



BSI Standards Publication

Space engineering - Telemetry and telecommand packet utilization

National foreword

This British Standard is the UK implementation of EN 16603-70-41:2017.

The UK participation in its preparation was entrusted to Technical Committee ACE/68, Space systems and operations.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2017
Published by BSI Standards Limited 2017

ISBN 978 0 580 95152 7

ICS 49.140

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 October 2017.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

English version

Space engineering - Telemetry and telecommand packet utilization

Ingénierie spatiale - Utilisation de la télémétrie et de la télécommande par paquets

Raumfahrttechnik - Telemetrie und Telekommando

This European Standard was approved by CEN on 30 June 2017.

CEN and CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN and CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN and CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN and CENELEC members are the national standards bodies and national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



**CEN-CENELEC Management Centre:
Avenue Marnix 17, B-1000 Brussels**

Table of contents

European Foreword.....	6
Introduction.....	8
1 Scope.....	9
2 Normative references	11
3 Terms, definitions and abbreviated terms.....	12
3.1 Terms from other standards.....	12
3.2 Terms specific to the present standard	12
3.3 Abbreviated terms.....	16
3.4 Nomenclature	17
4 Context and background	18
4.1 Introduction.....	18
4.2 Modelling the PUS.....	21
5 The PUS foundation model.....	25
5.1 Introduction.....	25
5.2 Convention	26
5.3 The generic service type abstraction level	26
5.4 The generic service deployment abstraction level.....	38
6 Service type system requirements	55
6.1 ST[01] request verification	55
6.2 ST[02] device access.....	63
6.3 ST[03] housekeeping.....	76
6.4 ST[04] parameter statistics reporting	109
6.5 ST[05] event reporting	119
6.6 ST[06] memory management	124
6.7 ST[07] (reserved).....	153
6.8 ST[08] function management.....	153
6.9 ST[09] time management.....	155
6.10 ST[10] (reserved).....	161

6.11	ST[11] time-based scheduling	161
6.12	ST[12] on-board monitoring	191
6.13	ST[13] large packet transfer.....	221
6.14	ST[14] real-time forwarding control.....	229
6.15	ST[15] on-board storage and retrieval.....	256
6.16	ST[16] (reserved).....	308
6.17	ST[17] test	308
6.18	ST[18] on-board control procedure	311
6.19	ST[19] event-action.....	331
6.20	ST[20] parameter management	341
6.21	ST[21] request sequencing	347
6.22	ST[22] position-based scheduling.....	357
6.23	ST[23] file management.....	390
7	Space to ground interface requirements.....	412
7.1	Introduction.....	412
7.2	Convention	414
7.3	Packet field type code.....	415
7.4	The CCSDS Space Packet.....	425
8	Service type interface requirements.....	432
8.1	ST[01] request verification	432
8.2	ST[02] device access.....	437
8.3	ST[03] housekeeping	442
8.4	ST[04] parameter statistics reporting	459
8.5	ST[05] event reporting	462
8.6	ST[06] memory management	465
8.7	ST[07] (reserved).....	475
8.8	ST[08] function management.....	475
8.9	ST[09] time management.....	476
8.10	ST[10] (reserved).....	478
8.11	ST[11] time-based scheduling	478
8.12	ST[12] on-board monitoring	490
8.13	ST[13] large packet transfer.....	507
8.14	ST[14] real-time forwarding control.....	510
8.15	ST[15] on-board storage and retrieval.....	520
8.16	ST[16] (reserved).....	538
8.17	ST[17] test	538

8.18	ST[18] on-board control procedure	540
8.19	ST[19] event-action.....	548
8.20	ST[20] on-board parameter management	552
8.21	ST[21] request sequencing	556
8.22	ST[22] position-based scheduling	561
8.23	ST[23] file management.....	574
9	Command Pulse Distribution Unit	584
9.1	Scope	584
9.2	System requirements.....	584
9.3	Interface requirements.....	586
Annex A	(informative) IEEE and MIL-STD real formats	588
A.1	IEEE standard format	588
A.1.1	General	588
A.1.2	Single-precision	589
A.1.3	Double-precision	589
A.2	United States Air Force military standard format.....	591
A.2.1	General	591
A.2.2	Simple-precision	591
A.2.3	Extended.....	592
Annex B	(informative) CRC and ISO checksum.....	593
B.1	The cyclic redundancy code (CRC)	593
B.1.1	General	593
B.1.2	Symbols and conventions	594
B.1.3	Encoding procedure	594
B.1.4	Decoding procedure.....	594
B.1.5	Verification of compliance	595
B.1.6	Software implementation.....	595
B.2	The ISO checksum	604
B.2.1	General	604
B.2.2	Symbols and conventions	605
B.2.3	Encoding procedure	605
B.2.4	Decoding procedure.....	606
B.2.5	Verification of compliance	606
B.2.6	Software implementation.....	606
Annex C	(informative) Summary of requests and reports for PUS standard services.....	612

C.1	Convention	612
C.2	Requests and reports	612
C.2.1	ST[01] request verification.....	612
C.2.2	ST[02] device access	614
C.2.3	ST[03] housekeeping	615
C.2.4	ST[04] parameter statistics reporting.....	617
C.2.5	ST[05] event reporting.....	618
C.2.6	ST[06] memory management.....	619
C.2.7	ST[07] (reserved)	621
C.2.8	ST[08] function management	621
C.2.9	ST[09] time management.....	621
C.2.10	ST[10] (reserved)	622
C.2.11	ST[11] time-based scheduling	622
C.2.12	ST[12] on-board monitoring.....	624
C.2.13	ST[13] large packet transfer	625
C.2.14	ST[14] real-time forwarding control	626
C.2.15	ST[15] on-board storage and retrieval	628
C.2.16	ST[16] (reserved)	630
C.2.17	ST[17] test	631
C.2.18	ST[18] on-board operations procedure.....	631
C.2.19	ST[19] event-action	633
C.2.20	ST[20] Parameter management	634
C.2.21	ST[21] request sequencing	634
C.2.22	ST[22] position-based scheduling	636
C.2.23	ST[23] file management	638
Annex D (informative) System and interface specification index		640
Bibliography.....		641

European Foreword

This document (EN 16603-70-41:2017) has been prepared by Technical Committee CEN-CENELEC/TC 5 "Space", the secretariat of which is held by DIN.

This standard (EN 16603-70-41:2017) originates from ECSS-E-ST-70-41C.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by April 2018, and conflicting national standards shall be withdrawn at the latest by April 2018.

This document supersedes EN 14776:2004 Space engineering – Ground systems and operations - Telemetry and telecommand packet utilization.

The main purpose of the update of EN 14776:2004 was the need:

- to remove the deficiencies of the previous issue and to inject lessons learned,
- to improve the standard to meet the need for future missions,
- to acknowledge the existence of new ECSS and CCSDS standards and to ensure consistency,
- to implement the updated drafting rules that apply to any ECSS Standards (e.g. naming each requirement to facilitate tailoring, traceability),
- maintaining backward compatibility when possible.

The main changes are:

- the introduction of the PUS foundation model that:
 - has been used to produce the "standard service types";
 - shall be used to produce the "mission-specific service types", i.e.:
 - ◇ adding new service types, subservice types, message types, etc;
 - ◇ adding capabilities to the "standard service types";
 - shall be used to produce the "mission services", i.e.:
 - ◇ creating the required services by:
 - ◆ "realising the service types", and
 - ◆ inheriting all mandatory subservices and minimum capabilities;
 - ◇ selecting, for each service, the additional capabilities, the optional subservices, etc;
 - ◇ creating the service specific definitions.

- a proper separation of system versus interface requirements.
- material common to the three ECSS branches (Space engineering, Space project management and Space product assurance moved to EN 16601-00 (EN version of ECSS-S-ST-00), material specific to product assurance to this standard,
- reorganization of the content of the document to separate descriptive text and requirements, including clarification, modification of requirements and implementation of change requests,
- pre-tailoring matrix (Clause 6) added,
- addition of new DRDs.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a standardization request given to CEN by the European Commission and the European Free Trade Association.

This document has been developed to cover specifically space systems and has therefore precedence over any EN covering the same scope but with a wider domain of applicability (e.g. : aerospace).

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

Introduction

The CCSDS Space Packet Protocol (CCSDS 133.0-B-1) and the ECSS-E-ST-50 series of standards address the end-to-end transport of telemetry and telecommand data between user applications on the ground and application processes on-board the spacecraft, and the intermediate transfer of these data through the different elements of the ground and space segments.

This packet utilization standard (PUS) complements those standards by defining the application-level interface between ground and space, in order to satisfy the requirements of electrical integration and testing and flight operations.

1

Scope

This Standard addresses the utilization of telecommand packets and telemetry packets for the purposes of remote monitoring and control of spacecraft subsystems and payloads.

