Fine position sensor set on the axis motor;
- Coarse position sensor set on the function orientation axis after the motor reducer.

A reference sensor is also used for the motor reference position setting.

The following primitives are defined in this service:

- Stepper motor pointing in relative mode or in cruise mode;
- Stepper motor reference setting;
- Stepper motor position acquisition.

Stepper motor motion command primitive:

This primitive shall be used for setting a relative motion or configuring the motor in a cruise mode. The motion is performed at the defined step rate.

SET_SM_MOTION_RQ (Motor_Id, Step_Nb, Step_rate)

Motor_Id = Stepper motor logical Identifier, Type = Unsigned Integer (PFC = 12)

Step_Number = Motion step number to apply, Type = Unsigned Integer (PFC = 12)
The 0 value indicates a cruise mode at the corresponding step rate.

Step_rate = Motion step rate to apply, Type = Real (PFC = 1)
Motion value is defined in Hertz.
Positive value = Forward direction
Negative value = Reverse direction
0 value = No motion, used to stop motion in cruise mode

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- Select the stepper motor to command and configure the associated electronics;
- Send direction step number and step rate to apply;
- Start motion or stop motion if the step rate value is 0.

SET_SM_MOTION_IND (Motor_Id, Status, Error)

Motor_Id = Stepper motor logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Device status, Type = Enumerated (PFC = 1)
1 = Failed
0 = OK

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Stepper motor reference position setting primitives:

This primitive shall be used for setting the motor in the reference position whatever the initial position is.

SET_SM_REFERENCE_RQ (Motor_Id)

Motor_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to motion the motor up to the reference position. The step rate shall be defined in the data base.
The motion shall automatically stop if the reference is not found after one entire turn.

SET_SM_REFERENCE_IND (Motor_Id, Status, Error)

Motor_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Reference position status flag, Type = Enumerated (PFC = 1)
1 = Reference position failed
0 = Reference position reached

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Stepper motor position acquisition primitive:

This primitive shall be used for stepper motor position acquisition.

GET_SM_SENSOR_RQ (Sensor_Id)

Sensor_Id = Position sensor logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- Select the position sensor;
- Acquire the position value.

The service is the same whatever the sensor precision and accuracy.

GET_SM_SENSOR_IND (Sensor_Id, Position, Error)

Sensor_Id = Selected position sensor logical Identifier, Type = Unsigned Integer (PFC = 12)

Position = Position sensor value, Type = Unsigned Integer (PFC = 12)
Position sensor LSB =1/2^16 x 2 PI radian

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.2 Payload & TTC services

7.3.2.1 Synthesizers service

This service shall be used for synthesizers configuration. This service shall set the frequency.

The service operations are the following:

- Set frequency.

SET_FREQUENCY_RQ (SYNTHE_Id, Frequency)

SYNTHE_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Frequency = frequency identifier, Type = Unsigned integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to configure the device.

SET_FREQUENCY_IND (SYNTHE_Id, Status, Error)

SYNTHE _Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = SYNTHE Error
0 = No error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.2.2 Power amplifiers service

This service shall be used for (EPC+TWTA) configuration and status acquisition.

The following primitives are defined in this service:

- Set autorestart configuration;
- Get TWTA global status.

TWTA autorestart setting primitive:

This primitive shall configure the autorestart.

SET_TWTA_AUTORESTART_RQ (TWTA_Id, TWTA_autorestart)

TWTA_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

TWTA_autorestart = autorestart of the EPC, Type = Boolean
1 = autorestart enable
0 = autorestart disable

On receipt of this primitive, the service shall send commands to an underlying communication service to configure the autorestart function (enable/disable).

SET_TWTA_ AUTORESTART _IND (TWTA_Id, Status, Error)

TWTA_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Fixed BitString (PFC = 3)
1XX = TWTA Error
0XX = No error
X1X = Autorestart occurred
X0X = No Autorestart occurred
XX1 = Autorestart function active
XX0 = Autorestart function inactive

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

TWTA status acquisition primitive:

This primitive shall be used for TWTA status acquisition.

GET_TWTA_STATUS_RQ (TWTA_Id)

On receipt of this primitive, the service shall send commands to an underlying communication service to acquire the global status of the TWTA.

GET_TWTA_STATUS_IND (TWTA_Id, Status, Error)

TWTA_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Fixed BitString (PFC = 3)
1XX = TWTA Error
0XX = No error
X1X = Autorestart occurred
X0X = No Autorestart occurred
XX1 = Autorestart function active
XX0 = Autorestart function inactive

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.2.3 Channel amplifier

This service shall be used for Channel Amplifier (CAMP) configuration. This service shall configure CAMP mode and gain.

The service operations are the following:

- Select FGM, ALC, BLANKING mode or AGEING compensation;
- Set ALC or FGM or AGEING gain.

SET_CAMP_CONFIG_RQ (CAMP_Id, CAMP_mode, CAMP_gain_level)

CAMP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

CAMP_mode = FGM, ALC, BLANKING mode or AGEING compensation, Type = Enumerated Parameter (PFC = 2)
00 = BLANKING mode (RF OFF)
01 = FGM mode (Fixed Gain mode)
10 = ALC mode (Automatic Level Control)
11 = AGEING compensation command
(Ageing compensation do not change the current mode)

CAMP_gain_level = level state to apply for ALC or gain state for FGM mode or Ageing compensation, Type = Unsigned Integer (PFC = 4)
0 = User defined minimum value to apply
255 = User defined maximum value to apply

On receipt of this primitive, the service shall send commands to an underlying communication service to configure the device.

