



# Space engineering

---

## Ground systems and operations — Telemetry and telecommand packet utilization

Published by: ESA Publications Division  
ESTEC, P.O. Box 299,  
2200 AG Noordwijk,  
The Netherlands

ISSN: 1028-396X

Price: € 30

Printed in The Netherlands

Copyright 2003 © by the European Space Agency for the members of ECSS

---

## Foreword

This Standard is one of the series of ECSS Standards intended to be applied together for the management, engineering and product assurance in space projects and applications. ECSS is a cooperative effort of the European Space Agency, national space agencies and European industry associations for the purpose of developing and maintaining common standards.

Requirements in this Standard are defined in terms of what shall be accomplished, rather than in terms of how to organize and perform the necessary work. This allows existing organizational structures and methods to be applied where they are effective, and for the structures and methods to evolve as necessary without rewriting the standards.

The formulation of this Standard takes into account the existing ISO 9000 family of documents.

This Standard has been prepared by the ECSS Working Group for Ground Systems and Operations, reviewed by the ECSS Engineering Panel and approved by the ECSS Steering Board.

*(This page is intentionally left blank)*

---

## Introduction

ECSS-E-50 and the CCSDS recommendations for packet telemetry and telecommand address the end-to-end transport of telemetry and telecommand data between user applications on the ground and application processes on-board the satellite, and the intermediate transfer of these data through the different elements of the ground and space segments.

This packet utilization standard (PUS) complements these 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.

*(This page is intentionally left blank)*

---

# Contents

<b>Foreword</b> .....	<b>3</b>
<b>Introduction</b> .....	<b>5</b>
<b>1 Scope</b> .....	<b>13</b>
<b>2 Normative references</b> .....	<b>15</b>
<b>3 Terms, definitions and abbreviated terms</b> .....	<b>17</b>
3.1 Terms and definitions .....	17
3.2 Abbreviated terms .....	20
<b>4 PUS operations concepts</b> .....	<b>23</b>
4.1 Introduction .....	23
4.2 Application processes .....	24
4.3 Packet design and routing .....	24
4.4 Telecommanding .....	26
4.5 Telemetry reporting .....	30
4.6 Software management .....	31
4.7 On-board operations scheduling .....	32
4.8 On-board monitoring .....	33
4.9 On-board operations procedures .....	34
4.10 Attaching actions to on-board events .....	34
4.11 On-board storage and retrieval .....	34
4.12 Telemetry generation and forwarding .....	35
4.13 Memory management .....	35
4.14 Diagnostic mode .....	36
4.15 Off-line testing .....	37
<b>5 Service specification</b> .....	<b>39</b>
5.1 Introduction .....	39
5.2 Conventions .....	40

5.3	Telecommand packet structure .....	42
5.4	Telemetry source packet structure .....	46
5.5	Standard services .....	50
<b>6</b>	<b>Telecommand verification service .....</b>	<b>51</b>
6.1	Scope .....	51
6.2	Service concept .....	51
6.3	Service requests and reports .....	52
6.4	Capability sets .....	55
<b>7</b>	<b>Device command distribution service .....</b>	<b>57</b>
7.1	Scope .....	57
7.2	Service concept .....	57
7.3	Service requests and reports .....	58
7.4	Capability sets .....	59
<b>8</b>	<b>Housekeeping and diagnostic data reporting service .....</b>	<b>61</b>
8.1	Scope .....	61
8.2	Service concept .....	61
8.3	Service requests and reports .....	64
8.4	Capability sets .....	70
<b>9</b>	<b>Parameter statistics reporting service .....</b>	<b>73</b>
9.1	Scope .....	73
9.2	Service concept .....	73
9.3	Service requests and reports .....	74
9.4	Capability sets .....	77
<b>10</b>	<b>Event reporting service .....</b>	<b>79</b>
10.1	Scope .....	79
10.2	Service concept .....	79
10.3	Service requests and reports .....	80
10.4	Capability sets .....	81
<b>11</b>	<b>Memory management service .....</b>	<b>83</b>
11.1	Scope .....	83
11.2	Service concept .....	83
11.3	Service requests and reports .....	84
11.4	Capability sets .....	89
<b>12</b>	<b>Function management service .....</b>	<b>91</b>
12.1	Scope .....	91
12.2	Service concept .....	91
12.3	Service requests and reports .....	91
12.4	Capability sets .....	93
<b>13</b>	<b>Time management service .....</b>	<b>95</b>
13.1	Scope .....	95
13.2	Service concept .....	95
13.3	Service requests and reports .....	96
13.4	Capability sets .....	97





<b>14</b>	<b>On-board operations scheduling service</b>	<b>99</b>
14.1	Scope	99
14.2	Service concept	99
14.3	Service requests and reports	103
14.4	Capability sets	116
<b>15</b>	<b>On-board monitoring service</b>	<b>119</b>
15.1	Scope	119
15.2	Service concept	119
15.3	Service requests and reports	123
15.4	Capability sets	134
<b>16</b>	<b>Large data transfer service</b>	<b>135</b>
16.1	Scope	135
16.2	Service concept	136
16.3	Service requests and reports	139
16.4	Capability sets	143
<b>17</b>	<b>Packet forwarding control service</b>	<b>145</b>
17.1	Scope	145
17.2	Service concept	145
17.3	Service requests and reports	146
17.4	Capability sets	151
<b>18</b>	<b>On-board storage and retrieval service</b>	<b>153</b>
18.1	Scope	153
18.2	Service concept	153
18.3	Service requests and reports	156
18.4	Capability sets	164
<b>19</b>	<b>Test service</b>	<b>167</b>
19.1	Scope	167
19.2	Service concept	167
19.3	Service requests and reports	167
19.4	Capability sets	168
<b>20</b>	<b>On-board operations procedure service</b>	<b>169</b>
20.1	Scope	169
20.2	Service concept	169
20.3	Service requests and reports	170
20.4	Capability sets	175
<b>21</b>	<b>Event-action service</b>	<b>177</b>
21.1	Scope	177
21.2	Service concept	177
21.3	Service requests and reports	178
21.4	Capability sets	180
<b>22</b>	<b>Summary of service requests and reports</b>	<b>183</b>

<b>23</b>	<b>Parameter types and structure rules</b>	<b>189</b>
23.1	Introduction	189
23.2	Conventions	190
23.3	Encoding formats of parameter types	190
23.4	Tailoring of packet structures for a mission	191
23.5	Simple parameter types	192
23.6	Structured field types	199
<b>Annex A</b>	<b>(normative) Examples</b>	<b>207</b>
A.1	Specification of the cyclic redundancy code (CRC)	207
A.2	Specification of the ISO checksum	212
A.3	Use of service type 2, device command distribution	214
<b>Annex B</b>	<b>(normative) Mission constants</b>	<b>215</b>
<b>Annex C</b>	<b>(normative) Spacecraft time protocols</b>	<b>217</b>
C.1	Introduction	217
C.2	Presentation of spacecraft elapsed time	217
C.3	Standard spacecraft time source packet	219
C.4	Spacecraft time correlation procedures	219
<b>Annex D</b>	<b>(normative) Command pulse distribution unit</b>	<b>221</b>
D.1	General requirements	221
D.2	Specification	222
D.3	Design requirements	224
<b>Bibliography</b>		<b>225</b>

## Figures

Figure 1:	Example of a telecommand execution profile	27
Figure 2:	Telecommand system layers	29
Figure 3:	Telecommand packet fields	42
Figure 4:	Telemetry source packet fields	46
Figure 5:	The relation between execution result and interlock status	102
Figure 6:	Parameter check definitions and check status	121
Figure 7:	The splitting of a service data unit in parts to be downlinked (uplinked)	136
Figure 8:	An example of a storage and retrieval approach	155
Figure 9:	States and transitions for an on-board operations procedure	170
Figure 10:	Diagram conventions used by the structure rules set #1	200
Figure A-1:	Standard packet check sequence generation	208
Figure A-2:	Encoder	209
Figure A-3:	Decoder	209

## Tables

Table 1: Telecommand acceptance and execution telemetry .....	27
Table 2: Standard services specified within this Standard .....	50
Table 3: Summary of telecommand verification service minimum capabilities .....	55
Table 4: Summary of telecommand verification service additional capabilities .....	55
Table 5: Summary of device command distribution service minimum capabilities .....	59
Table 6: Summary of device command distribution service additional capabilities .....	60
Table 7: Summary of housekeeping sub-service minimum capabilities .....	70
Table 8: Summary of diagnostic sub-service minimum capabilities .....	70
Table 9: Summary of report definitions control additional capabilities .....	71
Table 10: Summary of report definitions reporting additional capabilities .....	71
Table 11: Summary of sampling time offset reporting additional capabilities .....	71
Table 12: Summary of filtered mode minimum capabilities .....	72
Table 13: Summary of filtered mode additional capabilities .....	72
Table 14: Summary of parameter statistics reporting service minimum capabilities .....	77
Table 15: Summary of parameter statistics reporting service additional capabilities .....	78
Table 16: Summary of event reporting service minimum capabilities .....	81
Table 17: Summary of event reporting service additional capabilities .....	81
Table 18: Summary of memory management service additional capabilities .....	90
Table 19: Summary of function management service minimum capabilities .....	93
Table 20: Summary of rate control sub-service minimum capabilities .....	97
Table 21: Summary of time reporting sub-service minimum capabilities .....	97
Table 22: Decision table for the release status of a telecommand .....	101
Table 23: Summary of on-board operations scheduling service minimum capabilities ..	116
Table 24: Summary of on-board operations scheduling service additional capabilities ..	116
Table 25: Summary of on-board monitoring service minimum capabilities .....	134
Table 26: Summary of on-board monitoring service additional capabilities .....	134
Table 27: Summary of sending sub-service (downlink) minimum capabilities .....	143
Table 28: Summary of receiving sub-service (uplink) minimum capabilities .....	143
Table 29: Summary of sending sub-service (downlink) additional capabilities .....	144
Table 30: Summary of receiving sub-service (uplink) additional capabilities .....	144
Table 31: Decision table for the forwarding status of a packet .....	146
Table 32: Summary of packet forwarding control service minimum capabilities .....	151
Table 33: Summary of packet forwarding control service additional capabilities .....	152
Table 34: Summary of packet selection sub-service minimum capabilities .....	164
Table 35: Summary of storage and retrieval sub-service minimum capabilities .....	164
Table 36: Summary of packet selection sub-service additional capabilities .....	164
Table 37: Summary of storage and retrieval sub-service additional capabilities .....	165
Table 38: Summary of test service minimum capabilities .....	168
Table 39: Summary of on-board operations procedure service minimum capabilities ..	175

---

Table 40: Summary of on-board operations procedure service additional capabilities . .	175
Table 41: Summary of event-action service minimum capabilities . . . . .	180
Table 42: Summary of event-action service additional capabilities . . . . .	181
Table 43: Summary of requests and reports for PUS standard services . . . . .	183

---

## Scope

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

This Standard is structured as follows. Firstly a set of operational concepts is identified, covering all the fundamental requirements for spacecraft monitoring and control during satellite integration, testing and flight operations. A set of services that satisfy these operational concepts is then defined. The detailed specification of these services includes the structure and contents of the associated service requests (telecommand packets) and service reports (telemetry source packets).