This Standard does not address mission-specific payload data packets, but the rules contained herein can be extended to suit the requirements of any mission.

This Standard does not address audio and video data as they are not contained within either telecommand or telemetry packets.

This Standard defines a set of services that satisfy all the fundamental operational requirements for spacecraft monitoring and control during spacecraft integration, testing and flight operations, refer to ECSS-E-ST-70-11. It also specifies the structure and contents of the telecommand packets used to transport the requests and the telemetry packets used to transport the reports.

This Standard can be used by any mission, no matter what its domain of application, orbit or ground station coverage characteristics. However, it is not the intention that the PUS should be applied in its entirety to a given mission. The services defined in this Standard cover a wide spectrum of operational scenarios and, for a given mission, only a subset of these services is likely to be appropriate.

Choices are made early in the design phase of a new mission resulting in the need to tailor the PUS to suit the requirements of that mission. These choices include:

- the on-board system design and architecture, in terms of the number of on-board application processes, their on-board implementation (e.g. the allocation to on-board processors) and their roles (i.e. which functions or subsystems or payloads they support);
- which PUS services are supported by each application process.

Each mission usually documents the results of this design and selection process in a **"Space-to-Ground Interface Control Document"**.

Some missions implement a centralized architecture with a small number of application processes, whilst others have a highly-distributed architecture within which a correspondingly larger number of application processes are distributed across several on-board processors.

The specification of services in this Standard is adapted to the expectation that different missions require different levels of complexity and capability from a given service. To this end, all services are optional and a given service can be implemented at one of several distinct levels, corresponding to the inclusion of

one or more capability sets. The minimum capability set corresponds to the simplest possible level that also remains sensible and coherent. At least this set is included in every implementation of a given service.

The standardized PUS services fulfil the following criteria:

- Commonality: each standard service corresponds to a group of capabilities applicable to many missions.
- Coherence: the capabilities provided by each standard service are closely related and their scope is unambiguously specified. Each standard service covers all the activities for managing inter-related state information and all activities that use that state information.
- Self-containment: each standard service has minimum and well-defined interactions with other services or on-board functions.
- Implementation independence: the standard services neither assume nor exclude a particular spacecraft architecture (hardware or software).

This Standard mainly addresses the requirements that apply to the spacecraft and its components. The ground segment counterpart requirements related to the testing or the operations of the spacecraft and its components can be derived from these requirements and are not specified in this Standard. Tailoring the PUS for a mission is mainly a task for the operations team and the spacecraft manufacturer. This Standard assumes that the mission ground segment used to test or operate the spacecraft implements all standardized capabilities and as such, does not further constrain the mission tailoring process of these capabilities.

The PUS should be viewed as a "Menu" from which the applicable services and service-levels are selected for a given mission. This selection process is repeated for each on-board application process, since each application process is designed to provide a specific set of tailored services.

This standard may be tailored for the specific characteristics and constraints of a space project in conformance with ECSS-S-ST-00.

This Standard does not include any protection against inadequate operations. This is considered mission specific.

2

Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ECSS Standard. For dated references, subsequent amendments to, or revision of any of these publications do not apply. However, parties to agreements based on this ECSS Standard are encouraged to investigate the possibility of applying the more recent editions of the normative documents indicated below. For undated references, the latest edition of the publication referred to applies.

EN reference	Reference in text	Title
EN 16601-00-01	ECSS-S-ST-00-01	ECSS system – Glossary of terms
EN 16603-70	ECSS-E-ST-70	Space engineering – Ground systems and operations
EN 16603-70-01	ECSS-E-ST-70-01	Space engineering – Spacecraft on-board control procedures
EN 16603-70-11	ECSS-E-ST-70-11	Space engineering – Space segment operability
EN 16603-70-31	ECSS-E-ST-70-31	Space engineering – Ground systems and operations – Monitoring and control data definition
	CCSDS 133.0-B-1, September 2003	Space Packet Protocol, Blue Book
	CCSDS 301.0-B-4, November 2010	Time Code Formats, Blue Book

Terms, definitions and abbreviated terms

3.1 Terms from other standards

- a. For the purpose of this Standard, the terms and definitions from ECSS-S-ST-00-01 apply, in particular for the following terms:
 - 1. space system
 - 2. space segment
 - 3. spacecraft
 - 4. ground segment

3.2 Terms specific to the present standard

3.2.1 acceptance notification

notification that is generated by the acceptance and reporting subservice provider of the application process that hosts the subservice provider in charge of executing the related request

3.2.2 acceptance verification report

report generated by the acceptance and reporting subservice provider as a consequence of a request acceptance verification

NOTE The acceptance and reporting subservice for a request is hosted by the application process that hosts the subservice responsible for executing that request. Each acceptance verification report is reporting either the successful acceptance of a request or the failed acceptance. In case of successful acceptance, the request is sent to the subservice provider in charge of its execution. In case of failed acceptance, the request is rejected and as such, not sent to any subservice provider.

3.2.3 application process

element of the space system that can host one or more subservice entities

NOTE An application process resides either on-board or on ground. An on-board application process usually hosts some subservice providers but can

also host some subservice users. A ground application process usually hosts some subservice users. If a ground application process is remotely controlled by the ground monitoring and control system, that application process behaves as an on-board application process and can host some subservice providers.

3.2.4 capability

functionality of a service or a subservice

NOTE A capability is specified by a set of operational requirements for a function of the overall space system that can be remotely controlled by the ground monitoring and control system or by other on-board applications. This Standard mainly addresses these remote controlled related requirements and especially those applicable to the subservice providers.

3.2.5 data report

report generated by a subservice provider as part of the subservice functionality

NOTE A data report can be generated in response to a request or to an instruction to elicit some specific service data. A data report can also be generated autonomously, when reports are enabled by a request, or as part of a continuous reporting functionality.

3.2.6 event report

report related to an occurrence of an event

NOTE Event reports are generated by the subservice providers.

3.2.7 execution notification

notification that is generated by the subservice provider in charge of execution of the related instruction

NOTE An execution notification reports on the successful or failed execution of an instruction. This Standard does not specify how the notifications are implemented on-board, nor how the subservice providers in charge of their generation interact with the subservice providers in charge of generating the corresponding execution verification reports.

3.2.8 execution verification report

report generated by the execution reporting subservice provider of an application process as a consequence of the reception of one or more execution notifications

NOTE The execution reporting subservice for a request is hosted by the application process that hosts the subservice responsible for executing that request. Each execution verification report is reporting either a successful or a failed execution stage (start, progress or completion) of a request.

3.2.9 instruction

elementary constituent of a request that is generated by a subservice user for execution by a subservice provider

3.2.10 message

request or report

3.2.11 notification

elementary constituent of a report than is generated by a subservice provider for interpretation by a subservice user

3.2.12 object path

combination of a repository path and a file name or directory name

3.2.13 on-board file system

system used to control data organised in files

3.2.14 on-board memory

logical memory space

NOTE The on-board memories can potentially be managed by different on-board processors. The mapping between the on-board memories and the physical memories is out of the scope of this Standard.

3.2.15 on-board parameter

lowest level of elementary data item on-board

NOTE A parameter has a unique interpretation.

3.2.16 report

message made of one or more notifications generated by a subservice provider for interpretation by a subservice user

NOTE This Standard identifies three types of reports:

- verification reports,
- data reports, and
- event reports.

3.2.17 repository path

logical path to where a file or a directory is located

NOTE A repository path can either represent a physical path such as a directory path within a file system or a logical path such as a mounted device (e.g.

"/mm1"pointing to a mass memory device), a directory within a mounted device (e.g. "/mm1/dir1").

3.2.18 request

message consisting of one or more instructions generated by a subservice user for execution by a subservice provider

3.2.19 routing notification

notification that is generated by a routing and reporting subservice provider as a consequence of a request routing verification

3.2.20 routing verification report

report generated by a routing and reporting subservice provider as a consequence of a request routing verification

NOTE The routing verification reports are generated by the application processes that are involved in the routing of a request between a subservice user and a subservice provider. The routing and reporting subservice generates a failed routing verification report to inform a subservice user of the impossibility of pursuing the routing of the request, e.g. because of corruption of that request during the routing.

3.2.21 service

functional element of the space system that provides a number of closely-related functions that can be remotely operated

NOTE Each service is composed of one or more subservices, where each subservice involves a subservice provider and one or more subservice users. A subservice provider is in charge of performing some space system functions. A subservice user is in charge of issuing requests for the execution of those functions and of processing the resulting feedback.