SET_CAMP_CONFIG_IND (CAMP_Id, Status, Error)

CAMP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = CAMP Error
0 = No error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

NOTE: CAMP RF output power shall be acquired by monitoring.

7.3.2.4 Switching service

This service shall be used for switch configuration and status acquisition. This service is switch type independent. (random, incremental, etc).

The following primitives are defined in this service:

- Set switch configuration;
- Get switch configuration.

Switch configuration command primitive:

This primitive shall configure the addressed switch.

SET_SWITCH_CONFIG_RQ (SWITCH_Id, SWITCH_Position)

SWITCH_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

SWITCH_Position = switch position, Type = Unsigned Integer (PFC = 0)
0000 = Position 0
1111 = Position 15

On receipt of this primitive, the service shall send commands to an underlying communication service to configure the switch position.

SET_SWITCH_CONFIG_IND (SWITCH_Id, SWITCH_Position, Error)

SWITCH_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

SWITCH_Position = switch position, Type = Unsigned Integer (PFC = 0)
0000 = Position 0
1111 = Position 15

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Switch configuration acquisition primitive:

This primitive shall acquire the switch configuration.

GET_SWITCH_CONFIG_RQ (SWITCH_Id, SWITCH_Position)

SWITCH_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service shall send commands to an underlying communication service to acquire the switch position configuration.

GET_SWITCH_CONFIG_IND (SWITCH_Id, SWITCH_Position, Error)

SWITCH_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

SWITCH_Position = switch position, Type = Unsigned Integer (PFC = 0)
0000 = Position 0
1111 = Position 15

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.3.2.5 Digital Transparent Processor service

This service shall be used for DTP (Digital Transparent Processor) configuration. This service shall configure the different parameters of this device through the following primitives:

- Set DTP ON/OFF links (input, output);
- Set DTP sub-channel routing configuration (input_number, freq. Min, bandwidth, output_number, freq. Min, gain value);
- Get DTP RF power acquisition;
- Get DTP configuration.

Remarks:

- The ON/OFF of the DTP is done using the device power switch-on/off services.
- The "memory management services" shall be used if applicable to the DTP.
- The temperature and voltages are retrieved using the "simple services".

Links setting ON primitives:

This primitive shall be used for DTP input or output link switch ON.

SET_DTP_LINK_ON_RQ(DTP_Id, Link_number)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Link_number = link identification, Type = Unsigned Integer (PFC = 2)
0XXXXX = Input
1XXXXX = Output
Xyyyyy => number=yyyyy (0=00000 to 31=11111)

On receipt of this primitive, the service shall send commands to an underlying communication service to set ON the DTP input or output link

SET_DTP_LINK_ON_IND (DTP_Id, Link_number, Status, Error)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Link number = link identification, Type = Unsigned Integer (PFC = 2)
0XXXXX = Input
1XXXXX = Output
Xyyyyy => number=yyyyy (0=00000 to 31=11111)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = DTP Link Error
0 = No error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Links setting OFF primitives:

SET_DTP_LINK_OFF_RQ(DTP_Id, Link_number)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Link number = link identification, Type = Unsigned Integer (PFC = 2)
0XXXXX = Input
1XXXXX = Output
Xyyyyy => number=yyyyy (0=00000 to 31=11111)

On receipt of this primitive, the service shall send commands to an underlying communication service to set OFF the DTP input or output link.

SET_DTP_LINK_OFF_IND (DTP_Id, Link_number, Status, Error)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Link number = link identification, Type = Unsigned Integer (PFC = 2)
0XXXXX = Input
1XXXXX = Output
Xyyyyy => number=yyyyy (0=00000 to 31=11111)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = DTP Link Error
0 = No error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Routing configuration primitive:

SET_DTP_CONFIG_RQ (DTP_Id, Input_number, Freq_input_min, Bandwidth, Output_number, Freq_output_min, Gain_value).

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Input_number = Input link number, Type = Unsigned Integer (PFC = 2)
0yyyyy => number=yyyyy (0=00000 to 31=11111)

Freq_input_min = Sub_channel frequency of the <*Input_number*> link, Type = Unsigned Integer (PFC = 14)

Bandwidth = Bandwidth of the sub_channel, Type = Unsigned Integer (PFC = 14)

Output _number = Output link number, Type = Unsigned Integer (PFC = 2)
0yyyyy => number=yyyyy (0=00000 to 31=11111)

Freq_ output _min = Sub_channel frequency of the < *output _number*> link, Type = Unsigned Integer (PFC = 14)

Gain_value = gain of the sub-channel, Type=Unsigned Integer (PFC = 4)
0 = minimum value
255=maximum value

On receipt of this primitive, the service shall send commands to an underlying communication service to apply the required configuration which consists in connect:
an input sub-channel [<*Freq_input_min*>; < *Freq_input_min* >+<*Bandwidth*>] of the <*Input_number*> link to the output sub-channel [<*Freq_output_min* >; < *Freq_output_min* >+< *Bandwidth* >] of the <*Output_number*> link. The service shall also apply the <*Gain_value*> on the <*Output_number*> sub_channel.