This Standard is applicable to any mission, no matter what its domain of application, orbit or ground station coverage characteristics. However, it is not the intention that the PUS is applicable to a given mission in its entirety. The operational concepts and corresponding services contained in this Standard cover a wide spectrum of operational scenarios and, for a given mission, only a subset of these operational concepts and services is 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 the particular mission. These choices include:

- a. 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).
- b. Which PUS services are supported by each application process.

NOTE Tailoring is a process by which individual requirements or specifications, standards and related documents are evaluated and made applicable to a specific project by selection, and in some exceptional cases, modification of existing or

addition of new requirements (ECSS-M-00-02A, clause 3 (Reference [11])).

Each mission should document the results of this design and selection process in a space-ground interface control document (SGICD).

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 functionality 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, which also remains sensible and coherent. At least this set is included in every implementation of a given service.

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.

---

## 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 revisions 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 most recent editions of the normative documents indicated below. For undated references the latest edition of the publication referred to applies.

ECSS-P-001	Glossary of terms
ECSS-E-50 <sup>1)</sup>	Space engineering – Communication
ECSS-E-70	Space engineering – Ground systems and operations
CCSDS 102.0-B-5	Packet Telemetry, Blue Book, Issue 5, November 2000
CCSDS 201.0-B-3	Telecommand Part 1 – Channel Service, Blue Book, Issue 3, June 2000
CCSDS 202.0-B-3	Telecommand Part 2 – Data Routing Service, Blue Book, Issue 3, June 2001
CCSDS 202.1-B-2	Telecommand Part 2.1 – Command Operation Procedures, Blue Book, Issue 2, June 2001
CCSDS 203.0-B-2	Telecommand Part 3 – Data Management Service, Blue Book, Issue 2, June 2001
CCSDS 301.0-B-2	Time Code Format, Blue Book, Issue 2, April 1990
CCSDS 727.0-R-5	CCSDS File Delivery Protocol (CFDP), Red Book, Issue 5, August 2001

---

1) To be published.

*(This page is intentionally left blank)*



---

## Terms, definitions and abbreviated terms

### 3.1 Terms and definitions

The following terms and definitions are specific to this Standard in the sense that they are complementary or additional to those contained in ECSS-P-001 and ECSS-E-70.

#### 3.1.1

##### **additional capability set**

set of service capabilities, service requests and reports that a service may optionally provide

#### 3.1.2

##### **application process**

entity, uniquely identified by an application process ID (APID), capable of generating telemetry source packets and receiving telecommand packets

NOTE 1 Application processes are the providers of the services defined within this Standard.

NOTE 2 An application process can be implemented in software, firmware or hardware.

NOTE 3 There are no restrictions on the mapping between application processes and the usual functional subdivision of a satellite into subsystems and payloads. In a simple satellite, there can be a centralized application process which provides a number of “dumb” platform subsystems with collection of housekeeping data, the distribution of device commands, on-board operations scheduling or on-board monitoring. In a more complex satellite, each subsystem and payload can be served by its own independent application process.

NOTE 4 A microprocessor can host one or more application processes. An application process (presumably a rather complex one) can be distributed across two or more microprocessors.

**3.1.3****control loop**

mechanism to maintain a parameter, or set of parameters, within defined limits

NOTE A control loop is comprised of a set of measurements and responses (commands) related according to a function, algorithm or set of rules. In the case of a satellite, if the characteristic dynamic behaviour of the parameter(s) being controlled is such that the measurement frequency is high and the control response time short, then these control loops are implemented on-board (e.g. attitude control).

**3.1.4****device command**

telecommand which is routed to and executed by on-board hardware, e.g. a relay-switching telecommand or a telecommand to load an on-board register

**3.1.5****memory**

on-board memory area, either main memory or mass memory

**3.1.6****minimum capability set**

set of service capabilities, service requests and reports that are supported by all implementations of a service

**3.1.7****parameter**

lowest level of elementary data item on-board

NOTE 1 A parameter has a unique interpretation.

NOTE 2 Traditionally, each distinct telemetry parameter is assigned a "parameter ID" for ground identification purposes. Within this Standard, the concept of "parameter number (#)" is introduced for the on-board "identification" of parameters. The parameter# is unique across a given spacecraft and there is a one-to-one correspondence between parameter ID and parameter#.

**3.1.8****parameter validity**

Boolean value (true or false) that determines whether a parameter value is meaningful

NOTE A parameter is valid if the appropriate well-defined conditions prevail under which it is meaningful to interpret its telemetered value.

EXAMPLE The angular output of a gyro can only have a valid engineering meaning if the power to the gyro is "on"; at other times the output can be random (or should not be relied upon). Such a parameter is deemed conditionally valid, with its validity determined from the power status.

**3.1.9****platform subsystem or payload**

combination of units within the satellite platform or the payload that fulfils a well-defined and self-contained set of on-board functions

**3.1.10****service**

set of on-board functions offered to a service user that can be controlled and monitored (e.g. by a ground system) through a well-defined set of service requests and reports

NOTE 1 A service can interact with other on-board functions for the execution of its activities.

NOTE 2 See minimum capability set.

NOTE 3 See additional capability set.

**3.1.11****service data unit**

packet data field part of a service request, response or indication, i.e. the complete packet excluding the packet header fields

**3.1.12****service provider**

on-board application process which executes the service activities and which is the destination of the service requests and the source of the service reports

NOTE 1 Distinct instances of one and the same service can be provided by several on-board application processes.

NOTE 2 A given application process can provide several services.

**3.1.13****service report**

data exchange between a service provider (the report initiator) and a service user to provide information either relating to the execution of an activity initiated by the user (e.g. notification of completion of execution) or relating to a meaningful event which occurred during the internal execution of a continuous service activity (in the latter case, the report is a provider-initiated report)

NOTE 1 The initiation of a service report by an on-board service provider results in the sending of one or more telemetry source packets to a service user.

NOTE 2 When the only purpose of a service request is to obtain some service data, the service report used to pass the data is called a "service response". In all other cases, the service report is called a "service indication". A service indication is used to report on the progress or completion of the execution of a user-initiated activity and on events occurring during the execution of a continuous service activity.

NOTE 3 Clause 22 contains a summary of requests, responses and indications for the standard services defined in this Standard.

**3.1.14****service request**

data exchange between a service user (the request initiator) and a service provider to initiate the execution of a particular activity

NOTE 1 The invocation of an activity by a service user results in the transmission of one or more telecommand packets to the service provider, the reception of which initiates the corresponding activity.

NOTE 2 If several users can interact with a service, then additional information is passed to the recipient service provider to identify the user invoking the activity. This information is

used by the service provider to route the related service report(s) back to the user who invoked the activity.

NOTE 3 All user-initiated activities are optionally confirmed.

### 3.1.15

#### **service user**

entity (e.g. a ground system) that initiates the service requests and receives the service reports

NOTE The specification of the activities performed by a service user (e.g. to process a service report) is beyond the scope of this Standard.

### 3.1.16

#### **software function**

software entity implementing a mission-specific on-board function that can be controlled by commands from the ground

### 3.1.17

#### **sub-schedule**

distinct subset of an on-board command schedule in the sense that it can be independently started and stopped

NOTE Commands within the same sub-schedule can be interlocked with each other.

### 3.1.18

#### **supporting service**

service used only in the context of another service, i.e. its service requests are invoked as a consequence of executing the user-initiated or continuous activities of the supported service

### 3.1.19

#### **telecommand**

data packet transported by a telecommunications system to give instruction or data to equipment located on-board a spacecraft

NOTE In line with common usage within the space domain, the terms command and telecommand are used interchangeably within this Standard.

### 3.1.20

#### **telecommand function**

operationally self-contained control action which can comprise, or invoke, one or more lower-level control actions

### 3.1.21

#### **vital telecommand**

telecommand which is not hazardous, but which, if not executed at the expected time, can cause a significant degradation to the mission

## 3.2 Abbreviated terms

The following abbreviated terms are defined and used within this Standard.

<b>Abbreviation</b>	<b>Meaning</b>
<b>Ack</b>	acknowledgement
<b>AD</b>	applicable document
<b>ADT</b>	aborted data transfer
<b>AOCS</b>	attitude and orbit control system
<b>AOS</b>	acquisition of signal

---

<b>APID</b>	application process ID
<b>ASCII</b>	American Standard Code for Information Interchange
<b>CCSDS</b>	Consultative Committee for Space Data Systems
<b>CDS</b>	CCSDS day segmented
<b>CLCW</b>	command link control word
<b>CLTU</b>	command link transfer unit
<b>CPDU</b>	command pulse distribution unit
<b>CRC</b>	cyclic redundancy code
<b>CUC</b>	CCSDS unsegmented code
<b>EGSE</b>	electrical ground support equipment
<b>ESA</b>	European Space Agency
<b>GPS</b>	global positioning system
<b>ISO</b>	International Organization for Standardization
<b>LEO</b>	low Earth orbit
<b>LOS</b>	loss of signal
<b>LSB</b>	least significant bit
<b>MAP</b>	multiplexed access point
<b>MSB</b>	most significant bit
<b>OBT</b>	on-board time
<b>PAC</b>	packet assembly controller
<b>PC</b>	parameter code
<b>PCS</b>	packet check sequence
<b>PEC</b>	packet error control
<b>PFC</b>	parameter format code
<b>PSS</b>	procedures, specifications, standards
<b>PTC</b>	parameter type code
<b>PUS</b>	packet utilization standard
<b>RAM</b>	random access memory
<b>RF</b>	radio frequency
<b>RID</b>	report identification
<b>SAU</b>	smallest addressable unit
<b>SDU</b>	service data unit
<b>SGICD</b>	space-ground interface control document
<b>SID</b>	structure identification
<b>SSC</b>	source sequence count
<b>UTC</b>	universal time coordinated
<b>VC</b>	virtual channel

*(This page is intentionally left blank)*

---

## PUS operations concepts

### 4.1 Introduction

This clause describes operations concepts that form the basis for the services specified in the remainder of this Standard. Not all of the operation concepts (nor all aspects of a given concept) are applicable to a given mission. Applicability of the concepts is decided on a case-by-case basis, dependent on the mission operational requirements.

The concepts are presented in the following sequence:

- a. Overall concepts and guidelines for application process and packet design.
- b. Concepts relating to routine operations scenarios, such as telecommanding and telecommand verification, telemetry reporting, on-board software management, on-board operations scheduling, on-board monitoring, on-board storage and retrieval and packet forwarding control.
- c. Concepts that relate to non-routine operations scenarios, such as software maintenance and troubleshooting. This includes memory management, diagnostic data reporting and off-line testing.

Some of these operations concepts are prescribed within higher-level CCSDS packet recommendations; others are developed from experience with past and present space missions.

Some operations concepts, or the corresponding services, introduce quantities for which values cannot be globally defined, but which shall be defined on a mission-by-mission basis. Examples of such quantities are time intervals and sizes of parameter sets. These are termed “mission constants” and are identified within this Standard in right-angled parentheses and typical values are indicated. These mission constants are also summarized in annex B.

EXAMPLE <PKTS\_NUM\_STORED>.

## 4.2 Application processes

### 4.2.1 Background

Within the CCSDS packet telemetry and telecommand concept (CCSDS 102.0-B-5 to CCSDS 203.0-B-2), an on-board application process is defined as the source of telemetry source packets and the sink for telecommand packets. A given application process has a unique identifier, namely its application process ID (APID). These reference documents place no particular restrictions on the design of application processes or on the assignment of APIDs for a given mission, except that APID= 2047 (all “ones”) is reserved by CCSDS 102.0-B-5 for the idle packet.