3.2.22 subservice

elementary constituent of a service composed of exactly one subservice provider and the related subservice users that are interacting through dedicated sets of messages

3.2.23 subservice entity

operational element of a subservice hosted by an application process that acts as subservice user or subservice provider

3.2.24 subservice provider

operational element of a subservice that is in charge of execution of the subservice requests and generation of the subservice reports

3.2.25 subservice user

operational element of a subservice that is in charge of initiating the subservice requests and processing the subservice reports

3.2.26 transaction

set of messages related to the execution of exactly one capability which are exchanged between a subservice user and a subservice provider

NOTE The different types of transactions defined in this Standard are:

- request related transaction,
- autonomous data reporting transaction, and
- event reporting transaction.

3.2.27 verification report

routing, acceptance or execution verification report

3.3 Abbreviated terms

For the purpose of this Standard, the abbreviated terms from ECSS-S-ST-00-01 and the following apply:

Abbreviation	Meaning
ANSI	American National Standards Institute
AOCS	attitude and orbit control subsystem
APID	application process identifier
ASCII	American standard code for information interchange
CCSDS	Consultative Committee for Space Data Systems
CDS	CCSDS day segmented
CPDU	command pulse distribution unit
CRC	cyclic redundancy code
CUC	CCSDS unsegmented code
ESA	European Space Agency
FDIR	fault, diagnostic, isolation and recovery
FMON	functional monitoring
GPS	global positioning system
ID	identifier
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
LSB	less significant bit
MAP	multiplexer access point
MIL-STD	United States military standard
MSB	most significant bit
OBCP	on-board control procedure
PCS	packet check sequence
PFC	packet field format code

Abbreviation	Meaning
PMON	parameter monitoring
PTC	packet field type code
PUS	packet utilization standard
RAM	random access memory
ST	service type
TAI	international atomic time
TC	telecommand
TM	telemetry
UTC	coordinated universal time

3.4 Nomenclature

The following nomenclature applies throughout this document:

- a. The word “shall” is used in this Standard to express requirements. All the requirements are expressed with the word “shall”.
- b. The word “should” is used in this Standard to express recommendations. All the recommendations are expressed with the word “should”.

NOTE It is expected that, during tailoring, recommendations in this document are either converted into requirements or tailored out.

- c. The words “may” and “need not” are used in this Standard to express positive and negative permissions, respectively. All the positive permissions are expressed with the word “may”. All the negative permissions are expressed with the words “need not”.
- d. The word “can” is used in this Standard to express capabilities or possibilities, and therefore, if not accompanied by one of the previous words, it implies descriptive text.

NOTE In ECSS “may” and “can” have completely different meanings: “may” is normative (permission), and “can” is descriptive.

- e. The present and past tenses are used in this Standard to express statements of fact, and therefore they imply descriptive text.

Context and background

4.1 Introduction

This Standard addresses the need to standardize the way the space system functions are defined when involved in an interaction between space and ground.

This Standard introduces the concept of PUS services, consisting of PUS subservices. The services and subservices formalise the closely related and self-contained set of space system functions and all related entities and interaction artifacts.

Each PUS subservice is composed of PUS subservice entities, each one playing either the role of a subservice provider or the role of a subservice user. Each PUS subservice entity is hosted by an application process on-board or on-ground.

As depicted in Figure 4-1, it is usually understood that the on-board application processes host the subservice providers and the ground application processes the subservice users but this standard does not constrain those relationships. For example, a ground equipment can host some subservice providers so that the equipment can be remotely controlled by a mission control centre, a payload can host some subservice users for controlling solid-state mass memories (e.g. using file management subservices).

No particular topography is assumed in this Standard for how application processes and hosted PUS subservice entities are implemented or distributed, neither is any topography precluded. Thus:

- for a given mission, there can be any number of on-board application processes (with a minimum of one), each one hosting any number of PUS subservice entities (with a constraint that a given application process can only host a single subservice entity provider of a given type of subservice);
- there are no restrictions on the mapping between application processes and the usual functional subdivision of a spacecraft into subsystems and payloads (at one extreme, with a simple spacecraft topology, there can be a single application process within a centralized data management system which hosts PUS services for all the other platform subsystems and payloads; at the other extreme, intelligent subsystems and payloads can each be served by their own independent application processes and PUS services);

- an application process can be implemented in software, firmware or hardware;
- an on-board computer can host one or more application processes or an application process can be distributed across two or more on-board computers.

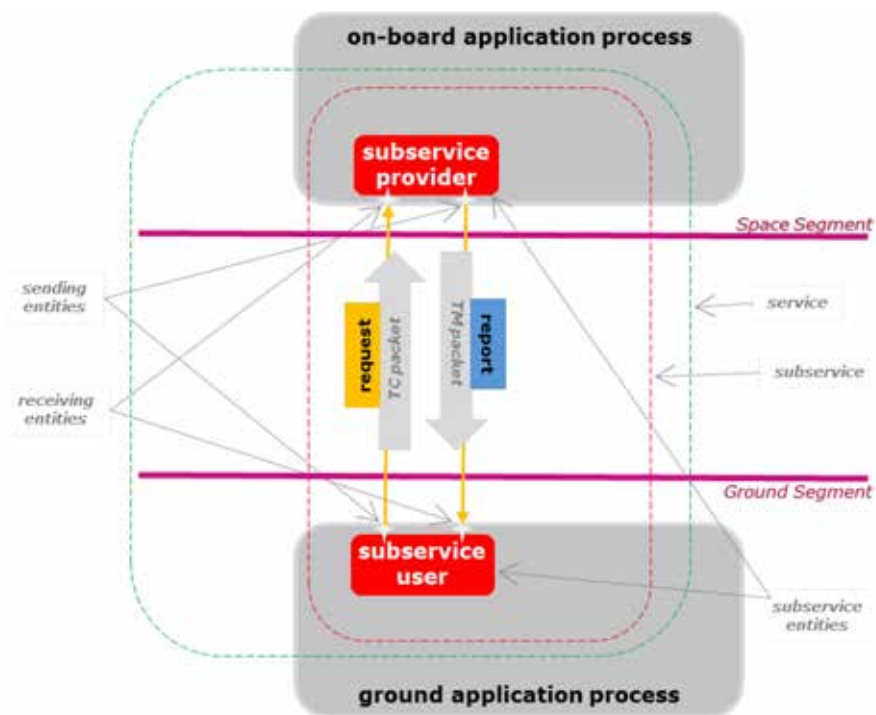


Figure 4-1 The space to ground PUS service system context

The information exchanged between a subservice user and subservice provider is termed a "**message**". A message is transmitted semantically unchanged by the transmission protocol that connects the subservice users and subservice providers.

A message sent by a subservice user to a subservice provider, to invoke the execution of on-board activities, is termed a "**request**". Each request contains one or more instructions, one for each activity to execute. A message sent by a subservice provider to a subservice user is termed a "**report**". Each report contains one or more notifications.

Three distinct categories of report are distinguished:

- the *verification reports*, which report on the routing, acceptance, start, progress and completion of the request execution;
- the *data reports*, which are generated:
 - on request, as one or multiple responses to the instructions of a request to elicit some specific service data,
 - autonomously as one or multiple reports activated by a request or, routinely, i.e. as part of a continuous reporting functionality;
- the *event reports*, which carry information related to the occurrences of the events detected by a service.

The request carries information used by the subservice provider to identify the subservice user that issued that request. This is especially interesting if several subservice users can send requests to a given subservice provider. It provides the means to the subservice provider to route the related verification reports and on-request data reports back to the subservice user who invoked the activity.

The routing of the autonomous data reports and of the event reports is either known implicitly (by design) or explicitly (e.g. by using an on-board routing table).

When messages (requests and reports) are exchanged between ground and space, they are encapsulated into CCSDS space packets, refer to clause 7.4.

Figure 4-2 provides an example of how PUS services can be deployed on-ground and on-board a spacecraft and how commanding with this Standard is understood.