SET_DTP_CONFIG_IND (DTP_Id, Status, Error)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = DTP configuration Error
0 = No error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Power acquisition primitive:

This primitive shall be used for a DTP link power acquisition

GET_DTP_POWER_RQ (DTP_Id, Link_number)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Link number = input link number, Type = Unsigned Integer (PFC = 2)
0yyyyy => number=yyyyy (0=00000 to 31=11111)

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- Select the link number;
- Perform a power acquisition on the related link.

GET_DTP_POWER_IND (DTP_Id, Link_number, Power_value[32], Status, Error)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Power_value[32] = table of 32 power values Type = Unsigned Integer (PFC = 12)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = DTP configuration Error
0 = No error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

DTP link configuration acquisition primitive:

This primitive shall be used for DTP configuration acquisition:

GET_DTP_CONFIG_RQ (DTP_Id, Link_number)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Link number = output link number, Type = Unsigned Integer (PFC = 2)
1yyyyy => number=yyyyy (0=00000 to 31=11111)

On receipt of this primitive, the service shall send commands to an underlying communication service to:

- Select the output link number and retrieve all information related to it (32 possible sub channel information).

GET_DTP_CONFIG_IND (DTP_Id, Link_number, Input_value[32], Input_Freq_value[32], Bandwidth[32], Gain_value[32], Output_freq_value[32], Status, Error)

DTP_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Link number = output link number, Type = Unsigned Integer (PFC = 2)
1yyyyy => number=yyyyy (0=00000 to 31=11111)

Input_value[32] = array of input link number related to the specified output link number, Type of array element = Unsigned Integer (PFC = 2)
0yyyyy => number=yyyyy (0=00000 to 31=11111) or
UNUSED=1xxxxx => link number not used in this configuration

Input_Freq_value[32] = array of Sub_channel frequency n°i, Type of array element = Unsigned Integer (PFC = 14)

Bandwidth[32] = array of Bandwidth of the sub_channel n°i, Type of array element = Unsigned Integer (PFC = 14)

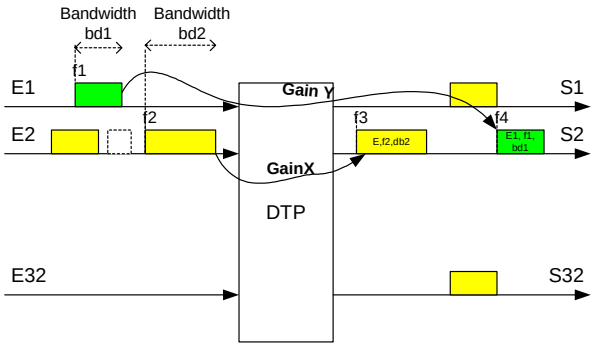
Gain_value[32] = array of gain of the related sub-channel n°i, Type=Unsigned Integer (PFC = 4)
0 = minimum value
255=maximum value

Output_freq_ value[32] = array of Sub_channel frequency of the output link number, Type of array element = Unsigned Integer (PFC = 14)

Status = Global status flag, Type = Enumerated (PFC = 1)
1 = DTP configuration Error
0 = No error

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

Example of the output link configuration of output n°2



The return values related to output link n°S2 will be:

Input_value[0] = E2 (000001_{BIN})
Input_freq_value[0] = f2,
Bandwidth[0] = bd2,
Output_freq_value[0] = f3
Gain_value[0] = Gain X

Input_value[1] = E1(000000_{BIN})
Input_freq_value[1] = f1,
Bandwidth[1] = bd1,
Output_freq_value[1] = f4
Gain_value[1] = Gain Y

Input_value[2] = UNUSED (111111_{BIN})
Input_freq_value[2] = 0,
Bandwidth[2] = 0,
Output_freq_value[2] = 0
Gain_value[2] = 0

…

Input_value[31] = UNUSED (111111_{BIN})
Input_freq_value[31] = 0,
Bandwidth[31] = 0,
Output_freq_value[31] = 0
Gain_value[31] = 0

7.4 Simple devices services

7.4.1 Pulse command service

This service shall be used for pulse command activation. Pulse command can be single type or matrix type without any distinction.

The service operations are the following:

- Generate a pulse command to a device.

SET_PULSE_RQ (PULSE_Id)

PULSE_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to generate a pulse command to a device.
Pulse command electrical interface or format depends on the device identifier.

SET_PULSE_IND (PULSE_Id, Status, Error)

PULSE_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Status = Optional device status, Type = Enumerated (PFC = 1)
1 = Open
0 = Close
This status is optional and shall be used for device bi stable state change according to the pulse service.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.4.2 Temperature acquisition service

This service shall be used for temperature acquisition.

Sensor type could be: Thermistor, Platinum Sensor, Semi-conductor sensor, optical sensor, etc.
The service interface shall be independent of the temperature sensor type.
Temperature range and accuracy depends on the device.

The service operations are the following:

- Get temperature;
- Engineering unit conversion;
- Format the acquired data.

GET_TEMPERATURE_RQ (Temp_Id)

Temp_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive the service shall send commands to an underlying communication service to acquire device temperature.

This service could generate one or several acquisition instructions depending on the used correction process. This service return the sensor temperature.