This Standard imposes two additional rules:

- a. APID=0 shall be reserved for the standard spacecraft time packet;
- b. each command pulse distribution unit (CPDU) shall have a dedicated APID.

### 4.2.2 The assignment of APIDs for a mission

The process of allocation of APIDs for a given mission shall consider factors such as:

- a. the number of on-board data sources or sinks (e.g. subsystems, payloads);
- b. the virtual channelization scheme for the mission;
- c. the requirements for separation of data from a given source or to a given sink.

Different APIDs shall be assigned to different on-board data sources or sinks and also within the same source or sink where there are packet streams with different bandwidth characteristics and operational significance. This may apply to:

- real-time and deferred data;
- science packets and housekeeping packets from an instrument.

An on-board entity that is both a data source and sink should use the same APID for its telemetry source packets and telecommand packets. Once assigned by the mission control authority, the allocation of APIDs should not be changed for a mission.

## 4.3 Packet design and routing

### 4.3.1 Identification of both packet source and destination

For missions with several sources for a given telecommand packet or sinks for a given telemetry source packet (e.g. where CCSDS/PUS packets are used for on-board communication between application processes or where there are multiple ground commanding sources), both the source and the destination of the packet shall be explicitly indicated. For those missions, the “end-to-end” identification of the packet is used for the selective routing, filtering or other processing of packets, either on ground or on-board.

One solution is to assign APIDs to suit this purpose, i.e. to define APIDs that identify both the source and the destination of a packet. However, the limited size of the APID field (11 bits) seriously limits the number of possible source/destination combinations. The source (or destination) ID of the telecommand (or telemetry source) packet should be included in the CCSDS secondary header (referred to in this Standard as the data field header). This approach is also in line with the guidelines contained in the corresponding CCSDS concept documents (see CCSDS 100.0-G-1, CCSDS 200.0-G-6). If a mission uses both a source and destination ID, the packet source sequence count shall be separately maintained for each source-destination couplet, to avoid ambiguity.



### 4.3.2 Missing telemetry source packets

The source sequence count (SSC) for telemetry source packets shall be incremented separately for each on-board application process. Where a given APID generates a number of different types of telemetry source packet, and there is a discontinuity (for whatever reason) in the SSC, there is potential ambiguity concerning which packets are missing. Where it is important for a mission to identify lost telemetry source packets at a level of granularity lower than the APID, this Standard makes provision for including a secondary counter (termed a packet sub-counter) within the data field header.

### 4.3.3 Packet size

Taking into account factors such as uplink bandwidth, the duration of ground station passes and the time to recover from telecommand failure, consideration should be given to the definition of a maximum telecommand packet length (<TCPKT\_MAX\_LENGTH>) that is less than the maximum of 65K octets defined by CCSDS 203.0-B-2. For analogous reasons, a maximum telemetry source packet length may be defined (<TMPKT\_MAX\_LENGTH>) that is less than the maximum defined by CCSDS 102.0-B-5.

There may be circumstances where a set of data is transmitted (uplink or downlink) that exceeds the maximum packet size, for example, a large memory load or dump or a report of the contents of an on-board command schedule. This data can be sent as a group (TM) or sequence (TC) of packets, using the grouping (sequence) flags provided in the packet primary header. However, this mechanism does not provide security in the event of missing packets. This Standard therefore defines a capability to transfer large service data units in a more secure manner, on either the uplink or the downlink. This large data transfer capability utilizes packets of the maximum defined size (the last packet can be smaller, of course) with the possibility for acknowledgement of receipt (either intermediate or final). If packets are rejected or missing their re-transmission shall be initiated in order to complete the data transfer.

### 4.3.4 Packet routing

#### 4.3.4.1 Telemetry source packets

The primary mechanism for telemetry bandwidth control is that of virtual channelization. This mechanism also provides the means to separate on ground data streams for distribution to different destinations. The allocation of data classes to virtual channels should not be changed dynamically during the course of a mission, however some missions may provide an on-board mechanism for the mapping of packets to VCs. In this case a report of the on-board mapping information shall be available.

Selection of telemetry source packets can be performed on the ground for priority routing, on the basis of operational significance, for example, where there is a bandwidth-limited communications link between a ground station and the control centre and forwarding of all the telemetry as it is received from the spacecraft in real time is not feasible. In addition to performing selection on the basis of VC, packets can be selected from within a given VC. This implies that the operational significance of a packet shall be indicated explicitly within the primary header or the data field header of the packet. This Standard defines fields for this purpose within the data field header (see clause 5).

#### 4.3.4.2 Telecommand packets

Commands of different operational priority should be assigned to different multiplexed access points (MAP), with the higher priority commands being routed to the priority MAP. With this mechanism, the transmission of a long telecommand packet can be “interrupted” by the transmission of another telecommand packet of higher priority that is issued later in time. This can apply when the packets are

destined for different application processes and when they are destined for the same application process.

## 4.4 Telecommanding

### 4.4.1 The different levels of control

It is expected that the routine operations of a spacecraft are effected at a high-level, utilizing the specialized services and functions of an application process, as defined in this Standard, or by mission-specific services. However, the spacecraft design will include a number of elementary (indivisible) control actions, implemented either in hardware or software. Examples of such control actions include: switching a relay, loading data into a hardware register or starting or stopping a software task.

High-level commands to the spacecraft (service requests, application function commands) can result in the eventual triggering of one or more such elementary control actions, for example, a request to power up an experiment in a given mode can generate a particular sequence of “relay on” commands and software configuration commands. In some cases, there can be several hierarchical layers between the highest and lowest level of control.

Although not used during routine operations, the ground system shall have direct access to all on-board elementary control actions. This ensures that the ground system has a complete set of capabilities when performing contingency operations or troubleshooting, for example, to by-pass an on-board function that can be malfunctioning.

A telecommand function is an operationally self-contained control action. A telecommand function can comprise, or invoke, one or more lower-level control actions. The design of each telecommand packet should ensure that it contains only one telecommand function.

### 4.4.2 Device commands

One particular group of elementary control actions are routed to, and executed by, on-board hardware. These are termed device commands and comprise hard-wired commands as well as On-Off and register load<sup>2)</sup> commands.

A number of different mechanisms exist for routing these device commands:

- a. They can be uplinked at telecommand segment level, by embedding one device command within a single telecommand segment. When the telecommand segment is received on-board, the device command is extracted and routed directly to the device. This mechanism can be used, for example, to by-pass a MAP or a packet assembly controller (PAC) or when an application process used to distribute device commands is not available (e.g. in the event of a software or processor anomaly).
- b. Most spacecraft include one or more command pulse distribution units (CPDUs), which are devices located within the decoder(s) that issue command pulses of specified duration. Each CPDU is addressed by means of a dedicated application process and is used for the distribution of vital (or “high-priority”) spacecraft commands with the minimum of on-board software intervention.
- c. Device commands can be uplinked within a telecommand packet to an application process that shall then extract the command and route it to the device.

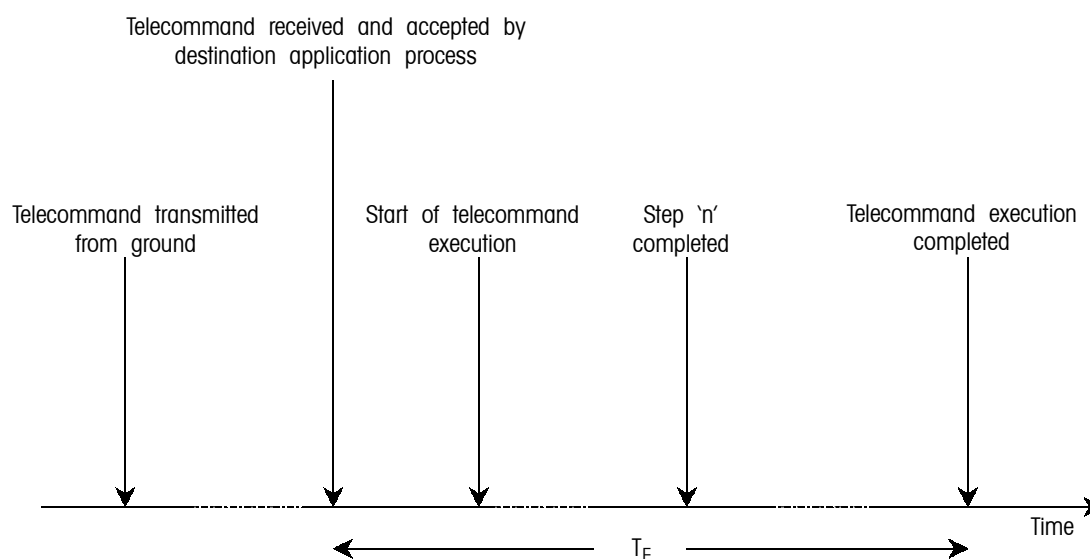
Certain device commands can be grouped or sequenced in such a way that they constitute a higher-level command function. In order to ensure that either the complete function is executed, or nothing at all, the capability should exist to

2) In the days of PCM telecommanding, these were called memory load commands. This term is now reserved for the loading of on-board (RAM) memories.

“pack” a number of device commands of the same type (On-Off or register load) within a single telecommand packet.

### 4.4.3 Telecommand verification

The basic concept for telecommand verification is that the on-board system should provide feedback on the execution of a given telecommand at whichever stages are meaningful for that telecommand. An example of a telecommand execution profile is shown in Figure 1. The delay shown between transmission of the telecommand from the ground and acceptance by the “destination” application process can correspond to propagation delay (deep-space missions) or the on-board storage of the telecommand prior to release to the application process.



**Figure 1: Example of a telecommand execution profile**

For an immediate telecommand, a separate notification of telecommand acceptance need not be given. For a short execution duration telecommand, only one execution verification report packet (combining start and completion status) can be sufficient. Requirements for the different stages of telecommand verification are summarized in Table 1 for telecommands with different execution characteristics:

**Table 1: Telecommand acceptance and execution telemetry**

	Immediate command	Delayed execution command
Short execution duration $T_E < 10 \text{ s}$	$A + E_t$ or $E_t$	$A + E_t$
Long execution duration $T_E > 10 \text{ s}$	$A + E_s + E_p + E_t$ or $E_s + E_p + E_t$	$A + E_s + E_p + E_t$
Where: A = Notification of telecommand acceptance E = Execution report: $E_s$ = Execution start $E_p$ = Execution progress (one or more reports) $E_t$ = Execution complete		

Failure of telecommand execution at each verifiable stage should always be reported. However, the selection of which reports of execution success are generated for each individual instance of the telecommand depends on the prevailing operational scenario (e.g. whether or not there is ground coverage). For this purpose, this Standard has defined an “Ack” field within the data field header of a telecommand packet.

The existence of the capability for different stages of telecommand verification does not imply that every telecommand shall be verifiable at all stages. Those stages that can be verified shall be specified for each telecommand and the ground system shall be configured to ensure that only verification of those stages can be requested.