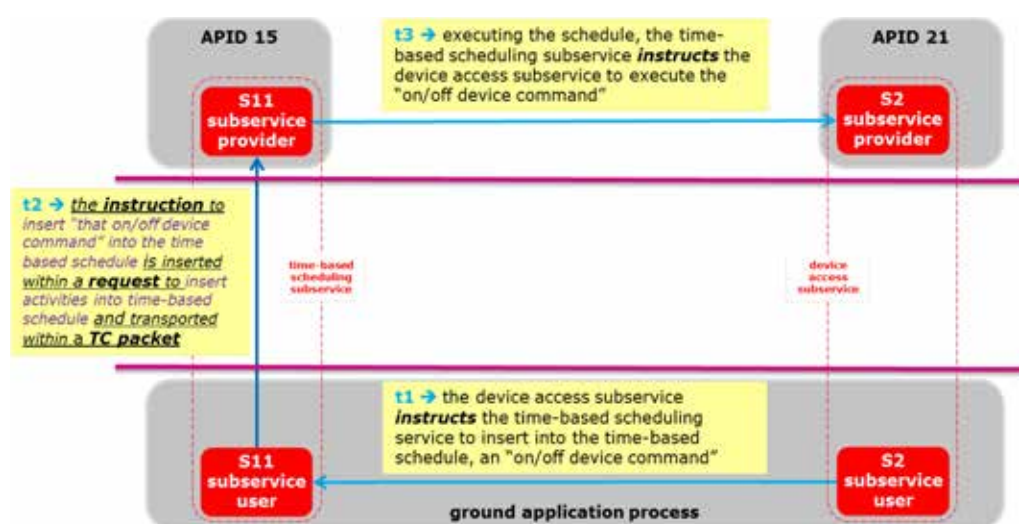


Figure 4-2 A PUS utilization example

The mechanisms which on-board application processes use to communicate with each other and with other on-board entities are implementation-dependent. Historically, spacecraft on-board interfaces have been specified and implemented on a project-by-project basis and any reuse of interfaces has usually been a by-product of reuse of existing spacecraft busses. While it is true that there are a limited number of physical interfaces available for use on-board a spacecraft, the services and access to these interfaces vary considerably between implementations. This Standard does not specify how requests, instructions, reports and notifications are implemented on-board or on-ground. It also does not specify who is in charge of encoding and decoding the telemetry and the telecommand packets.

4.2 Modelling the PUS

4.2.1 General

The overall PUS concept addressed in this Standard adopts a multi-layer modelling approach. The resulting model formalises the foundations of the PUS entities, in terms of system and interface requirements, together with their instantiation in space and on ground. Requirements can be applied as is or tailored for mission specific needs.

The multi-layer model, depicted in Figure 4-3 consists of:

- the *PUS foundation model*,
- the *standard service type model*,
- the *mission-specific service type model*, and
- the *space system service model*.

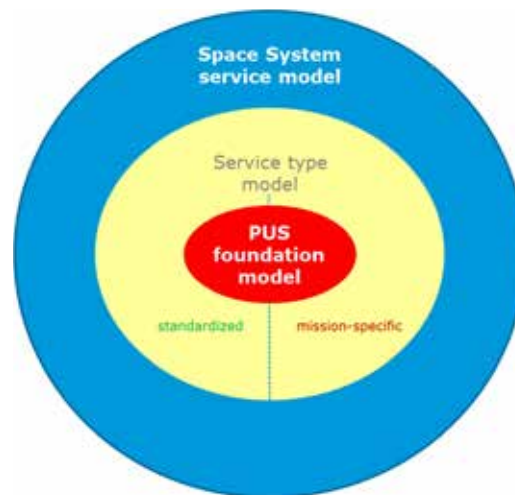


Figure 4-3 The PUS model

Central to the modelling approach is the concept of a *service type*, which is a container for all requirements related to an interaction between space, and ground capability dedicated to the fulfilment a service objective.

The system requirements, specified in clause 6, define the *semantics* of each service type including:

- the service type concept and related architecture;
- the message type concept and related architecture;
- the overall service type topology, focusing on the message exchange between the subservice users and the subservice providers.

The interface requirements define the layout and the format (i.e. the *syntax*) of the interaction protocol used between ground and space service entities. The interface requirements in clauses 7 and 8, specify:

- how requests are transported within PUS telecommand packets compliant with the CCSDS Space Packet Protocol;

- how reports are transported within PUS telemetry packets compliant with the CCSDS Space Packet Protocol.

4.2.2 The PUS foundation model

The PUS foundation model defines the PUS generic concepts, related terms and definitions and the business rules that:

- have been used by the authors of this Standard for producing the Standard service type model,
- apply to each mission that applies this Standard and define a level of tailoring of the service type model, and
- apply to the architects of the mission-specific space system (i.e. both the space segment and the ground segment) who develop and instantiate the tailored service type model for the mission.

The PUS foundation model addresses a generic and abstract definition of the PUS service type model that applies to each service type whether it is standardized or mission-specific.

The PUS foundation model contains the generic rules that apply to each mission that tailors this Standard:

- when creating mission-specific subservice types within a standardized service type;
- when adding mission-specific service type capabilities and related message types to standardized service types and subservice types;
- when creating mission-specific service types with associated subservice types, service type capabilities and related message types.

The PUS foundation model also contains the generic rules that apply to each implementation of a service type.

The PUS foundation model is specified in clause 5.

4.2.3 The service type model

4.2.3.1 Introduction

The PUS service type model includes:

- the standardized service types as specified in this Standard, and
- mission-specific extensions in terms of:
 - add-ons to the standard service types,
 - mission-specific service types.

4.2.3.2 Standard service types

This Standard contains the specification of a set of standard PUS service types. The choice of which service types are used by a new mission depends on the mission requirements. All service types are optional and a given service type

can be implemented at any of several distinct levels and its parameters and functions can be tailored.

The standard service types are listed in Table 4-1. They include:

- service types that provide basic functions such as collecting parameter statistics.
- service types that hold requests and release them to another service as appropriate. The time-based scheduling, the position-based scheduling and the event-action service types are examples of service types that hold and release requests following the occurrences of specified events.
- service types that provide standardized interfaces, for example to on-board devices, to an OBCP (on-board control procedure) engine or to an on-board file handling system.

The requirements specification of each of the standard service types consists of two parts:

- a system requirements specification contained in clause 6 that defines the actions of the service, including its behaviour when it receives a request. The system specification is concerned with the semantics of the requests and reports.
- an interface requirements specification contained in clause 8 that defines the syntax of the requests and reports for a service type. The fields in a request or report are defined using the standard PUS field types specified in clause 7.3.

Table 4-1: The standardized service types

service type	
name	ID
request verification	1
device access	2
housekeeping	3
parameter statistics reporting	4
event reporting	5
memory management	6
(reserved)	7
function management	8
time management	9
(reserved)	10
time-based scheduling	11
on-board monitoring	12
large packet transfer	13
real-time forwarding control	14
on-board storage and retrieval	15
(reserved)	16
test	17
on-board control procedure	18
event-action	19
parameter management	20
request sequencing	21

service type	
name	ID
position-based scheduling	22
file management	23
Note: The reserved service type identifiers were used in previous versions of this Standard. This Standard no longer promotes the use of these service types but does not preclude that existing implementations are reused for new missions.	

4.2.3.3 Mission-specific service types

When applying the PUS Standard, a mission instantiates this Standard by tailoring it for their needs. That instantiation results in a mission-specific packet utilization definition document that is rendered applicable to all partners involved in that mission.

The mission-specific packet utilization definition document contains the mission-specific service type model that includes:

- all PUS standardized service types considered suitable for use by that mission, each one tailored according to the mission needs,
- all mission-specific additional service types.

4.2.4 The space system service model

The space system service model results from the deployment of the service type model for a given mission, i.e. resulting from the space system architecture of that mission.

The space system service model contains the service topology in terms of:

- the instances of the service types and related hosting application processes, and
- for each instance, its full specification resulting from the tailoring of the related service type.

Deploying the space system service model implies for each on-board application process, selecting the services and related subservice providers to be hosted by that application process. This Standard specifies the following interdependencies between services:

- the request verification service is accessible to any other service within the same application process;
- the event reporting service and the large packet transfer service are accessible to any other service;
- the on-board monitoring service and the event-action service require the presence of an event reporting service;
- if an on-board control procedure service supports the capability for configuring the OBCP execution observability level, then that service requires the presence of an event-reporting service, refer to clause 6.18.4.8.

The PUS foundation model

5.1 Introduction

The PUS foundation model specifies a generic service and service type model in the form of a set of generic concepts with the associated business rules. The PUS foundation model provides rules that are applicable to any service type, i.e. standardized or mission-specific and any of their instances (i.e. the services).

As any service type definition relies on the PUS foundation model, the architectural consistency of each service type is ensured.

The PUS foundation model defines generic concepts and associated requirements related to two levels of abstractions, i.e.:

- The *generic service type abstraction level*, which specifies the set of generic object types and business rules that are required for ensuring the overall consistency of the service type model. This abstraction level includes all generic object types used to produce, by specialization, the standardized and the mission specific service types.
- The *generic service deployment abstraction level*, which specifies the set of generic object types and business rules that are required to capture the space system service model. This abstraction level includes all generic object types used to capture, by instantiation, the space system services resulting from the space system overall architecture.

The generic service type abstraction level specifies:

- the service type, the subservice type and the capability type;
- the subservice provider, the subservice user;
- the message type, i.e.:
 - the request type and the instruction type,
 - the report type and the notification type;
- the transaction type and its type-dependent definitions, i.e.:
 - for a request related transaction:
 - ◊ the request type,
 - ◊ the associated execution notification type, and
 - ◊ if some service data are generated in response to such a request, the related data report type;
 - for an autonomous data reporting transaction, the data report type;
 - for an event reporting transaction, the event report type.