GET_TEMPERATURE_IND (Temp_Id, Temp_Value, Error)

Temp_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Temp_Value = Device temperature, Type = Real (PFC=1)
Temperature value is expressed in Kelvin degree.
Value delivered by the device with or without any correction or computing.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.4.3 Pressure acquisition service

This service shall be used for pressure acquisition.

The service shall be independent of the pressure sensor type.
Pressure range and accuracy depend on the device.

The service operations are the following:

- Get pressure;
- Engineering unit conversion;
- Format the acquired data.

GET_PRESSURE_RQ (Pressure_Id)

Pressure_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service sends acquisition instructions to an underlying communication service for transmission to the device interfacing the pressure sensor.

This service could generate one or several acquisition instructions depending on the used correction process.

This service returns the sensor pressure value.

GET_PRESSURE_IND (Pressure_Id, Pressure_Value, Error)

Pressure_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Pressure_Value = Device pressure, Type = Real (PFC = 1)
Pressure value is expressed in Pascal.
Value delivered by the device with or without any correction or computing.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.4.4 Current acquisition service

This service shall be used for current acquisition.

The service shall be independent of the current sensor type.

The service operations are the following:

- Get current;
- Engineering unit conversion;
- Format the acquired data.

GET_CURRENT_RQ (Current_Id)

Current_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service sends acquisition instructions to an underlying communication service for transmission to the device interfacing the current sensor.

This service could generate one or several acquisition instructions depending on the used correction process.

This service returns the current value.

GET_CURRENT_IND (Current_Id, Current_Value, Error)

Current_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Current_Value = Device current, Type = Real (PFC = 1)
Current value is expressed in Ampere
Value delivered by the device with or without any correction or computing

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.4.5 Voltage acquisition service

This service shall be used for temperature acquisition.

The service shall cover all types of Analogue to Digital Converter.

The service operations are the following:

- Get voltage;
- Engineering unit conversion;
- Format the acquired data.

GET_VOLTAGE_RQ (Voltage_Id)

Voltage_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service sends acquisition instructions to an underlying communication service for transmission to the device interfacing the Analogue to Digital Converter.

This service could generate one or several acquisition instructions depending on the used correction process.

This service returns the voltage value.

GET_VOLTAGE_IND (Voltage_Id, Voltage_Value, Error)

Voltage_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Voltage_Value = Device voltage, Type = Real (PFC = 1)
Voltage value is expressed in Volt.
Value delivered by the device with or without any correction or computing.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

7.4.6 Frequency acquisition service

This service shall be used for frequency acquisition.

The service shall cover all the different type of frequency transducer.

The service operations are the following:

- Get frequency;
- Engineering unit conversion;
- Format the acquired data.

GET_FREQUENCY_RQ (Frequency_Id)

Frequency_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

On receipt of this primitive, the service sends acquisition instructions to an underlying communication service for transmission to the device acquiring the frequency.

This service could generate one or several acquisition instructions depending on the used correction process.

This service returns the frequency value.

GET_FREQUENCY_IND (Frequency_Id, Frequency_Value, Error)

Frequency_Id = Device logical Identifier, Type = Unsigned Integer (PFC = 12)

Frequency_Value = Device frequency, Type = Real (PFC = 1)
Frequency value is expressed in Hertz.
Value delivered by the device with or without any correction or computing.

Error = Error flag, Type = Boolean
1 = acquisition not performed
0 = acquisition performed

8 Design requirements

When a service defined in the present document is used, the following rules apply:

- the mnemonic as specified in the present document shall be uniquely used for service reference (software, specification, database, documentation);
- all services attributes shall be defined in the system database.

Annex A (informative): End to end example

The goal of this annex is to provide end to end example including all impacts of these recommendations on a space segment. The chain covers from Space Segment Control (SCC) level to on satellite equipment level.

Example of temperature acquisition:

The thermal control of the IRES uses SSDHI service to interface with SOIF C&DA service to acquire the temperature from a remote terminal connected to the central computer by OBDH data bus.

The temperature sensor is acquired by an analogue acquisition chain controlled by an electronic bloc interfaced to the Remote Terminal Unit (RTU) by a Digital Serial (DS16) line.

Source of communication	Destination of communication	Information source	Used information	SSDHI/SOIF services	Param.	Layer
Interrogation						
Thermal control Application SW	SSDHI service			Regulation of IRES Temp (Id)		User application
IRES temperature	SOIF service	Satellite SDB	-IRES/TH sensor correspondence -Sensor type	GET_TEMPERATURE_RQ(Temp_Id)	Id=TH32	Application interface service
TH32	OBDH service	Satellite SDB	-RTU Id -Channel Id	C&DA SOIF service Read device (Id)	Id=TH32	Application layer (SOIF)
TH32 comes from RTU 2 channel 45	HW service	Data bus format given by the SDB	-Format of OBDH interrogation	Write interog OBDH (RTU Id,TH32)	RTU Id=RTU 2 TH32 = channel 45	Data link layer
RTU 2 line 45 uses µRTU function commanding electronic function	serial I/F DS16 Nb 26 of the µRTU	RTU IDS	HW implementation	HW channel access		User interface
DS16 Nb 26 acquires	Analog signal generated by the TH	RTU IDS	HW implementation	HW acquisition		User interface
Response						
Analog channel DS16 inside RTU And OBDH Data bus		RTU IDS	HW implementation	HW puts response on OBDH response bus	The OBDH respond automatically and SOIF service will just have to get the message	User interface + Data link layer
Application layer C&DA service		Correlation is performed in SW Service requirement	-interrogation/ response correspondence is done by HW	C&DA SOIF service Read device indication (Data)	Raw data	Application layer (SOIF)
SSDHI Service		Transfer function is in SDB	- transfer function parameters	GET_TEMPERATURE_IND(Temp_Id, Temp_Value, Error)	Id=TH32 Data = EUC data Engineering unit converted	Application interface service
NOTE: If we use networks, relations/connections between networks are identified in the SDB.						