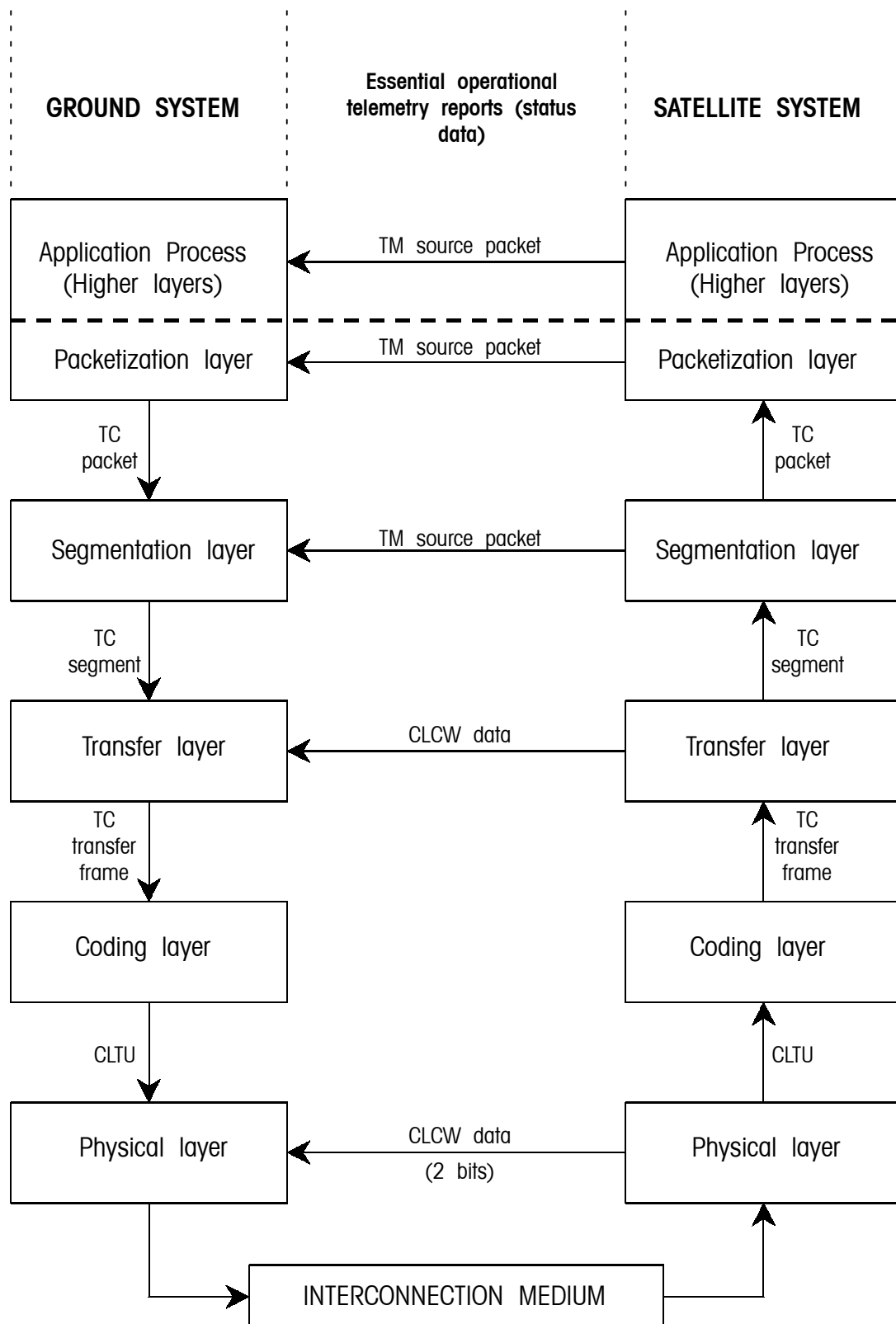
Figure 2 shows the different layers of the telecommand system; the layers below the dotted line are subject to the CCSDS recommendations (CCSDS 201.0-B-3 to CCSDS 203.0-B-2). This Standard addresses the application process layer, above the dotted line.

The packets notifying telecommand acceptance and telecommand execution verification and any other telemetry source packets used for end-to-end verification of telecommand execution represent progressively higher level protocols within the application process layer.

Telecommand verification reports shall unambiguously identify the command being verified and, in the case of failure, shall contain ancillary data to assist in the identification of the nature and reason for the failure.

This Standard defines a number of standard failure codes for the acceptance stage of telecommand verification. Failure codes for later stages are defined on a mission-specific basis.

The telecommand verification concept defined here refers to telecommands originating from ground. This includes commands pre-loaded into an on-board schedule for release to an application process at a later time (see subclause 4.7). However, the concept does not apply to commands generated autonomously on-board; the notification to the ground system of the invocation (and any subsequent verification) of these commands shall be done using event reporting (see subclause 4.5.2).



**Figure 2: Telecommand system layers**

## 4.5 Telemetry reporting

### 4.5.1 Housekeeping data reporting

Every spacecraft has a number of predefined housekeeping packets, whose content, layout and generation frequency is specified by the spacecraft manufacturer. The criteria used in designing these packets shall ensure that:

- a. the spacecraft status is reported in a complete and unambiguous manner;
- b. the frequency of packet generation is compatible with any requirements on response time from control loops that are closed via the ground.

An appropriate reserved downlink bandwidth shall be provided for these packets (even if it is not always used).

To adapt to changing operational requirements, the capability shall be included to modify both the data content and generation frequency of housekeeping packets during the mission. Different data reporting schemes can be used due to factors such as:

- what was considered at the design stage, in terms of data content and packet frequency, can be less than optimal after some period in-orbit;
- sensor characteristics and equipment performance can evolve with time;
- there can be on-board failures and anomalies.

The normal mechanism for reporting of housekeeping data is to downlink housekeeping telemetry packets on a periodic basis. However, since the ground system is essentially only interested in being informed of any changes in status or performance, an additional mode of reporting is defined in which housekeeping packets are only generated when parameters within them have changed in value. In this filtered reporting mode, certain parameters can be filtered out from this change comparison process (e.g. “noisy” analogue parameters that continuously change value). In addition, the threshold for the change in value that triggers the generation of the packet can also be specified by ground. Since this filtered mode can result in long passages of time where no packet is generated, the concept shall support the production of a packet after a specified timeout period (the packet is generated periodically, but with a much longer period).

The capabilities to define new housekeeping packets and for generating filtered reports are optional. The minimum implementation of the housekeeping function consists of periodic reporting of predefined packets.

### 4.5.2 Event reporting

In addition to periodic reporting of predefined parameter sets, the spacecraft shall report whenever an event of operational significance is detected on-board, e.g. predefined milestones in the normal progress of activities initiated either from ground or autonomously on-board from the detection of anomalies by a failure detection and recovery function.

Because such event reports are infrequent (and hence consume low bandwidth) and are often unexpected, they should be designed to convey information about the event clearly and unambiguously. This can be achieved in two ways:

- a. flagging the severity of the report in the data field header, in order to facilitate ground system detection and routing;
- b. including an appropriate set of auxiliary data within the report itself so that the ground system can understand the context and cause of the on-board event.

### 4.5.3 Statistical data reporting

Another mechanism for reducing the quantity of engineering data reported to the ground system is to evaluate statistics (mean, min, max, standard deviation) on-board for a parameter set and to report only the results of this evaluation. This mechanism is appropriate for missions with limited ground coverage (e.g. low-earth orbiters), where the report can be used to provide a summary of the behaviour of parameters during the previous period of “no contact”.

In the minimum implementation of this function, the set of parameters and the attributes being evaluated shall be predefined and the statistics shall be evaluated continuously. However, optional extensions include the capability for the ground system to modify the set of parameters or their attributes and the intervals of time over which the statistics are evaluated.

### 4.5.4 Time information

Annex C defines a mechanism for the spacecraft to report its time reference to the ground system, via the standard spacecraft time packet, and procedures to be used on ground to perform a correlation between on-board time (OBT) and ground time. The frequency of generation of the standard spacecraft time packet can be fixed for a mission, or it may be modified in-orbit (e.g. if the accuracy is a function of mission phase or experiment operation mode).

The knowledge of this correlation enables the ground system to reconstitute accurately the on-board time of other information reported in the telemetry, such as the time of occurrence of an event or the on-board sampling time of a parameter. The timing information may be either:

- a. explicitly reported, i.e. the event or the parameter sampling is time-stamped; or
- b. deduced from implicit knowledge of time offsets or on-board sampling sequences – in this case, the on-board time reference shall be reported within the data field header of the packet.

It is assumed that any on-board synchronization with a master clock is done autonomously; however, the possible mechanisms for on-board time distribution or synchronisation are beyond the scope of this Standard.

Missions using the global positioning system (GPS) for their on-board time reference have different requirements for the exchange of time information between the satellite and the ground segment. This concept is not covered by this Standard.

## 4.6 Software management

This Standard identifies a number of standard services that can be provided by one or more application processes. These services are standardized because they relate to on-board activities and capabilities that are generic in nature, both across many missions, as well as across different application processes for the same mission.

In addition, a given application process can support one or more software functions that are invoked from the ground. These functions relate to the specialized role assigned to the application process on-board the spacecraft, for example, responsibility for controlling the operation of a payload instrument or an AOCS subsystem. The nature of this control can vary quite considerably and is not prescribed or constrained in any way by this Standard. In principle, it covers all on-board nominal operations for subsystems and payloads including predefined mode change operations.

These functions can be implemented as mission-specific services, with their own request (TC) and report (TM) data structures and service models. However, they can also be implemented using a generic, high-level function service. In this



concept, the function itself is specialized (e.g. configure experiment mode, set AOCS control mode), but the associated packet structure is standardized.

At whichever level the ground system interacts with on-board software (service-level, function-level), all communications between the ground system and on-board software shall be effected by means of telecommand and telemetry source packets specifically designed for the purpose. This communication should not be achieved by using general-purpose memory write (memory load) and memory read (memory dump) packets.

## 4.7 On-board operations scheduling

There are two scenarios where the capability for the on-board execution of operations that have been loaded in advance from the ground shall be implemented:

- a. Those missions that perform operations outside of ground contact because of limited ground station visibility (e.g. LEO spacecraft) or signal propagation delays (e.g. deep-space probes).
- b. Those missions whose operations concept is to minimize the dependency on the ground segment. Thus, a geostationary telecommunications or meteorological mission can perform all of its routine operations in this manner, even though the spacecraft is permanently in view of a ground station. This approach potentially increases the availability of operational services or mission products, since the continuous availability of the uplink is eliminated.

The simplest form of on-board operations scheduling consists of storing time-tagged commands that have been loaded from ground and releasing them to their destination application process(es) when their on-board time is reached. This is a purely open-loop function with no feedback being generated by the destination application process. This Standard includes the following (optional) scheduling functions:

- a. The command schedule may be partitioned into sub-schedules. These sub-schedules can be independently controlled (started, stopped) and can be used to group commands for different destinations (e.g. instruments) or commands constituting a related “sequence”. There is no size limit on a sub-schedule (except the size limit of the schedule as a whole) since it is only a logical partition of the command schedule.
- b. Commands can be interlocked, i.e. the release of a given command can be preconditioned on the successful (or unsuccessful) execution of another command that was released at an earlier time from the schedule. A given command can only be interlocked with a single earlier command. This implies a feedback from the destination application process (a sort of on-board telecommand verification loop). This capability provides simple preconditioning of command release (e.g. for safety). However, it also provides for alternative sequences of commands to be sent from the schedule depending on the success or failure of commands released at an earlier time (IF, THEN...ELSE capability). Interlocking is implemented at sub-schedule level, i.e. commands which are interlocked reside in the same sub-schedule.
- c. Because operations are often rescheduled in real life operations, commands can be time-shifted “en-bloc”, by supplying a delta-time from the ground system. This avoids the complicated procedure (often used in the past for time-tagged queues) of having first to delete them from the command schedule and then to reload them with new execution times.
- d. A relative (rather than absolute) execution time can be attached to a command with an indication of an “event” to which this relative time relates. These events can be mission-specific in nature; however a number of standard events are defined, such as the starting of the (overall) command schedule or the sub-schedule in which the command resides or the execution time (as



opposed to the release time) of the command with which this command is interlocked.