The generic service deployment abstraction level specifies:

- the system context of the service, in terms of the involved system objects of relevance to the service functionality, e.g. the space segment, the ground segment, the application process, the on-board parameter, the on-board memory;
- the service, the subservice and the capability exposed by the subservice;
- the message, i.e.:
 - the request, the instruction slot and the instruction,
 - the report, the notification slot and the notification;
- the transaction.

5.2 Convention

This Standard uses two types of identification mechanisms:

- names for human communication, and
- identifiers for communicating with the spacecraft.

Names and identifiers are always unique in a given context.

The wider context that is considered by this Standard is the (single) spacecraft. This means that, for this Standard, when a name or an identifier is declared as unique within a given context, that context is implicitly understood as a context within the spacecraft.

5.3 The generic service type abstraction level

5.3.1 General

- a. Each service type shall be uniquely identified by exactly one service type name.
- b. Each service type shall be uniquely identified by exactly one service type identifier that is an unsigned integer greater than or equal to 1, and less than or equal to 255.

NOTE The service type identifiers are used in the telemetry packet secondary header (refer to clause 7.4.3.1) and in the telecommand packet secondary header (refer to clause 7.4.4.1), together with a message subtype identifier to uniquely identify a message type.

- c. Each standard service type shall have a service type identifier less than or equal to 127.

NOTE The standard service types are specified in the different versions of this Standard. When mission specific functionalities, identified by a mission specific service type, are considered adequate for

being standardized, a new standard service type is created. When a standard service type is no longer considered adequate for remaining a standard, that service type is removed from the Standard; its service type identifier is not reused.

- d. Each mission specific service type shall be associated with a service type identifier greater than or equal to 128.

5.3.2 Subservice type

- a. Each service type shall define at least one subservice type.

NOTE This Standard introduces the concept of subservices that group and isolate the functions of a service.

- b. Each subservice type shall be defined by exactly one service type.
- c. Each subservice type shall be uniquely identified by exactly one subservice type name.
- d. For each subservice type, whether the realization of that subservice type is implicitly required for each realization of the service type or required by tailoring shall be declared when specifying that subservice type.

NOTE 1 An example of a subservice type that is implicitly required is the "parameter monitoring" subservice type. Each realization of the "on-board monitoring" service type is implicitly required to include a realization of that subservice type, refer to requirement 6.12.2.1.1a and clause 6.12.3.

NOTE 2 An example of a subservice type that is required by tailoring is the "functional monitoring subservice", refer to requirement 6.12.2.1.2a and clause 6.12.4.

- e. For each subservice type, whether multiple realizations of that subservice type are allowed within a single service shall be declared when specifying that subservice type.

NOTE An example of a subservice type where multiple realizations are allowed within a single service is the "packet selection" subservice type, refer to requirement 6.15.2.1.2a and clause 6.15.4.

- f. For each subservice type, the observables shall be declared when specifying that subservice type.

NOTE These observables are on-board parameters that are provided by the related subservice, refer for example to the observables of the parameter monitoring subservice in clause 6.12.3.13.

5.3.3 Message type

5.3.3.1 General

- a. Each message type shall be uniquely identified by exactly one message type name.
- b. Each message type shall be uniquely identified by exactly one message type identifier.

NOTE These identifiers are used in the telemetry packet secondary header (refer to clause 7.4.3.1) and in the telecommand packet secondary header (refer to clause 7.4.4.1) to identify the type of messages transported by these packets but also in specific requests and reports, e.g. in the requests to add report types to the application process forwarding control table (refer to clause 6.14.3.4.1).

- c. Each message type identifier shall be composed of:
 - 1. the service type identifier of the service type that contains that message type;
 - 2. a message subtype identifier that uniquely identifies that message type within that service type.
 - d. Each message subtype identifier shall be an unsigned integer greater than or equal to 1, and less than or equal to 255.
 - e. Each standard message type identifier shall have a message subtype identifier less than or equal to 127.
- NOTE** The standard message type identifiers are the identifiers specified in this Standard.
- f. Each mission specific message type that belongs to a standard service type shall have a service subtype identifier greater than or equal to 128.
 - g. Each message type shall either be:
 - 1. a request type, or
 - 2. a report type.

NOTE 1 For item 1, refer to clause 5.3.5.2.

NOTE 2 For item 2, refer to clause 5.3.3.3.

5.3.3.2 Request type

- a. Each request type shall define one or more instruction types.

NOTE 1 An example of a request type that defines exactly one instruction type is the "modify parameter monitoring definitions" request type specified in clause 6.12.3.9.4. The single related instruction type is the "modify a parameter monitoring definition" instruction type specified in requirement 6.12.3.9.4c.

NOTE 2 An example of a request type that defines more than one instruction type is the "report parameter

monitoring definitions" request type specified in clause 6.12.3.10. The related instruction types are specified in requirement 6.12.3.10b, i.e.:

- the "report a parameter monitoring definition" instruction type,
- the "report all parameter monitoring definitions" instruction type.

NOTE 3 The decision to link several instruction types to the same request type instead of having a request type for each instruction type is an operational issue. For example, if an instruction type acts on one instance of a system object and another instruction type on all instances of that system object, if the operational criticality of the "one" instruction differs from the operational criticality of the "all" instruction, this Standard recommends to define two request types.

- b. Each instruction type shall be defined for exactly one request type.
- c. Each instruction type shall be uniquely identified by exactly one instruction type name.
- d. For each request type and for each instruction type of that request type, whether that request type provides a single instruction slot or multiple instruction slots for that instruction type shall be declared when specifying that request type.

NOTE For some instruction types, it make sense to allow multiple instructions in a request and, for others, it does not. Although an instruction type offers the possibility to have multiple instructions of that type inside a single request, that multiple instructions capability is a decision taken at request type level.

An example of an instruction type that offers the possibility to have multiple instructions inside a single request is the "report a parameter monitoring definition" instruction type specified in requirement 6.12.3.10b for which the request to "report parameter monitoring definitions" defined in clause 6.12.3.10 provide the capability to have multiple instructions inside a single request.

An example of an instruction type for which it does not make sense to allow multiple instructions in a request is the "report all parameter monitoring definition" instruction type also specified in requirement 6.12.3.10b.

- e. For each request type that contains several instruction types, the allowed combinations of instruction types that can be used in a request of that request type shall be declared when specifying that request type.

NOTE An allowed combination of instruction types means that the realizations of two or more of those instruction types can be merged in a single request of the corresponding request type, see for example the add report types to the application process storage-control configuration specified in clause 6.15.4.4.1.

- f. For each instruction type, the instruction arguments used by that instruction type, their definition and their ordering within the instruction type shall be declared when specifying that instruction type.
- g. For each request type that provides multiple instruction slots, if that request type constrains the scope of the instructions that can be issued within a request of that type, the argument or set of arguments of the related instruction types that define that scope shall be grouped together in the definition of the request type.

NOTE This requirement avoids constructing and issuing a request with multiple times the same instruction argument value or set of argument values. For example, the request type to time-shift scheduled activities identified by request identifier has a time-offset argument that precedes the instruction slots. That time offset applies to each instruction in the request (as specified in clause 6.11.9.3).

- h. For each request type, the definition of the request arguments provided by that request type, their definition and their ordering within the request type shall be declared when specifying that request type.

NOTE A request type argument can be an instruction type argument (or set of instruction type arguments) as specified in requirement 5.3.3.2g, or a directive argument (or set of directive arguments) specifying, for example,

- an on-board condition to allow executing the instructions of the requests of that type,
- a mode to set (e.g. the configuration execution flag of the request to apply parameter functional reporting configurations, refer to clause 6.3.5.3).

5.3.3.3 Report type

- a. Each report type shall either be:
1. a data report type,
 2. a verification report type, or
 3. an event report type.

NOTE 1 For item 1, an example of a data report type is the housekeeping parameter report type specified in clause 6.3.3.3.

NOTE 2 For item 2:

- the verification report types are those specified in clause 6.1, i.e. the request verification service type.
- the verification reports are used in the request related transactions, refer to clause 5.3.5.2.

NOTE 3 For item 3, the event report types are those specified in clause 6.5.4, see also clause 5.3.5.4.

- b. Each report type shall define exactly one notification type.

NOTE If a report type is associated to a request related transaction type (i.e. that report type is a response type) and associated to an autonomous data reporting transaction type (i.e. that report type is also an autonomous data report type), the same notification type is used for both transaction types.

- c. Each notification type shall be defined for exactly one report type.
- d. Each notification type shall be uniquely identified by exactly one notification type name.
- e. For each report type and for each notification type of that report type, whether that report type provides a single notification slot or multiple notification slots for that notification type shall be declared when specifying that report type.

NOTE For some notification types, it makes sense to allow multiple notifications in a report. For others, it does not. Although a notification type offers the possibility to have multiple notifications of that type inside a single report, that multiple notifications capability is a decision taken at report type level.