Annex B (informative): Comments and recommendations for usage of the present document

The object of this annex is to provide complementary information on:

- Identifier type definition;
- Distinction between status and error parameters;
- Recommendations on services;
- SOIF services not kept in the present document;
- Services not defined in the present document.

B.1 Identifier type definition

Most of the primitives concern a device defined by an Identifier.

The devices are onboard components which user software can control or monitor. These devices can be simple devices like sensors and actuators, or can be more complex devices like payload computers.

This identifier will allow to define:

- Device type to access;
- Device access way;
- Physical support for the data access;
- Data link encoding.

EXAMPLE: Let us consider the GET_TEMP_RQ and GET_TEMP_IND primitive:

- At application interface services level, the identifier will determine the device type (calibration purpose if needed) and the access way (use of the application layer primitives).
- At sub-network layer, the identifier will make the correspondence with the physical media and the data link encoding.
- We will note that the device identifier is independent of the media redundancy.
- The present document specifies an Unsigned Integer (PFC=12) i.e. up to 2^16 = 65536 Identifiers.

B.2 Distinction between status and error parameters

In the present document, status and error parameters are used in Indication primitives.

The status parameter is associated to the device (or function) internal state. It reflects the device (or function) current state.

The error parameter does not reflect the device (or function) internal status.

The error parameter means that the primitive execution has failed and that the other indication primitive parameters are not valid.

B.3 Recommendations on services

Specific services:
The services which are linked to very specific devices should be implemented by proprietary services.

Switching service:
This service can be used for RF switching, configuration selection in power generation, etc.

Device power switch on service:
This service could initialize the device in different modes according to Device_Id. If the equipment requires parameters to be initialized, a proprietary service will be used.

Gyro and Sun Sensor services:
For 2 axis cases the z axis is not fed.

Star Tracker service:
To add elapsed time between the start of AOCS cycle and half of integration time it is necessary to create a proprietary service.

B.4 SOIF services not kept in the present document

The following services have been removed in order to not overlap present document with SOIF work.

They will be defined by the SOIF standardization group at the application layer level and will provide access to the underlying communication protocol.

- File transfer service;
- File management service;
- Time distribution;
- Unconfirmed message service;
- Confirmed message service;
- Stream service;
- Serial digital command service;
- Serial digital acquisition service;
- Analogue acquisition service;
- Digital acquisition service.

B.5 Services not defined in the present document

Services to be implemented at higher level of the application layer:

Failure Detection Isolation and Recovery related services: architecture dependant.

On board monitoring service: software architecture dependant, and strongly impacts CPU and communication loads. This service has to be implemented in application layer, above this Interface level.

On board processing service: discarded because concerns proprietary applications to be implemented above this interface level.

Forecast services for future evolution:

- Global Positioning Service;
- Camera service;
- Active Antenna service;
- Battery balancing for Li-Ion batteries.

Units or major sub-systems not requiring specific services:

- Receivers: use of the identified low level services (on-off, serial links, etc.)
- Device pointing: use motor commands, because computing is performed by the ground, and if an on-board vector translation is necessary it is in the application layer above this Interface level.
- TT&C services: use of the identified low level services (on-off, frequency setting, etc.).

Annex C (informative): SOIF description

C.1 General

SOIF provides an internationally agreed set of specifications for standard services and interfaces to be used onboard spacecraft. These specifications have been produced by the CCSDS P1K sub-panel and have been approved and endorsed by the CCSDS member agencies. The SOIF standard services and interfaces simplify the way software accesses the underlying spacecraft hardware, assures interoperability between onboard applications, and increases the potential for re-use of both hardware and software components.

Typically, each SOIF service is specified using two documents, a service specification, which defines the capability that is provided by the service, and a service interface specification, which defines the users interface to the service in terms of service primitives. This approach ensures that different implementations of a given service will provide the same interface to user applications.

The SOIF services and interfaces are designed to be used directly by user applications, but may also be used as the foundation blocks of more capable or more specialized services. For example, a generic device driver for star trackers could be built on top of the SOIF command and data acquisition service.

The SOIF services are illustrated in figure C.1.

Amongst the SOIF services, the Command and Data Acquisition service is likely to be most relevant to the present document, and is described in the following clause.