## 4.8 On-board monitoring

Some missions use an on-board parameter monitoring capability that is functionally similar to the traditional monitoring performed on the ground. The primary purpose of this function is to reduce downlinking all the low-level housekeeping data to the ground system, which in turn implies that the ground system is continuously available to monitor this data.

The on-board monitoring function can be provided by one or more application processes and the maximum number of parameters which can be simultaneously monitored per application process is `<MONLIST_MAX_CHECKS>`. The following types of check can be applied (depending on the parameter and its type):

- limit-check, i.e. a check that the parameter value lies within a specified range;
- expected-value-check, i.e. a check against a specified value;
- delta-check, i.e. a check of the change in value of a parameter against a pair of thresholds.

A parameter can be conditionally valid, in which case its validity shall be determined from the value (“TRUE” or “FALSE”) of an associated Boolean “validity parameter”. A conditionally valid parameter shall only be checked if it is valid. Several pairs of limits or expected values or delta checks can be specified per parameter, to a maximum of `<MONLIST_MAX_CHECKS>`, each with an associated “check selection parameter”. The parameter shall be independently checked against each check where the check selection parameter is currently “TRUE”.

Parameters and their monitoring attributes can be added or deleted from the monitoring list and control over the monitoring process can be effected both globally (for the application process) and at the level of individual parameters. Thus, the monitoring of a subset of parameters can be temporarily disabled at the same time the remainder of the parameters in the list are enabled.

Check transition reports are generated to notify the ground system of all check transitions. A check transition occurs whenever:

- a parameter fails one of its enabled checks;
- a parameter passes a check having previously been in a failed state (e.g. it is back within limits);
- a checked parameter becomes valid;
- a check becomes enabled.

Optionally, an event report can be generated if a monitoring violation occurs, for example, if a parameter value exceeds the high limit specified for it. This event report can then be used to trigger the initiation of an on-board recovery action (see subclause 4.10).

In the simplest on-board implementation of this function, the set of parameters to be monitored is predefined and the monitoring function can merely be enabled or disabled. A given parameter shall be checked only against a single check and no parameter validity checking is performed. Successive extensions to the function shall provide:

- the concept of multiple checks;
- the concept of parameter validity;
- the possibility to add or delete or modify parameters and their monitoring attributes in the monitoring list;
- reporting of the current monitoring list and out-of-limits list;
- the generation of an event report to trigger an on-board recovery action.

## 4.9 On-board operations procedures

Many operational activities shall be performed on-board using standard services, mission-specific services or other software functions. However, a mission may also have a set of on-board operations procedures that can be loaded from the ground system, which then has control over their subsequent execution. These procedures are similar in concept to the standard procedures that are executed in a stepwise manner on the ground. In principle, these procedures can also be invoked autonomously on-board, e.g. as the result of the on-board detection of a specified event.

The operations language used for the specification of these procedures is beyond the scope of this Standard. However, a procedure shall have a structure comprised of logical sub-elements or steps. In addition to starting and stopping a procedure, it shall be controllable at a lower level by suspending and resuming execution at the level of its component steps.

In the same manner as for operations procedures that are executed in the ground system, run-time parameters can be provided to an on-board operations procedure, either at the time that it is started or during execution.

## 4.10 Attaching actions to on-board events

As an extension to the on-board capability for detecting events and reporting them asynchronously to the ground system (see subclause 4.5.2), an action can be defined that is executed autonomously on-board whenever a given event is detected.

The class of events that can result in such an action are those events that also give rise to an event report. The associated action may be a telecommand of any type, i.e. a standard telecommand packet (as defined in this PUS) or any mission-specific telecommand.

A telecommand can initiate one of several possible types of action, including:

- directly reconfiguring on-board hardware;
- starting an on-board operations procedure (see subclause 4.9);
- starting a command sub-schedule (see subclause 4.7), which can contain a predefined command sequence.

## 4.11 On-board storage and retrieval

A capability to store packets on-board (e.g. in a mass memory) and to dump them, on request, to the ground should be considered under the following circumstances:

- a. Ground station coverage is intermittent. In this case, the on-board storage capacity shall be sized to store all packets generated on-board for spacecraft monitoring and control purposes, for a duration at least equal to the longest non-coverage period plus a mission-dependent margin `<PKT_STORAGE_TIME>`. These packets shall then be retrieved during subsequent ground station passes according to a selection strategy based upon the operational significance of the stored packets.

EXAMPLE Access to anomaly report packets is needed on the ground with shorter delays than routine status reporting packets.

For these types of mission, there can also be user-generated requirements for the on-board storage and retrieval of payload data.

- b. To recover lost packets. For missions with continuous ground coverage, the loss of one-shot packets (i.e. event-driven or request-driven packets) can be remedied by the short-term on-board storage of the last `<PKTS_NUM_STORED>`, typically the last 100. Telecommand verification packets are request-driven in this context. The ground system can detect that

a packet has been lost by checking the source sequence count or the packet sub-counter (see earlier) for a given application process.

The storage function (service) can be provided by a different application process from that which generates the packets.

EXAMPLE There may be a centralized storage and retrieval function that serves all on-board application processes.

For this reason, the function is split into two distinct parts: one part that performs the selection of packets to be stored and a second part that performs the control of the storage and retrieval functions. The latter part shall be located with the application process(es) that manages the packet store(s).

Depending on the on-board design, individual packet stores can be either circular, where the oldest packets are overwritten when the packet store is full (this is the only option for the lost packet recovery store), or bounded, where storage terminates when the packet store is full.

The storage of packets shall not be interrupted if the ground system requests a retrieval or deletion from an on-board packet store.

Retrieval from a packet store over a specified (storage) time window, or for a specified range of packets, should be provided.

In the simplest implementation of the storage and retrieval function, there is no selection possibility for the packets to be stored, but the storage can be enabled and disabled. The extended capabilities of this service include:

- a. adding and deleting packets to or from storage selection,
- b. reporting of packet storage selection,
- c. time-stamping of stored packets,
- d. deletion of packets from a packet store.

## 4.12 Telemetry generation and forwarding

The generation, on-board routing and transmission to the ground of telemetry source packets utilizes various resources, such as on-board processing capacity, on-board communications networks and downlink bandwidth, which are not unlimited. The generation and forwarding of telemetry source packets shall therefore be designed such that these resources are only used when actually needed, i.e. appropriate controllability of the generation and forwarding functions shall be provided.

Where applicable, generation control shall be effected at the level of the individual originating service, i.e. it is not a centralised function.

However, forwarding control shall be effected at a higher level, either:

- a. at the level of the originating application process (in this design option, a packet may be generated by a given service but not forwarded by the application process); or
- b. by a centralised application process which receives all telemetry source packets to be transmitted to the ground system and which controls the routing on the downlink.

## 4.13 Memory management

Direct management by the ground system of on-board memory areas is not a routine operational activity, but an “off-line” activity in support of troubleshooting operations or for the purposes of undertaking maintenance of on-board software. For such purposes, read access should be available to any on-board memory and the ground system should have the means to load any on-board RAM area.

To provide flexibility when loading (or dumping) memory, the capability should exist to load (or dump) either a contiguous area of memory or to load (or dump)

several dispersed areas at the same time; also known as a “scatter-load” (or “scatter-dump”). Optionally, on-board protection shall be provided to ensure that corrupted data cannot be loaded into memory. This shall be achieved by calculating a checksum over the data area(s) to be loaded and comparing this with a checksum that is provided by the ground system in the load packet.

Where implemented, the on-board end-to-end verification of a memory load shall consist of confirming that the data have been correctly loaded into their destination memory, by reading them back from the memory and comparing them with the original load data.

To check the content of an on-board memory (e.g. because a corruption is suspected), the capability shall be provided to request the calculation of a checksum on-board (across a specified memory area) and to send the result to the ground via telemetry. Comparing the checksum to a known reference value avoids the need to dump the suspected area to ground to check its content against a ground-based memory image.

Within the CCSDS recommendations there is a file delivery protocol (the CFDP, see CCSDS 727.0-R-5) designed specifically for use with on-board memories that are organized as “Filestores”. The CFDP provides the capability for the delivery of files from ground-to-space and vice versa, as well as capabilities for the management of files and directories within the filestore. Missions that use file management may also define their own mission-specific additions to the services defined within this Standard.

NOTE File delivery concerns the transport of data files between space and ground, regardless of their content, whilst this Standard defines the syntax and semantics of the data to be transported, regardless of the transport mechanism. As such, the two aspects are complementary.

## 4.14 Diagnostic mode

For the purposes of anomaly investigation (troubleshooting), a telemetry-sampling setup quite different from that used in normal operations can be defined. A selected set of parameters pertaining to the problem shall be sampled on-board at a higher rate than normal and then telemetered to the ground system. This scenario is termed “Diagnostic mode” and is synonymous with the terms “Dwell mode” and “Oversampling mode”, which have been used in the past. When operating in diagnostic mode, the ground system shall carefully select the extent to which these diagnostic packets shall occupy the downlink bandwidth taking into account the other prevailing operational requirements.

Functionally, diagnostic packets shall have the same characteristics as house-keeping packets, for example, new packets can be defined and their generation can be controlled. However, their usage is quite distinct and they shall therefore be identified separately for routing (and possible storage) and processing purposes. Hence, they have different “operational significance” information within the packet data field header.

All on-board telemetry parameters shall be accessible for sampling and reporting via diagnostic packets. In order to ensure that high-rate sampling can be achieved a minimum sampling interval shall be definable for all on-board parameters (<DIAG\_MIN\_INTERV>, typically 10 ms to 30 ms). In practice, the sampling interval selected for different subsets of parameters can vary quite considerably. The minimum sampling interval can be used for parameters which can vary rapidly (e.g. AOCS sensor data), whilst for slowly varying parameters (e.g. temperatures), a longer sampling interval (e.g. one second or more) can be used.

## 4.15 Off-line testing

For troubleshooting or pre-operational validation purposes, an on-board unit or process can be operated in an off-line manner, i.e. not as a part of any active control loop, but nevertheless providing all its normal outputs to the telemetry for analysis on the ground. For these purposes, each application process shall provide a set of specialized (i.e. mission-specific) end-to-end test functions that can be exercised under ground control.

As a minimum, a standard “are you alive” function shall be provided for testing the end-to-end connection between the ground system and an application process.

*(This page is intentionally left blank)*

---

## Service specification

### 5.1 Introduction

Clause 4 identified a number of operations concepts that imply distinct sets of capabilities to be implemented on-board the satellite along with corresponding monitoring and control facilities on the ground.