An example of a notification type that offers the possibility to have multiple notifications inside a single report but for which it is explicitly required to have only one notification per report is the housekeeping parameter report structure report specified in clause 6.3.3.6.

5.3.4 Capability type

- a. Each subservice type shall define at least one capability type.

NOTE Each capability type defines one or more interrelated functions of the subservice type. A capability type can represent:

- a single function, e.g. for "the capability to distribute on/off device commands" specified in clause 6.2.4.2;
- a set of two or more exclusive-or related functions, e.g. for the exclusive-or constraint to use either the CUC format or the CSD format (but not both) when reporting the on-board time, refer to requirement 6.9.4.1a;
- a set of two or more inclusive-or related functions, e.g. for the inclusive-or constraint to provide at least one means to load OBCPs, refer to requirement 6.18.4.4.1a;
- a set of interrelated functions, e.g. for the capability to enable and disable the scrubbing of a memory specified in clause 6.6.6.1.4 and 6.6.6.1.5 whereas the decision to provide the capability to enable the scrubbing of a memory implies to provide the capability to disable the scrubbing of a memory (refer to requirement 6.6.6.1.5a).

- b. For each capability type defined by a subservice type, the applicability constraints of that capability type shall be declared when specifying that subservice type.

NOTE The applicability constraint of each standardized capability type is specified in clause 6 (see also Annex C). For example:

- a "minimum" applicability constraint means that each related subservice provides that capability (see for example Table C-1);
- a "by declaration" applicability constraint means that for each related subservice, whether that capability is provided by that subservice is a decision to take when specifying that subservice (See for example requirement 6.3.3.4.1a);
- an "implied by another capability type" applicability constraint means that if a subservice provides that other capability then that subservice also provides that implied capability (see for example requirement 6.3.3.4.2a);
- a "by declaration and only if another capability type is provided" applicability constraint means

that the decision to include that capability depends on the decision taken for that subservice to provide that other capability (see for example requirement 6.2.5.3a and the associated note).

Applicability constraints can also be defined for a set of capability types. For example:

- an exclusive-or applicability constraint means that a subservice can provide at most one of the related capabilities (see for example requirement 6.9.4.1a);
- an inclusive-or applicability constraint means that a subservice provides at least one of the related capabilities (see for example requirement 6.2.3a).

5.3.5 Transaction type

5.3.5.1 General

- a. Each transaction type shall be defined by exactly one capability type.
- b. Each transaction type shall either be:
 1. a request related transaction type,
 2. an autonomous data reporting transaction type, or
 3. an event reporting transaction type.

NOTE 1 For item 1, refer to clause 5.3.5.2.

NOTE 2 For item 2, refer to clause 5.3.5.3.

NOTE 3 For item 3, refer to clause 5.3.5.4.

5.3.5.2 Request related transaction type

5.3.5.2.1 General

- a. Each request related transaction type shall involve exactly one request type.

NOTE The verification report types introduced in clause 5.3.3.3 are involved in the request related transaction types as a consequence of the execution verification profile specified in clause 5.3.5.2.3.

- b. Each request type shall be involved in exactly one request related transaction type.

5.3.5.2.2 Response type

- a. Each request type shall be linked to at most one data report type.

NOTE 1 An example of a request type that is linked to a data report type is the "report parameter monitoring definitions" request type. The linked

data report type, playing the role of the response type, is the "parameter monitoring definition report", refer to requirement 6.12.3.10a.

NOTE 2 As stated in requirement 5.3.3.3b, each data report type defines exactly one notification type. The link that exists between a request type and a report type implies that each instruction type defined by that request type is linked to the notification type defined by that report type.

- b. For each instruction type that is linked to a notification type, whether a realization of that instruction type can cause the generation of multiple notifications shall be declared when specifying that instruction type.

NOTE 1 An example of an instruction type whose realization can cause the generation of multiple notifications is the "report all parameter monitoring definitions" instruction type, refer to requirement 6.12.3.10h.

NOTE 2 An example of an instruction type whose realization causes the generation of a single notification is the "report a parameter monitoring definition" instruction type, refer to requirement 6.12.3.10g.

5.3.5.2.3 Execution verification profile

- a. For each request type, the pre-conditions to verify prior to starting the execution of each request of that type shall be declared when specifying that request type.

NOTE 1 An example of such a request-type-specific pre-conditions is the existence of the parameter functional reporting definition indicated by the argument of the "add parameter report definitions to a parameter functional reporting definition" request type, refer to requirement 6.3.5.6.1c.1.

NOTE 2 This Standard does not list the checks to perform to avoid the execution of a request that has no effect if the absence of such check causes no operational ambiguity. It is for the mission to decide if and where to perform the checks, i.e. on-board or on-ground

- b. For each instruction type, the pre-conditions to verify prior to starting the execution of each instruction of that type shall be declared when specifying that instruction type.

NOTE An example of such instruction-specific pre-conditions is the existence within the parameter functional reporting definition of the parameter report definition indicated by the instruction-specific argument of the instruction to "add a parameter report definition to a parameter

functional reporting definition", refer to requirement 6.3.5.6.1f.1.

- c. For each request type that provides a multiple instruction slots capability, whether the subservice verifies the suitability of all instructions contained within each request of that type before authorizing the start of execution of that request shall be declared when specifying that request type.

NOTE 1 This Standard applies the operational concept that verifying on-board the suitability of all instructions before authorizing the start of execution of a request implies the failure of that start of execution if not all instructions are suitable for execution.

NOTE 2 An example of a request type whose realizations can only be executed if all their instructions are suitable for execution is the request to "load raw memory data areas", refer to requirement 6.6.3.3.2e.

NOTE 3 An example of a request type whose realizations can be executed without ensuring that all their instructions are suitable for execution is the request to "enable parameter monitoring definitions", refer to requirement 6.12.3.6.2d. The instructions contained within such a request are by nature independent.

- d. For each instruction type, the conditions to verify during the execution of each instruction of that type shall be declared when specifying that instruction type.
- e. For each instruction type, the post-conditions to verify at the end of the execution of each instruction of that type shall be declared when specifying that instruction type.
- f. For each request type, the post-conditions to verify at the end of the execution of each request of that type shall be declared when specifying that request type.
- g. For each request type, the execution verification profile used to report the start, progress and completion of execution of each request of that type shall be declared when specifying that request type.

NOTE The execution verification profile can include any of the following:

- for each request-specific successful start of execution condition to notify, a code value that refers to that condition;
- for each request-specific failed start of execution condition to notify, a failure notice made of a code value that refers to that condition together with any number of associated parameters whose values are reported to support the processing of that failed execution notification;

- for each instruction-specific successful start of execution condition to notify, a code value that refers to that condition;
 - for each instruction-specific failed start of execution condition to notify, a failure notice made of a code value that refers to that condition together with any number of associated parameters whose values are reported to support the processing of that failed execution notification;
 - for each instruction-specific successful progress of execution condition to notify, a code value that refers to that condition;
 - for each instruction-specific failed progress of execution condition to notify, a failure notice made of a code value that refers to that condition together with any number of associated parameters whose values are reported to support the processing of that failed execution notification;
 - for each instruction-specific successful completion of execution condition to notify, a code value that refers to that condition;
 - for each instruction-specific failed completion of execution condition to notify, a failure notice made of a code value that refers to that condition together with any number of associated parameters whose values are reported to support the processing of that failed execution notification;
 - for each request-specific successful completion of execution condition to notify, a code value that refers to that condition;
 - for each request-specific failed completion of execution condition to notify, a failure notice made of a code value that refers to that condition together with any number of associated parameters whose values are reported to support the processing of that failed execution notification.
- h. Each progress of execution notification shall provide the means to uniquely identify the instruction that progress of execution is notified.
- NOTE This identification is used by the subservice user that has initiated the execution of that instruction.
- i. For each instruction type, the functionality that the subservice performs when executing an instruction of that type shall be declared when specifying that instruction type.

NOTE An example of such subservice functionality can be found in 6.3.5.6.1i.

- j. For each request type, the request-specific functionality that the subservice performs when executing a request of that type shall be declared when specifying that request type.

5.3.5.3 Autonomous data reporting transaction type

- a. Each autonomous data reporting transaction type shall involve exactly one data report type.

NOTE 1 Examples of autonomous data report types are:

- the housekeeping parameter report type (refer to clause 6.3.3.3),
- the diagnostic parameter report type (refer to clause 6.3.4.3),
- the check transition report type (refer to clause 6.12.3.7).

NOTE 2 It is noted that some data reports can be generated autonomously but also in response to specific requests. This is for example the case of the housekeeping parameter reports that can be generated periodically according to a collection interval (refer to requirement 6.3.3.2c), but are also generated as the response of a request to generate a one shot report for housekeeping parameter report structures (refer to clause 6.3.3.7).