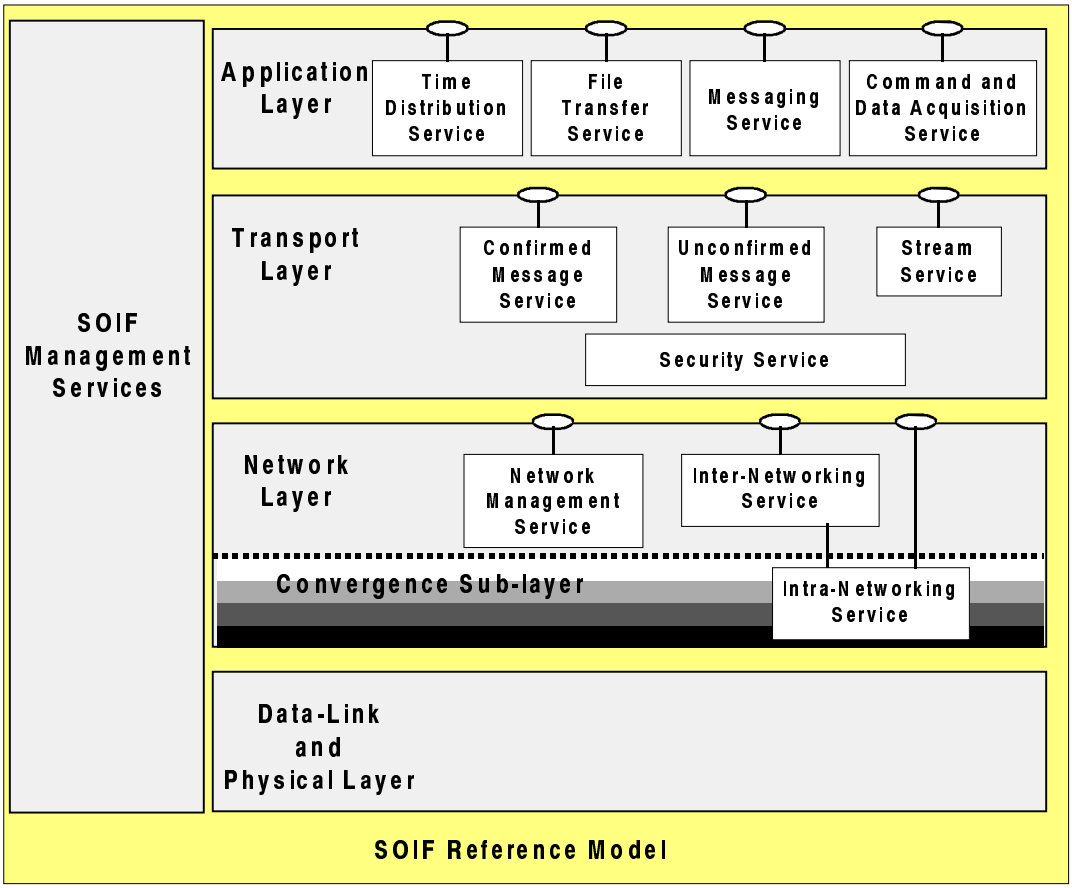


Figure C.1: The SOIF Services

C.2 SOIF Command and Data Acquisition Service

The SOIF command and data acquisition service enables users to issue commands and retrieve data from sensors and actuators without requiring detailed knowledge of the hardware, the underlying interfaces, or the location of the sensors and actuators. In operation, the service allows users to read or write to a local representation of the sensor or actuator, and the service translates those read and write accesses into the operations that are needed to read or write the real device.

C.3 Architectural Context

Figure C.2 shows the command and data acquisition service within the SOIF architecture. The key points to note are:

- The command and data acquisition service resides in the application layer;
- Below the physical layer there are locally connected sensors and the spacecraft bus;
- Locally connected sensors are accessed via P-SAP's (P for Physical). These can be called directly from the application layer;
- The spacecraft bus is accessed via one or more T-SAP's (T for Transport) or L-SAP's (L for Data Link) that can be reached directly from the application layer;
- The command and data acquisition service could use the SOIF messaging service, which also resides in the SOIF application layer.