These sets of capabilities constitute “services”, the monitoring and control of which shall be achieved via a well-defined set of interactions between the provider of the service (i.e. an on-board application process) and the user(s) of the service (e.g. a ground system).

This Standard contains the specification of those standard services which can be provided by on-board application processes in fulfilment of the operations concepts for satellite monitoring and control.

A service specification defines:

- a. what can be requested of the service;
- b. what can be issued by the service to notify the user of the success or failure of its requests and to report events detected during the execution of the service activities;
- c. what internal activities the service shall perform (to process a request or to detect or process events relevant to the service).

The specification of the information to be maintained by a service and of the activities to be performed by a service are expressed as functional requirements. How these functional requirements are implemented is not enforced by this Standard (however, the actual implementation of the service state information and activities shall be functionally equivalent).

The following criteria are used for identifying and defining these standard services:

- **Commonality**

A service should correspond to a group of functionalities applicable to many missions.

- **Coherence**

The capabilities to be provided by a service should be closely related and their scope should be unambiguously specified. A service should cover all the activities for managing inter-related state information and all activities which use that state information. A service should have minimum and well-defined interactions with other services or software functions.

- **Implementation independence**

A service should neither assume nor exclude a particular satellite on-board architecture (hardware or software).

This clause contains:

- Conventions which are used within the later service specifications.
- Items which are common between services. This includes the overall structure of telecommand and telemetry source packets, which is defined elsewhere (CCSDS 102.0-B-5, CCSDS 203.0-B-2) and those standard elements of packets which are defined within this Standard.
- A summary of the standard services defined within this Standard.

## 5.2 Conventions

In the following clauses, the specification of a service consists of:

- a. An introduction containing the rationale for, and the scope of, the service.
- b. The service concept describing the continuous activities and the state information to be maintained by the service, where these are not explicitly identified in the corresponding operations concept.

The activity and information concepts correspond to a service implementation which provides the full set of capabilities. However, a given implementation may use an alternative but compatible activity and information design, and only implement those aspects applicable to the minimum capabilities set and the subset of additional capabilities which it supports.

Some services may be constituted of distinct parts which may be implemented in isolation from other parts (and even, in some cases, may be implemented within a different application process). The concept of sub-service is introduced for these distinct service parts.

- c. The definition of the structure and content of the service requests and reports of the user initiated and continuous activities. Each distinct service request and report is uniquely identified by a couplet of numbers. The first pertains to the service to which it belongs (the Service Type) and the second is a unique number within the service (the Service Subtype).

An activity specification contains the following information:

1. For a user-initiated activity, the parameters of the service request and the structure of the corresponding service data unit to be placed in the telecommand packet used to transport the request.
2. The actions to be executed by the service provider to perform the activity. This includes an indication of error or failure cases wherever identified during the specification of an activity. However, this Standard does not identify all possible such error or failure cases. Nor does it preclude the identification of implementation-specific errors when a service is designed and implemented for a particular mission.

Many errors relate to checks performed on data contained in service requests. These errors should be reported as telecommand verification failures where the appropriate level of telecommand verification is supported. Otherwise, they should be reported by means of an event report.



3. For each report (response or indication) that can be issued during the execution of the activity, the parameters of the service report and the structure of the corresponding service data unit to be placed in the telemetry source packet(s) used to send the report.
4. The definition of the minimum and additional capability sets of the service, if any.

The service data units contained within a service request or report are preceded by their Name and their Service Type (x) and Service Subtype (y) in the format (x,y). The structure of the service data unit is presented in a graphical tabular format:

Parameter1	Parameter2	Parameter3
Unsigned Integer	Boolean	Enumerated

The first row of the table gives the name of the parameters of the service data unit. The second row indicates the type of the parameters. The parameter types are defined in clause 23.

Where the presence of a field, or group of fields is optional, this is indicated by the text “Optional” below the corresponding fields in the packet structure diagram. A field, or group of fields is optional if either:

- its presence is determined at the level of the mission, application process or service instance; or
- its presence is determined by the value(s) of one or more preceding fields in the service data unit.

The level at which a given field is optional is explained within the corresponding definition for that field.

The omission of an optional field may imply that the value to be used is known by both the service provider and the service user, for example, the service provider may use a fixed value or a “current value” which can be set by the service user through other means. The service provider may even use the value(s) of other preceding field(s) in the service request or report to access a fixed or modifiable look-up table in which the values are contained.

Where a field, or group of fields, constitutes an entry in a fixed-length or a variable-length array, this is indicated below the table by: “← Repeated N times →”, where N is the number of repetitions within the array. In the case of a variable-length array, N is given explicitly at the start of the array; in the case of a fixed-length array, N is known implicitly for the mission. Where the term “array” is used within this Standard, this should not be interpreted to imply any particular implementation connotation. Instead, one of the array structure types formally defined in clause 23 is intended.

Following each service data unit, a description is given of each of the contained fields. However, to avoid repetition, this description is not repeated if the same field appears (with the same interpretation) in subsequent service data units.

Within the specification of a service data unit, the following abbreviations are used:

Boolean parameter	“Boolean”
Enumerated parameter	“Enumerated”
Integer parameter unsigned integer signed integer	“Unsigned Integer” “Signed Integer”
Real parameter	“Real”
Bit-string parameter fixed length bit string variable length bit string	“Fixed BitString” “Variable BitString”
Octet-string parameter fixed length octet string variable length octet string	“Fixed OctetString” “Variable OctetString”
Character string parameter fixed length character string variable length character string	“Fixed CharString” “Variable CharString”
Time parameter absolute time parameter relative time parameter	“Absolute Time” “Relative Time”
Parameter with a deduced type	“Deduced”
Telecommand packet	“Any TC”
Telemetry packet	“Any TM”
A data structure following the structure rule set #1 defined in clause 23.	“Any”

## 5.3 Telecommand packet structure

### 5.3.1 Overview

All telecommand packets shall conform to the structure defined in CCSDS 203.0-B-2 and shown in Figure 3. Standardization of the shaded part of the telecommand packet structure is a primary purpose of this Standard.

Packet Header (48 Bits)							Packet Data Field (Variable)			
Packet ID				Packet Sequence Control		Packet Length	Data Field Header (Optional) (see Note 1)	Application Data	Spare	Packet Error Control (see Note 2)
Version Number (=0)	Type (=1)	Data Field Header Flag	Application Process ID	Sequence Flags	Sequence Count					
3	1	1	11	2	14					
16				16		16	Variable	Variable	Variable	16

**Figure 3: Telecommand packet fields**

NOTE 1 Although the data field header field is indicated as optional in CCSDS 203.0-B-2, this Standard specifies it to be mandatory for all telecommand packets except for the CPDU telecommands described in annex D and clause 7 of this Standard, which have no data field header. However, for

consistency with other telecommand packets, future missions may choose to include a data field header for CPDU telecommands, if this is feasible for their CPDU implementation.

NOTE 2 The packet error control field is not explicitly identified in CCSDS 203.0-B-2. However, this Standard specifies it to be mandatory for all telecommand packets.

### 5.3.2 Packet header

#### Packet ID (16 bits)

Version Number:

By changing the version number, future variations of the telecommand packet structure can be introduced. At present, however, only one version number shall be used: (value = 0).

Type:

This bit distinguishes between telecommand packets and telemetry source packets. For telecommand packets, the type = 1.

Data Field Header Flag:

This indicates the presence or absence of a data field header. With the exception of CPDU telecommands, all telecommand packets shall have a data field header, for which this bit shall be set to 1.

Application Process ID (APID):

The APID corresponds uniquely to an on-board application process which is the destination for this packet. The choice of APID values is mission-specific.

#### Packet sequence control (16 bits)

Sequence Flags:

The sequence flags are intended for use if a series of packets are sent in a particular sequence. The interpretation of the sequence flags shall be:

- 01 means first packet of a sequence of packets;
- 00 means continuation packet;
- 10 means last packet of a sequence of packets;
- 11 means “stand-alone” packet.

All standard telecommand packets defined within this Standard are stand-alone packets.

Sequence Count:

This field is provided to identify a particular telecommand packet so that it can be traced within the end-to-end telecommand system. A separate sequence count shall be maintained by each telecommand source for each APID.

When an acknowledgement of a packet is generated (see “Ack” field in the data field header), the packet sequence control field shall be included in the telemetry acknowledge packet as the identifier of the telecommand packet being acknowledged.

**Packet length (16 bits)**

The packet length field specifies the number of octets contained within the packet data field. The number shall be an unsigned integer “C” where:

$$C = (\text{Number of octets in packet data field}) - 1$$

NOTE The actual length of the entire telecommand packet, including the packet header, is 6 octets longer. The largest telecommand packet length is 65 542 octets.

**5.3.3 Packet data field****Data Field Header (variable length)**

The data field header is the subject of definition within this Standard; its content depends on the nature of the service requests defined in the remainder of this Standard; however all data field headers shall have the same basic structure, as follows:

CCSDS Secondary Header Flag	TC Packet PUS Version Number	Ack	Service Type	Service Subtype	Source ID	Spare
Boolean (1 bit)	Enumerated (3 bits)	Enumerated (4 bits)	Enumerated (8 bits)	Enumerated (8 bits)	Enumerated (n bits)	Fixed BitString (n bits)

← Optional → ← Optional →

**CCSDS Secondary Header Flag:**

According to CCSDS 203.0-B-2, this bit shall be set to zero to indicate that the PUS data field header is a “non-CCSDS defined secondary header”.

**TC Packet PUS Version Number:**

By changing the PUS version number, future variations of the telecommand packet can be introduced. Users of this Standard shall use version 1 (value = 1)<sup>3)</sup>.

**Ack:**

This field shall be used to indicate which acknowledgements, in the form of telecommand verification packets, shall be sent to the ground to notify acceptance and to verify execution of this telecommand packet. This shall relate only to acknowledgement of successful acceptance and execution, since failure reports are generated by default; all zeros means no success acknowledgements shall be sent.

In addition to informing the on-board application process of the various telecommand verification packets expected, the ground system also knows what stages of verification to expect for this telecommand packet and can report on “packets uplinked but not yet verified”.

The bit settings corresponding to these stages are as follows:

- 1 (bit 3 of the Ack field set): acknowledge acceptance of the packet by the application process
- 1- (bit 2 of the Ack field set): acknowledge start of execution
- 1-- (bit 1 of the Ack field set): acknowledge progress of execution
- 1--- (bit 0 of the Ack field set): acknowledge completion of execution (whether successful or failed)

3) Version 0 was used by the ESA PUS.

A binary value of “1111” in the Ack field indicates that acknowledgement of at least 4 different steps in the telecommand execution is expected in the telemetry.

**NOTE** Acknowledgement packets are routed only to the source of the original telecommand. This means, for instance, that the ground system only receives acknowledgements for those telecommands that originate from the ground system and not for any telecommands that can be generated autonomously on-board.

#### Service Type:

This indicates the service to which this packet relates.

Service types 0 to 127 shall be reserved for this Standard, service types 128 to 255 are mission-specific.

#### Service Subtype:

Together with the service type, the subtype uniquely identifies the nature of the service request constituted by this telecommand packet.

Within standard services, subtypes 0 to 127 shall be reserved for this Standard, subtypes 128 to 255 are mission-specific. Within mission-specific services, all subtypes (0 to 255) are available for mission-specific use.