- b. Each data report type shall be involved in at most one autonomous data reporting transaction type.

5.3.5.4 Event reporting transaction type

- a. Each event reporting transaction type shall involve exactly one event report type.

NOTE This Standard defines four types of event reports according to the severity level of their associated events:

- the informative event report type,
- the low severity event report type,
- the medium severity event report type, and
- the high severity event report type.

The message subtype identifier gives the severity level of the event report types, refer to clause 6.5.4. For example, all event reports for low severity events have the same message type. i.e. the same combination of service type identifier and message subtype identifier. There is no means, at event report type level, to identify the event that is associated to the related event reports. For that

event association, this Standard defines the concept of event definitions. Each event definition is associated to a single event and a single event report type. Each event definition is uniquely identified by the combination of the application process that generates the corresponding event reports and an event definition identifier that is unique within the context of that application process (refer to clause 6.5.3).

- b. Each event report type shall be involved in exactly one event reporting transaction type.

5.3.6 Tailoring the generic service type abstraction level

- a. Tailoring the generic service type abstraction level shall consist of:
 1. adding mission-specific service types;
 2. adding mission-specific subservice types;
 3. adding mission-specific capability types;
 4. adding mission-specific message types.

NOTE Reducing the standardized functional capabilities offered by the generic service type abstraction level (i.e. clause 5.3) is not recommended since it can negatively affect the reuse of existing elements (hardware or software).

5.4 The generic service deployment abstraction level

5.4.1 Introduction

The services are functional entities that involve both ground elements and on-board elements.

A service is composed of one or more subservices. Each subservice involves:

- one or more subservice users, each one hosted by an application process that resides on-ground or on-board, and
- exactly one subservice provider that is usually hosted by an on-board application process.

The communication between the subservice entities (i.e. a subservice user and a subservice provider) consists of exchanging messages between these entities. When messages are exchanged between the ground segment and the space segment, these messages are transported in CCSDS packets as specified in clause 7.

5.4.2 Application process

5.4.2.1 General

- a. Each application process shall either be:
 1. an on-board application process, or
 2. a ground application process.
- b. Each application process that hosts at least one subservice provider shall be identified by an application process identifier that is unique across the system that hosts that subservice provider.

NOTE 1 This Standard acknowledges that the same application process identifier can be used to identify several application processes. This is for example the case during the space system development where different representations of a given application process are used, e.g. a simulated version of an application process used for testing the ground segment but also during operations, e.g. in case of cold redundancy.

NOTE 2 The system introduced in this requirement can be, for example, the spacecraft that hosts the on-board application process. The concept of system identifier is also used in this Standard to uniquely identify that system across the overall space system. This Standard does not further elaborate on this system concept and its identifier.

- c. Each application process identifier shall be an unsigned integer that is less than or equal to 2046.

NOTE 1 This application process identifier is used to identify the on-board application process that is the destination for a request and the source for a report.

NOTE 2 The APID 2047 is reserved for idle packets. The APID 0 is reserved for spacecraft time packets. Other APID values are reserved, refer to the space assigned numbers authority registry (see bibliography).

- d. Each application process that hosts at least one subservice user shall be identified by an application process user identifier that is unique within the context of the overall space system.

NOTE 1 The subservice users are in charge of issuing requests and processing reports. As such, an application process that can only receive reports also has an application process user identifier.

NOTE 2 The application process user identifier is used:

- as "source identifier" for any request generated by that application process (see also the source

ID field of the telecommand packet secondary header specified in requirement 7.4.4.1b), and

- as "destination identifier" for any report whose final destination is that application process (see also the destination ID field of the telemetry packet secondary header specified in requirement 7.4.3.1b).

NOTE 3 This Standard acknowledges that the same application process user identifier can be used to identify several application processes, e.g. in case of cold redundancy.

- e. Each application process user identifier shall be an unsigned integer that is greater than or equal to 0, and less than or equal to 65535.
- f. For each report that it generates, each on-board application process shall time tag that report using the on-board reference time.
- g. For each application process, whether that application process time tags the reports before collecting the values of the constituting parameters or after shall be declared when specifying that application process.

NOTE When a report contains parameter values acquired at different times (e.g. housekeeping reports with multiple samples of the same parameter), the acquisition time of each set of parameter values can be deduced from the time tag of the report.

- h. For each application process, whether that application process provides the capability to report the status of the on-board time reference used when time tagging reports shall be declared when specifying that application process.
- i. For each application process, whether that application process provides the capability to count the type of generated messages per destination and report the corresponding message type counter shall be declared when specifying that application process.
- j. Each application process that provides the capability to count the type of generated messages per destination and report the corresponding message type counter shall maintain, per destination, a counter for each message type that it generates.

5.4.3 Interfaced system objects

5.4.3.1 Introduction

Each service interacts with objects of the overall space system. These system objects are either:

- defined within the scope of a service, or
- defined externally, e.g. an on-board memory that is defined at spacecraft level and used by several services.

The system objects that are defined within the scope of a service are maintained by that service and their visibility is often limited to that service. They expose properties that are used by the service to perform its functionality.

The system objects that are externally defined have their own existence independently of any service. They expose properties that are accessed by some services for the purpose of e.g. performing the service functionality, monitoring and controlling those system objects.

The system objects introduced in this Standard are:

- the on-board parameters, refer to clause 5.4.3.2;
- the on-board memories, refer to clause 5.4.3.3;
- the virtual channels, refer to clause 5.4.3.4;
- the on-off devices, refer to clause 6.2.4;
- the registries, refer to clause 6.2.5;
- the CPDUs, refer to clause 6.2.6 and clause 9;
- the physical and the logical devices, refer to clause 6.2.7.1.1 and clause 6.2.7.2.1;
- the housekeeping parameter report structures, refer to clause 6.3.3.2;
- the diagnostic parameter report structures, refer to clause 6.3.4.2;
- the parameter functional reporting definitions, refer to clause 6.3.5.2;
- the event definitions, refer to clause 6.5.3;
- the functions, refer to clause 6.8.3.1;
- the time-based sub-schedules, refer to clause 6.11.5.1;
- the time-based scheduling groups, refer to clause 6.11.6.1;
- the parameter monitoring definitions, refer to clause 6.12.3.3;
- the functional monitoring definitions, refer to clause 6.12.4.2;
- the packet stores, refer to clause 6.15.3.1;
- the on-board control procedures, refer to clause 6.18.4.1;
- the request sequences, refer to clause 6.21.4;
- the position-based sub-schedules, refer to clause 6.22.7.1;
- the position-based scheduling groups, refer to clause 6.22.8.1;
- the on-board file systems, refer to clause 5.4.5.

5.4.3.2 On-board parameter

- a. Each on-board parameter shall be identified by exactly one on-board parameter identifier that is unique across the entire spacecraft.

NOTE 1 An on-board parameter represents e.g. a measurement taken from an on-board sensor or a software parameter held in memory.

NOTE 2 A service may need to acquire a reading of an on-board parameter for the purposes of its routine

activity (for example, to monitor its value, to use its value to determine the validity of another on-board parameter, to use its value in a calculation etc.).

NOTE 3 The "baseline" set of on-board parameters is defined during the spacecraft design process. However, the flexibility can also exist to define new parameters in orbit or to change the definition of an existing on-board parameter or to set the value of an on-board parameter (refer to clause 6.20). This capability is of course restricted to software parameters held in on-board memory and the on-board software design can additionally have built-in protections to ensure against the overwriting of essential on-board parameters.

- b. The set of on-board parameter minimum sampling intervals used to access the on-board parameters shall be declared when specifying the spacecraft architecture.

NOTE This Standard foresees that different spacecraft subsystems may use different on-board parameter minimum sampling intervals, e.g. the platform uses a parameter minimum sampling interval of 125 ms but the payload uses an interval of 500 ms.

- c. Each on-board parameter shall be associated to exactly one on-board parameter minimum sampling interval.

NOTE 1 This on-board parameter minimum sampling interval is used as the unit for expressing time intervals used by the subservices that access the on-board parameters, for example, the housekeeping or monitoring services. refer also to requirement 6.12.3.3f.

NOTE 2 This requirement does not imply that for each on-board parameter, one can associate an on-board parameter minimum sampling interval but that such an interval is associated to a group of parameters, e.g. all parameters of a platform, all parameters of a payload.

- d. All on-board parameters accessed by an application process shall be associated to the same on-board parameter minimum sampling interval.

5.4.3.3 On-board memory

5.4.3.3.1 General

- a. Each on-board memory shall be identified by exactly one on-board memory identifier.

NOTE 1 The on-board memory concept introduced in this Standard is for logical memories, i.e. any logical memory space, potentially managed by different

on-board processors. The mapping with physical memories is out of the scope of this Standard.