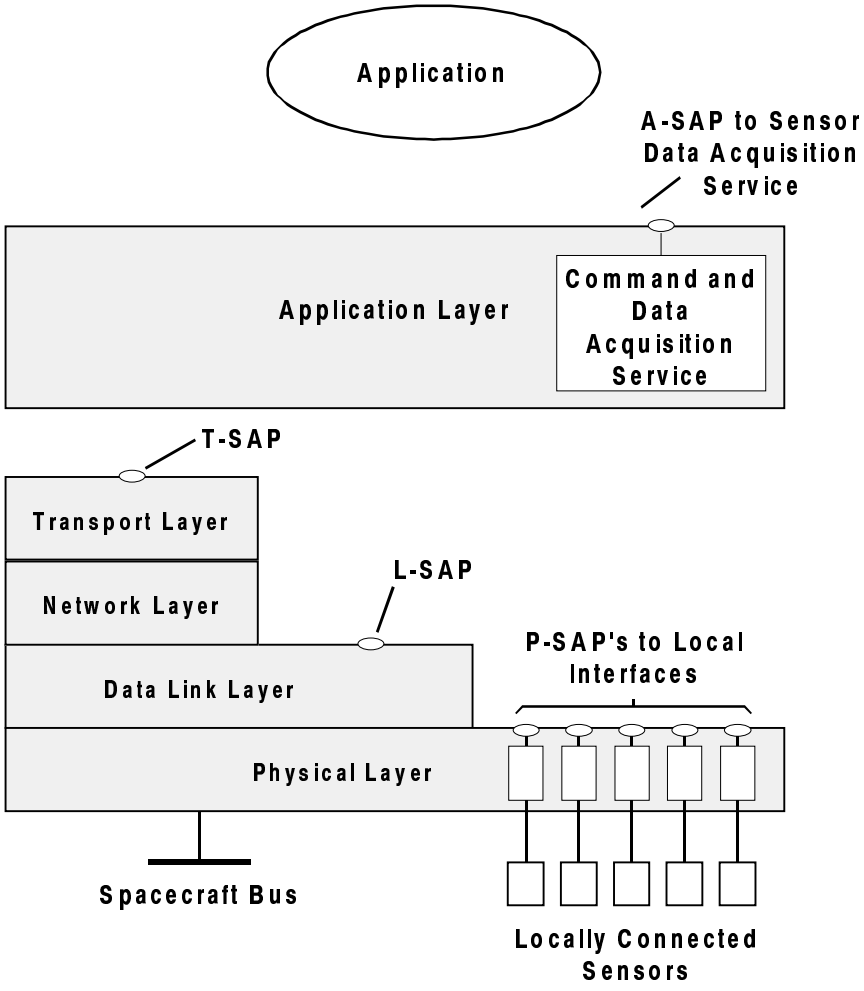


Figure C.2: Architectural Context of the SOIF Command and Data Acquisition Service

C.4 Service Operation - Service User Perspective

The command and data acquisition service provides two primitives, ReadDeviceReq and WriteDeviceReq, through which an onboard application can read and write data from and to simple onboard devices such as sensors and actuators.

The ReadDevReq primitive is used to read a device, and requires a single parameter, device ID, that identifies the device to be read. After receiving a ReadDeviceReq, the service responds with a ReadDevInd either containing the requested data, or indicating an error. The device ID must be unique within the domain of the spacecraft. The user does not need to know the location of the sensor, or how it is physically accessed in the hardware.

The WriteDeviceReq is used to write data to a device, and requires two parameters, the device ID and the data to be written.

C.5 Service Operation - Service Providers Perspective

The function of the command and data acquisition service is to read and write data from and to specified onboard devices on receipt of service requests from the user. The location of the device will typically be related to the location of the service user in one of the following ways:

- The device is locally connected;
- The device is connected directly to the spacecraft bus;
- The device is connected to a remote node that can be reached via the spacecraft bus.

In the first case, the command and data acquisition service can simply read or write the sensor via the appropriate local P-SAP. This is the simplest case.

In the second case, the command and data acquisition service can access the device directly via the spacecraft bus by issuing the appropriate request through the L-SAP. If the spacecraft bus is Mil. Std 1553B for example, this might involve issuing a command word and acquiring the response. An L-SAP can be used because there are no transport or network layer services required, the bus address of the sensor must be known by the sensor data acquisition service, and the appropriate command must be formatted.

In the third case, where the device is connected to another node of the onboard network, the local command and data acquisition service must transfer the request to its counterpart in the node to which the device is connected. I.e. if the service user is hosted on node A, and the device is connected to node B, the command and data acquisition service in node A sends a message to its counterpart in node B, indicating the device to be accessed. The command and data acquisition service in node B then performs the requested operation (read or write) and may relay acquired data back to node A in another message.

In this case, the SOIF network layer services are used to transfer the messages between the command and data acquisition service entities. Transport layer services may also be used, and the SOIF messaging service might provide the required transfer capability.

Annex D (informative): Parameter type code and format

Parameter type code and format are those of Packet Utilization Standard document.

They are defined in chapter 23.3 of Packet Utilization standard ESA PSS 07 101 Issue 1.

This clause summarizes the parameter types used in the present document.

It defines the physical encoding rules for each type, i.e. the permitted lengths of the parameter fields and the internal format used to encode values. This clause does not define the conversion of data parameters into physical or engineering units or user messages.

D.1 Encoding formats of parameter types

The parameter type defines the range of possible parameter values. A given parameter type can vary in format and length. Each combination of parameter type and encoding format has an associated parameter code, which defines the type and its physical encoding.

The parameter code shall be used whenever a definition of a parameter field is required. The parameter codes shall be applicable to both Telecommand and telemetry data.

The parameter code PC, is defined as follows:

Parameter Type Code (PTC)	Parameter Format Code (PFC)
4 bits	4 bits

The parameter code is written as (PTC, PFC) in the tables below.

D.2 Parameter type definitions

D.2.1 Boolean

Parameter Type	PTC	PFC	Length	Value/Range
Boolean	1	0	1 bit	0 = false, 1 = true

D.2.2 Enumerated Parameter

Enumerated parameters are parameters with distinct integer values only involved in logical operations(as opposed to numeric operations). The values that such a parameter can take are discrete and un-ordered. An error code is a typical example.

Parameter Type	PTC	PFC	Length
Enumerated Parameter	2	1	1 bit
	2	2	2 bits
	2	3	3 bits
	2	4	4 bits
	2	5	5 bits
	2	6	6 bits
	2	7	7 bits
	2	8	8 bits
	2	12	12 bits
	2	16	16 bits

D.2.3 Unsigned Integer

Parameter Type	PTC	PFC	Length	Value/Range
Unsigned Integer	3	0	4 bits	{0 to 15}
	3	1	5 bits	{0 to 31}
	3	2	6 bits	{0 to 63}
	3	3	7 bits	{0 to 127}
	3	4	8 bits	{0 to 255}
	3	5	9 bits	{0 to 511}
	3	6	10 bits	{0 to 1 023}
	3	7	11 bits	{0 to 2 047}
	3	8	12 bits	{0 to 4 095}
	3	9	13 bits	{0 to 8 191}
	3	10	14 bits	{0 to 16 383}
	3	11	15 bits	{0 to 32 767}
	3	12	2 octets	{0 to 65 536}
	3	13	3 octets	{0 to 2^{24}-1}
	3	14	4 octets	{0 to 2^{32}-1}