The definition of service type and subtype is unique across all application processes.

#### Source ID:

This field indicates the source of the telecommand packet.

**EXAMPLE** Different control centres on the ground or a given on-board source.

The length of this field is known implicitly for the mission.

This field shall be systematically omitted if the mission has only one telecommand source or has no requirement to distinguish between sources.

#### Spare:

Spare bits shall be introduced, in order to pad the data field header to an integral number of words (octets or longer, if, for example, the on-board processors are based on a 16-bit, 24-bit, 32-bit or 64-bit architecture).

All spare bits used for padding purposes shall be set to zero.

#### Application Data (variable length)

The telecommand application data constitutes the data element of the service requests from the service user (e.g. the ground system).

#### Spare (variable length)

In order that the overall packet size is an integral number of words, spare bits may be used for padding purposes at the end of the application data.

#### Packet Error Control (PEC)(16 bits)

The packet error control field transports an error detection code that is used by the receiving application process to verify the integrity of the complete telecommand packet. The type of the PEC (either a checksum or CRC) is fixed for the complete mission for each application process and is defined by the mission constant <TC\_CHECKSUM\_TYPE>.

## 5.4 Telemetry source packet structure

### 5.4.1 Overview

All telemetry source packets shall conform to the structure defined within CCSDS 102.0-B-5 and shown in Figure 4. Standardization of the shaded part of the telemetry packet structure is a primary purpose of this Standard.

Packet Header (48 Bits)						Packet Data Field (Variable)				
Packet ID				Packet Sequence Control		Packet Length	Data Field Header (Optional) (see Note 1)	Source Data	Spare (Optional)	Packet Error Control (Optional)
Version Number (=0)	Type (=0)	Data Field Header Flag	Application Process ID	Grouping Flags	Source Sequence Count					
3	1	1	11	2	14					
16				16		16	Variable	Variable	Variable	(see Note 2)

**Figure 4: Telemetry source packet fields**

NOTE 1 Although the data field header is indicated as optional in CCSDS 102.0-B-5, this Standard specifies its inclusion for all telemetry source packets except for the “spacecraft time source packet” described in annex C and clause 13 of this Standard, which has no data field header. However, for consistency with other telemetry packets, future missions may choose to include a data field header for the spacecraft time source packet, if this is feasible for their on-board implementation.

NOTE 2 This (optional) field is not defined in CCSDS 102.0-B-5, but is introduced in this Standard. When present, it shall be 16 bits in length.

### 5.4.2 Source packet header

#### Packet ID (16 bits)

Version Number:

By changing the version number, future variations of the telemetry source packet structure can be introduced. At present, however, only one version number shall be used: (value = 0).

Type:

This bit distinguishes between telecommand packets and telemetry source packets. For telemetry source packets, the type = 0.

Data Field Header Flag:

This indicates the presence or absence of a data field header. With the exception of the spacecraft time source packet, all telemetry source packets shall have a data field header, for which this bit shall be set to 1.

**Application Process ID (APID):**

The APID corresponds uniquely to an on-board application which is the source of this packet. The choice of Application Process ID values is mission-specific.

APID = 0 is reserved for the time packet.

APID = "1111111111" (eleven ones) is reserved for idle packets.

**Packet Sequence Control (16 bits)****Grouping Flags:**

The grouping flags shall be used when a number of telemetry source packets originating from the same application process are sent in a group. The interpretation of the grouping flags shall be:

- 01 means first packet of a group of packets;
- 00 means continuation packet;
- 10 means last packet of a group of packets;
- 11 means "stand-alone" packet.

All standard telemetry source packets defined within this Standard are stand-alone packets.

**Source Sequence Count:**

A separate source sequence count shall be maintained by each APID and shall be incremented by 1 whenever it releases a packet. If the application process can send distinct packets to distinct destinations using the optional Destination ID field shown below, then a separate source sequence count is maintained for each destination. Therefore the counter corresponds to the order of release of packets by the source and enables the destination (e.g. the ground system) to detect missing packets. This counter should never re-initialize; however, under no circumstances shall it "short-cycle" (i.e. have a discontinuity other than to a value zero). The counter wraps around from  $2^{14}-1$  to zero.

**Packet Length (16 bits)**

The packet length field specifies the number of octets contained within the packet data field. The number shall be an unsigned integer "C" where:

$$C = (\text{Number of octets in packet data field}) - 1$$

NOTE The actual length of the entire telemetry source packet, including the source packet header, is 6 octets longer.

### 5.4.3 Packet data field

#### Data Field Header (variable length)

The data field header shall be the subject of definition within this Standard; its content depends on the nature of the service reports defined in the remainder of this Standard; however all data field headers have the same basic structure, as follows:

Spare	TM Source Packet PUS Version Number	Spare	Service Type	Service Subtype	Packet Sub- counter	Destination ID	Time	Spare
Fixed BitString (1 bit)	Enumerated (3 bits)	Fixed BitString (4 bits)	Enumerated (8 bits)	Enumerated (8 bits)	Unsigned Integer (8 bits)	Enumerated	Absolute Time	Fixed BitString (n bits)

◀ Optional ▶ ◀ Optional → ◀ Optional ▶ ◀ Optional →

#### Spare:

To maintain symmetry with the telecommand packet data field header, this bit shall be reserved and set to zero.

NOTE In the previous version of CCSDS 102.0-B-5, this was a “CCSDS secondary header flag”.

#### TM Source Packet PUS Version Number:

By changing the PUS version number, future variations of the telemetry source packet can be introduced. It is expected to introduce a new version in the future for telemetry source packets that contain more than one service data unit.

Users of this Standard shall use version 1 (value = 1)<sup>4)</sup>

#### Spare:

Spare bits shall be introduced in order to make up an integral octet. These spare bits shall be set to zero.

#### Service Type:

This indicates the service to which this telemetry source packet relates.

Service types 0 to 127 shall be reserved for this Standard, service types 128 to 255 are mission-specific.

#### Service Subtype:

Together with the service type, the subtype uniquely identifies the nature of the service report constituted by this telemetry source packet.

Within standard services, subtypes 0 to 127 shall be reserved for this Standard, subtypes 128 to 255 are mission-specific. Within mission-specific services, all subtypes (0 to 255) are available for mission-specific use.

The definition of service type and subtype shall be unique across all application processes and the combination of these fields can be used on the ground to determine the processing priority level.

#### Packet Sub-counter:

This is a counter that increments each time the application process releases a new packet of this type and subtype. If the application process can send distinct packets to distinct destinations using the optional Destination ID field shown below, then a separate source sequence count shall be

4) Version 0 was used by the ESA PUS.



maintained for each destination. Whilst the source sequence count enables the destination (e.g. the ground) to detect a transmission gap, this packet sub-counter gives more information on the nature of the particular packets that are missing. This field shall be systematically omitted if this information is not applicable for this application process.

#### Destination ID:

This field identifies the destination of the telemetry source packet. The length of this field is known implicitly for the mission. This field shall be systematically omitted if the mission has only a single destination (i.e. the ground system) for telemetry source packets or has no requirement to distinguish between destinations.

#### Time:

The on-board reference time of the packet, expressed in either the CUC or CDS format (as defined in CCSDS 301.0-B-2). The choice between CUC and CDS formats and the resolution of the time field can vary from mission to mission and even between different application processes within the same satellite. For some missions or application processes, this time field may even not be present. The presence or absence of the field and its encoding are explicitly defined by the mission constants <APPL\_TIME\_CODE> of which there are as many as there are on-board application processes. The representations of the time field are those defined in clause 23 of this Standard.

If the CDS format is used, the standard CCSDS epoch of 1958 January 1 shall be applicable, see CCSDS 301.0-B-2.

The packet on-board time corresponds to any well-defined time before the sampling of any data within the packet.

#### Spare:

Spare bits shall be introduced in order to pad the data field header to an integral number of words. These spare bits shall be set to zero.

#### Source Data (variable length)

The telemetry source data constitutes the data element of the service reports to the service user (e.g. the ground system).

#### Spare (variable length)

In order that the overall packet size is an integral number of words, spare bits may be used for padding purposes at the end of the source data.

#### Packet Error Control (optional, 16 bits)

The packet error control field transports an error detection code that can be used by the ground system to verify the integrity of the complete telemetry source packet. The presence of the PEC and its type (either a checksum or CRC) shall be fixed for the complete mission for each application process and defined by the mission constant <TM\_CHECKSUM\_TYPE>.

## 5.5 Standard services

The standard services listed in Table 2 are specified within the following clauses. The services can have a minimum capability set and one or more additional capability set(s); this is indicated in the corresponding service clause.

For each service, the minimum capability set corresponds to the minimum coherent implementation of that service.

**Table 2: Standard services specified within this Standard**

Service Type	Service Name
1	Telecommand verification service
2	Device command distribution service
3	Housekeeping & diagnostic data reporting service
4	Parameter statistics reporting service
5	Event reporting service
6	Memory management service
7	Not used
8	Function management service
9	Time management service
10	Not used
11	On-board operations scheduling service
12	On-board monitoring service
13	Large data transfer service
14	Packet forwarding control service
15	On-board storage and retrieval service
16	Not used
17	Test service
18	On-board operations procedure service
19	Event-action service

---

## Telecommand verification service

### 6.1 Scope

The telecommand verification service provides the capability for explicit verification of each distinct stage of execution of a telecommand packet, from on-board acceptance through to completion of execution. In this sense, it is a supporting service for the telecommand packets (service requests) belonging to all other standard and mission-specific services.

Although this service provides the possibility for explicit telecommand verification, there is no implication that all telecommands for a given mission shall necessarily be verifiable at each distinct stage. For many telecommands, the application process can have little or no knowledge of the command execution profile or effects. The extent to which the telecommand verification process is supported on-board for a given telecommand is specified on a mission-specific basis.

This service satisfies the concepts described in subclause 4.4.3 of this Standard.

Any number of on-board application processes may provide a single instance of the telecommand verification service.

Telemetry source packets and telecommand packets belonging to the telecommand verification service shall be denoted by Service Type = 1.

### 6.2 Service concept

The following stages of telecommand processing are identified:

- a. Acceptance of the telecommand by the destination application process. At this stage of telecommand processing, all checks to be applied to the telecommand packet prior to the start of execution shall be performed. This shall include verification that the telecommand has not been corrupted (by checksumming the telecommand) and that the application process supports the service or sub-service, requested. This element of the telecommand verification service should therefore be supported for all telecommands.
- b. Start execution of the telecommand. This stage may follow immediately after reception, but for a complex command it may also be delayed pending some level of pre-execution validation. The checks performed at this stage shall include feasibility and validity checks, such as confirmation that the telecommand parameters are correctly encoded, are within their range of values and are valid for the current state of the service.

- c. Progress of execution. The various steps which reflect the progress of execution of the telecommand shall be telecommand-specific in nature.
- d. Completion of execution of the telecommand.

The telecommand verification service shall generate a report if a telecommand fails at any of its identified stages of execution. It shall also generate a report of successful completion of the same stages, if this has been requested in the acknowledgement flags in the telecommand packet header. These reports shall provide auxiliary data for the ground system to fully understand the report (e.g. to identify the nature and cause of a telecommand failure).

Since the distinction between telemetry reports of telecommand verification success and failure is extremely important operationally, different subtypes are defined for these distinct outcomes. This shall facilitate the selection of telecommand verification packets on this basis, for example, for the purposes of priority routing or storage either on-board or on ground.