NOTE 2 Each physical memory is associated to a memory smallest addressable unit that specifies the minimum number of bytes that can be addressed. Each logical memory, identified by the memory identifier, is associated to a memory access alignment constraint that specifies the minimum number of bytes used by the services to address the corresponding physical memory.

NOTE 3 This Standard does not preclude that the same memory identifier is used by several on-board memories provided that they cannot be accessed at the same time, e.g. in the case of memory cold redundancy.

NOTE 4 Access to a given memory can be by either absolute addressing or relative addressing. For relative addressing, a base address (either an explicit address or a symbolic address, such as a table name) and an offset from this base address are specified.

- b. At any time, each on-board memory identifier shall uniquely identify exactly one on-board memory that is unique across the entire spacecraft.
- c. For each on-board memory, the following characteristics of that memory shall be declared when specifying that memory:
 - 1. the memory access alignment constraint;
 - 2. the memory size, in bytes;
 - 3. the allowed operations;
 - 4. the addressing scheme.

NOTE For item 4, refer to clause 5.4.3.3.2.

- d. When declaring the characteristics of an on-board memory, the allowed operations shall be one of the following:
 - 1. "read only";
 - 2. "read and write";
 - 3. "write only".
- e. For each on-board memory, whether scrubbing that memory is supported shall be declared when specifying that memory.
- f. For each on-board memory, whether write protecting that memory is supported shall be declared when specifying that memory.

5.4.3.3.2 Addressing scheme

- a. For each on-board memory, whether an absolute addressing scheme for that memory is exposed in the space to ground interface shall be declared when specifying that memory.
- b. Absolute addressing implies that the memory addresses and related offsets shall be expressed in bytes.

- c. For each on-board memory, whether a base plus offset addressing scheme for that memory is exposed in the space to ground interface shall be declared when specifying that memory.

NOTE Base plus offset addressing means that the memory addresses are byte offsets from a base reference. A base reference gives (explicitly or implicitly) the address within the memory which is used as the byte-zero reference for the offset. The base reference can itself be an absolute address or a symbolic address e.g. the name of a table, a parameter set or a file whose absolute address is implicitly known on-board.

- d. Base plus offset addressing implies that the base references when expressed as an absolute address and related offsets shall be expressed in bytes.

NOTE Base plus offset addressing implies that the byte offsets are offsets from the first byte of the referenced area within the object referenced by the base independently of the actual physical storage within the memory used to store the related data.

5.4.3.4 Virtual channel

- a. The list of virtual channels defined for downlinking reports and their characteristics shall be declared when specifying the space to ground interface.

NOTE For the virtual channel, refer to ECSS-E-ST-50-03. See also clause 7.1.2.2.

- b. For each virtual channel defined for downlinking reports, the virtual channel identifier used to refer to that virtual channel shall be declared when specifying that virtual channel.

5.4.4 Checksum algorithm

- a. For each checksum algorithm used on-board, the list of subservice providers that use that checksum algorithm shall be declared when specifying the spacecraft architecture.

NOTE 1 This requirement is justified by the system need to ensure that all subservice providers that provide means to checksum a specific data object use the same checksum algorithm. For example, if a file contains an OBCP that can be checksummed by the OBCP service and that file is also managed by a memory service, the same checksum algorithm is used by both services.

NOTE 2 The checksum algorithm implies the type of checksum i.e. ISO or CRC, and the size of the checksum.

NOTE 3 The checksum algorithm to use to checksum all telemetry packets and the checksum algorithm to use for all telecommand packets are specified in requirements 7.4.3.2e and 7.4.4.2d.

5.4.5 On-board file system

- a. Each on-board file system shall be identified by exactly one on-board file system identifier that is unique across the entire spacecraft.

NOTE For the on-board file system, refer also to clause 6.23.

- b. Each object in an on-board file system shall be uniquely identified by an object path that is the combination of a repository path and an object name.

NOTE The term object refers to a file or to a directory.

- c. For each on-board file system, whether that file system supports files with unbounded size shall be declared when specifying that file system.

NOTE A file of unbounded size means that the file is only limited by the actual available physical memory size.

- d. The set of file attributes supported by each on-board file system shall be declared when specifying that file system.

NOTE For example, the file type, its creation date.

- e. For each on-board file system, whether that file system provides the capability to lock files shall be declared when specifying file system.

- f. An on-board file system shall not be accessed by more than one file management service.

5.4.6 Service

- a. Each service shall be of exactly one service type.
- b. For each subservice type whose realization is implicitly required, each service of the related service type shall provide at least one subservice of that subservice type.

NOTE An example of a subservice type whose realization is implicitly required is the parameter monitoring subservice type of the on-board monitoring service type, refer to requirement 6.12.2.1.1a.

- c. For each subservice type whose realization is required by tailoring and for each service of the service type that defines that subservice type, whether the realization of that subservice type is required for that service shall be declared when specifying that service.

NOTE An example of a subservice type whose realization is required by tailoring is the functional monitoring subservice type of the on-board

monitoring service type, refer to requirement 6.12.2.1.2a.

- d. For each subservice type that allows multiple realizations within a single service, each realization of that subservice type shall be declared when specifying that service.

NOTE An example of a subservice type that allows multiple realizations within a single service is the packet selection subservice type of the on-board storage and retrieval service type, refer to requirement 6.15.2.1.2a.

- e. The service topology of the overall space system shall be declared when specifying the space system architecture.

NOTE The service topology includes:

- the list of subservices provided by each service,
- the on-board service topology, i.e. for each service, the subservice provider of each related subservice and the on-board subservice users, if any, of each subservice, and
- the ground service topology, i.e. for each service, the subservice users of each related subservice.

5.4.7 Subservice

5.4.7.1 General

- a. Each subservice shall be of exactly one subservice type.
- b. Each subservice shall belong to exactly one service.

NOTE The type of a subservice is one of the subservice types defined for the related service type.

5.4.7.2 Subservice entity

5.4.7.2.1 General

- a. Each subservice entity shall belong to exactly one subservice.
- b. Each subservice entity shall be hosted by exactly one application process.
- c. Each subservice entity shall be either a subservice user or a subservice provider.

NOTE A subservice entity is identified by the subservice that it belongs to and the application process that hosts it.

5.4.7.2.2 Subservice provider

- a. Each subservice shall provide exactly one subservice provider.

NOTE A subservice provider is an operational element of a subservice that is in charge of execution of the

subservice requests and generation of the subservice reports. The subservice providers are usually hosted by the on-board application processes.

5.4.7.2.3 Subservice user

- a. Each subservice shall provide at least one subservice user.

NOTE A subservice user is an operational element of a subservice that is in charge of initiating the subservice requests and processing the subservice reports. The subservice users are either hosted by the ground application processes or the on-board application processes.

5.4.8 Capability

- a. Each subservice shall provide at least one subservice capability.
- b. For each subservice and for each capability type defined by the corresponding subservice type, the inclusion of the related capability in that subservice shall comply with the applicability constraints of that capability type.

NOTE For the applicability constraints of a capability type, refer to requirement 5.3.4b.

5.4.9 Failed progress of execution

- a. For each request type for which a failed progress of execution can be reported, whether the corresponding failed progress of execution notifications are reported within failed progress of execution verification reports or as part of the completion of execution verification report for the related requests shall be declared when specifying the request type related subservice.

NOTE This requirement also applies to the standardized request types specified in clause 6 that do not specify the related failed progress of execution notifications reporting policy.

5.4.10 Transaction

- a. Each subservice shall provide the means to manage all transactions that it initiates according to the mission operational requirements.

NOTE A transaction is either:

- a request related transaction,
- an autonomous data reporting transaction, or
- an event reporting transaction.

- b. Each transaction shall be initiated and maintained by exactly one subservice.

NOTE Each transaction involves one or more messages exchanged between a subservice user and a subservice provider.

A request related transaction involves:

- a request,
- depending on the acknowledgement specified for that request (refer to clause 5.4.11.2.2) and the execution verifications of that request (refer to clause 5.4.11.2.3), zero or more verification reports,
- depending on the successful execution of the instructions contained within that request, if that request type is linked to a response type, one or more responses (refer to clause 5.3.5.2).

An autonomous data reporting transaction involves an autonomous data report.

An event reporting transaction involves an event report.

5.4.11 Message

5.4.11.1 General

- a. Each message shall be of a single message type.

NOTE The message type is specified in clause 5.3.3. A message is either a request or a report.

5.4.11.2 Request

5.4.11.2.1 General

- a. Each request shall be generated by exactly one subservice user.

NOTE 1 By convention, a request is said to be generated by the application process that hosts the subservice user that generates that request.

NOTE 2 If the application process that generates the request is a ground application process, by convention, the request is also said to be generated by

- the monitoring and control system that hosts that application process,
- the ground segment.

- b. Each request shall be addressed to exactly one subservice provider.