D.2.4 Signed Integer

Parameter Type	PTC	PFC	Length	Value/Range
Signed Integer	4	0	4 bits	{-8 to 7}
	4	1	5 bits	{-16 to 15}
	4	2	6 bits	{-32 to 31}
	4	3	7 bits	{-64 to 63}
	4	4	8 bits	{-128 to 27}
	4	5	9 bits	{-256 to 255}
	4	6	10 bits	{-512 to 511}
	4	7	11 bits	{-1 024 to 1 023}
	4	8	12 bits	{-2 048 to 2 047}
	4	9	13 bits	{-4 096 to 4 095}
	4	10	14 bits	{-8 192 to 8 191}
	4	11	15 bits	{-16 384 to 16 383}
	4	12	2 octets	{-32 768 to 32 767}
	4	13	3 octets	{-2^{2} to 2^{23}-1}
	4	14	4 octets	{-2^{31} to 2^{31}-1}

D.2.5 Real

Parameter Type	PTC	PFC	Length	Sign S	Exponent E	Fraction b_i
Real	5	1	4 octets	bit 0	bit 1 to bit 8	bit 9 to bit 31
	5	2	8 octets	bit 0	bit1 to bit 11	bit 12 to bit 63

Two formats for real numbers shall be allowed:

PC(5,1) and PC(5,2) are 32-bit single precision and 64 bit double precision format according to IEEE Standard 754 [2]. (single precision format used also for internal parameter of DSP21020.)

NOTE: 32-bit single precision format according to MIL-STD-1750-A (used also for interface of DSP21020 with AIU/AOCMS-SW) is not proposed.

Each format permits the representation of numerical values of the form:

$(-1)^S \times 2^E \times (b_0.b_1.b_2\ldots\ldots b_{p-1})$, where:

- $(b_0.b_1.b_2\ldots\ldots b_{p-1})$ means $\frac{b_0}{2^0}+\frac{b_1}{2^1}+\frac{b_2}{2^2}+\ldots\ldots\frac{b_{p-1}}{2^{p-1}}$
- S = 0 or 1
- E = any integer between Emin and Emax, inclusive
- bi = 0 or 1
- p = number of significant bits

The value of a single precision parameter is:

Exponent	Fraction	Value
255	<> 0	not a defined number
255	0	(-1)^(sign) × infinity
< > 255, <> 0	any	(-1)^(sign) × 2^(-127) × (1.fraction)
0	<> 0	(-1)^(sign) × 2^(-126) × (0.fraction)
0	0	0

EXAMPLE 1:

sign = 0
exponent = 127
fraction = 110 0000 0000 0000 0000 0000
value = -1^0 × 2^(127-127) × $(1.110\ 0000\ 0000\ 0000\ 0000\ 0000)_{BIN} = 1{,}75_{DEC}$

EXAMPLE 2:

sign = 0
exponent = 0
fraction = 110 0000 0000 0000 0000 0000
value = -1^0 × 2^(0-126) × $(0.110\ 0000\ 0000\ 0000\ 0000\ 0000)_{BIN} = 2\text{^-}126 \times 0{,}75_{DEC}$

The value of a double precision parameter is:

Exponent	Fraction	Value
2 047	<> 0	not a defined number
2 047	0	(-1)^(sign) × infinity
< > 2 047, <> 0	any	(-1)^(sign) × 2^(exponent-1 023) × (1.fraction)
0	<> 0	(-1)^(sign) × 2^(-1 022) × (0.fraction)
0	0	0

D.2.6 Bit-String

Parameter Type	PTC	PFC	Length field	Length
BitString	6	0	n = Unsigned Integer	n bits
	6	1	N/A	1 bit
	6	2	N/A	2 bits
	6	n (<> 0)	N/A	n bits
NOTE: PFC = 0 for Variable BitString PFC <> 0 for fixed BitString				

D.2.7 Octet-String

Parameter Type	PTC	PFC	Length field	Length
OctetString	7	0	n = Unsigned Integer	n octets
	7	1	N/A	1 octet
	7	2	N/A	2 octets
	7	n (<> 0)	N/A	n octets
NOTE: FC = 0 for Variable OctetString PFC <> 0 for fixed OctetString				

Annex E (informative): Bibliography

- Packet Utilization Standard, European Space Agency, ESA PSS-07-101, Issue 1, May 1994 (ECSS version ECSS-E-70-41).
- "Spacecraft Onboard Interface, White Book", CCSDS, April 2001.
- "Strawman Application Layer Services for SOIF" by Chris Plummer may 2001.
- "A common Terminology and Principles of layering for SOIF" by Chris Plummer, May 2001.
- ECSS-E-50 and SOIF- "Roles and Relationships". ESA.
- Interface meeting ETSI - ESA 16 May 2001.
- "CCSDS spacecraft onboard interfaces concept and rationale": Green Book.

History

Document history		
V1.1.1	October 2002	One-step Approval Procedure OAP 20030221: 2002-10-23 to 2003-02-21
V1.1.1	February 2003	Publication