## 6.3 Service requests and reports

### 6.3.1 General

Currently no user-initiated service requests are defined.

### 6.3.2 Telecommand acceptance

The reports of acceptance of a telecommand packet shall be as follows:

#### Telecommand Acceptance Report - Success (1,1)

Telemetry source packet, source data:

Telecommand Packet ID	Packet Sequence Control
2 octets	2 octets

Telecommand Packet ID:

This is a copy of the corresponding fields from the packet header of the telecommand to which this verification packet relates.

Packet Sequence Control:

This is a copy of the corresponding fields from the packet header of the telecommand to which this verification packet relates.

## Telecommand Acceptance Report - Failure (1,2)

Telemetry source packet, source data:

Telecommand Packet ID	Packet Sequence Control	Code	Parameters
2 octets	2 octets	Enumerated	Any

Optional

Code:

The code indicates the reason for the failure of the telecommand at this verification stage.

At the acceptance stage, the following standard code values are defined:

0 = illegal APID (PAC error);

1 = incomplete or invalid length packet;

2 = incorrect checksum;

3 = illegal packet type;

4 = illegal packet subtype;

5 = illegal or inconsistent application data.

Other values of the code can be application process specific or command-specific (i.e. dependent on combinations of the type, subtype and individual command function). The code is an identifier for the auxiliary information provided with this report, i.e. the parameters field.

Parameters:

The parameters field provides complementary information relating to the particular value of the code field. For full interpretation of failure of a command, knowledge of the nature of the command shall be available. The parameters field shall conform to the structure rules set #1 and each parameter shall be one of the data types defined in this Standard (see clause 23).

### 6.3.3 Telecommand execution started

The reports of start of execution of a telecommand packet shall be as follows:

#### Telecommand Execution Started Report - Success (1,3)

Telemetry source packet, source data: Same as for subtype 1.

#### Telecommand Execution Started Report - Failure (1,4)

Telemetry source packet, source data:

Telecommand Packet ID	Packet Sequence Control	Code	Parameters
2 octets	2 octets	Enumerated	Any

Optional

### 6.3.4 Telecommand execution progress

The reports of progress of execution of a telecommand packet shall be as follows:

#### Telecommand Execution Progress Report - Success (1,5)

Telemetry source packet, source data:

Telecommand Packet ID	Packet Sequence Control	Step Number
2 octets	2 octets	Enumerated

Step Number:

This indicates the intermediate step number of the telecommand execution profile whose execution has been completed. The values it can take are telecommand-specific.

#### Telecommand Execution Progress Report - Failure (1,6)

Telemetry source packet, source data:

Telecommand Packet ID	Packet Sequence Control	Step Number	Code	Parameters
2 octets	2 octets	Enumerated	Enumerated	Any



### 6.3.5 Telecommand execution complete

The reports of completion of execution of a telecommand packet shall be as follows:

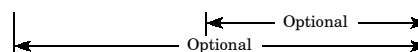
#### Telecommand Execution Completed Report - Success (1,7)

Telemetry source packet, source data: Same as for subtype 1.

#### Telecommand Execution Completed Report - Failure (1,8)

Telemetry source packet, source data:

Telecommand Packet ID	Packet Sequence Control	Code	Parameters
2 octets	2 octets	Enumerated	Any



## 6.4 Capability sets

### 6.4.1 Minimum capability set

The minimum capability set shall consist of the service reports given in Table 3.

**Table 3: Summary of telecommand verification service minimum capabilities**

Subtype	Service request, report or capability
1	Telecommand Acceptance Report - Success
2	Telecommand Acceptance Report - Failure

### 6.4.2 Additional capability sets

The telecommand verification service can implement the additional capabilities given in Table 4. These options can be effected at the level of individual telecommands.

**Table 4: Summary of telecommand verification service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
3	Telecommand Execution Started Report - Success	4
4	Telecommand Execution Started Report - Failure	3
5	Telecommand Execution Progress Report - Success	6
6	Telecommand Execution Progress Report - Failure	5
7	Telecommand Execution Completed Report - Success	8
8	Telecommand Execution Completed Report - Failure	7

*(This page is intentionally left blank)*



---

## Device command distribution service

### 7.1 Scope

The device command distribution service provides the capability for the distribution of:

- a. On-Off and register load commands,
- b. Command pulse distribution unit (CPDU) commands for the re-configuration of vital spacecraft functions.

This service satisfies the concepts described in subclause 4.4.2 of this Standard.

Any number of on-board application processes may provide a single instance of the device command distribution service.

Telemetry source packets and telecommand packets belonging to the device command distribution service shall be denoted by Service Type = 2.

### 7.2 Service concept

#### 7.2.1 On-Off and register load commands

Following reception of a service request containing On-Off commands or register load commands:

- a. The commands shall be unpacked (if more than one command is present); an On-Off command consists of an On-Off address; a register load command consists of a register address and data to be loaded in the register.
- b. The command(s) shall be reformatted, to the structure applicable by an internal command bus or the destination device driver. The structure of the address field is mission-dependent and may be comprised of several subfields.
- c. The command(s) shall be distributed. If the packet contains more than one command, these shall be distributed in the same sequence in which they occur within the packet.

An example of an OBDH 24-bit command accommodated within this service data unit structure is given in annex A.3.

### 7.2.2 CPDU commands

A CPDU is a simple redundant unit, solely accessible from the ground, which can issue command pulses of specified duration on output lines driving vital spacecraft actuators. Each CPDU can have up to 256 output lines.

Each CPDU has a dedicated APID and can receive “CPDU telecommand packets” containing command pulse instructions. A CPDU telecommand packet has no data field header but has a packet error control field and shall be carried inside a single telecommand segment.

Following reception of a CPDU telecommand packet, it shall be checked as defined in annex D and if no error is found, the CPDU shall process the command pulse instructions in the same sequence as they occur within the packet.

For a given implementation, there shall be a limit on the number of command pulse instructions in a CPDU telecommand packet (at least 12 and at most 120), identified by the mission constant <CPDU\_MAX\_INSTR>.

## 7.3 Service requests and reports

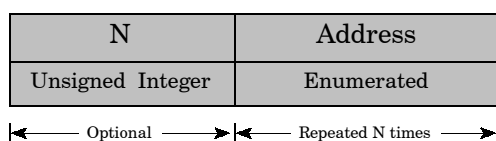
Currently no service reports are defined.

### 7.3.1 Distributing On-Off commands

The request for the distribution of On-Off command(s) by means of a telecommand packet shall be:

#### Distribute On-Off Commands (2,1)

Telecommand packet, application data:



N:

The number of On-Off commands which follow ( $N > 0$ ).

This field shall be systematically omitted if the service only supports the distribution of one On-Off command at a time (i.e.  $N=1$ ).

Address:

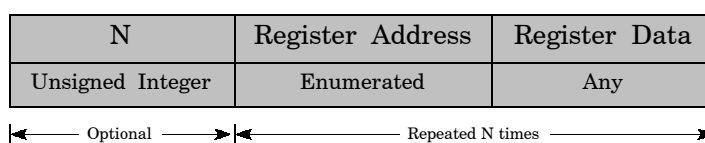
This gives the hardware address to which the On-Off command shall be routed.

### 7.3.2 Distributing register load commands

The request for the distribution of register load command(s) by means of a telecommand packet shall be:

#### Distribute Register Load Commands (2,2)

Telecommand packet, application data:



N:

The number of register load commands which follow ( $N > 0$ ).

This field shall be systematically omitted if the service only supports the distribution of one register load command at a time (i.e.  $N=1$ ).

**Register Address:**

This gives the hardware address of the register.

**Register Data:**

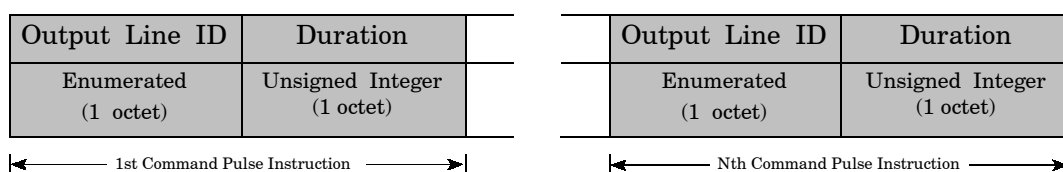
The register data consist of a set of parameters whose structure is known implicitly from the foregoing register address and which conforms to the structure rule set #1 (see clause 23).

### 7.3.3 Distributing CPDU commands

The request for the generation of Command Pulses on the output lines of a CPDU shall be:

#### Distribute CPDU Commands (2,3)

CPDU telecommand packet, application data:

**Output Line ID:**

This identifies the CPDU output line on which the command pulse is issued. The values it takes are mission specific.

**Duration:**

A value between 0 and 8 which determines the duration of the command pulse as follows:

$$\text{Command pulse duration} = \langle \text{CPDU\_DURATION\_UNIT} \rangle * 2^{\text{Duration}}$$

where  $\langle \text{CPDU\_DURATION\_UNIT} \rangle$  is defined for the CPDU (between 10 ms and 15 ms).

The number of command pulse instructions in the CPDU telecommand packet is variable.

On reception of this request, the CPDU shall perform the processing outlined in subclause 7.2.2 and specified in annex D.

## 7.4 Capability sets

### 7.4.1 Minimum capability set

The minimum capability set shall consist of either:

- Distribute CPDU Commands (2,3) (since this is supported by a dedicated application process); or
- one of the service requests given in Table 5.

**Table 5: Summary of device command distribution service minimum capabilities**

Subtype	Service request, report or capability
1	Distribute On-Off Commands
2	Distribute Register Load Commands

### 7.4.2 Additional capability sets

A device command distribution service that does not implement subtype 3 “Distribute CPDU Commands” can also implement one of the additional service requests given in Table 6.

**Table 6: Summary of device command distribution service additional capabilities**

Subtype	Service request, report or capability
1	Distribute On-Off Commands
2	Distribute Register Load Commands

---

## Housekeeping and diagnostic data reporting service

### 8.1 Scope

This service, along with the parameter statistics reporting and event reporting services, provides for the reporting to the ground of all information of operational significance that is not explicitly provided within the reports of another service.

The service consists of two independent sub-services which cover, respectively, the requirements for:

- a. housekeeping data reporting (both periodic and filtered in nature);
- b. diagnostic data reporting.

This service satisfies the concepts described in subclauses 4.5.1 and 4.14 of this Standard.

Any number of on-board application processes may provide a single instance of the housekeeping and diagnostic data reporting service.

Telemetry source packets and telecommand packets relating to the housekeeping and diagnostic data reporting service shall be denoted by Service Type = 3.

### 8.2 Service concept

#### 8.2.1 Housekeeping service concept

##### 8.2.1.1 General

The housekeeping data reporting sub-service shall sample sets of housekeeping parameters in accordance with a set of reporting definitions stored on-board. There is a predefined set of such definitions on-board (designed by the satellite manufacturer) for the housekeeping monitoring of the mission. However, these definitions may be modified or deleted and new definitions may be added, modified or deleted by the ground system, at any time.

A structure identification (SID) is associated with each distinct reporting definition and associated housekeeping parameter report. The SID shall be used on the ground, together with the APID and knowledge of the nature of the packet (i.e. that it is a housekeeping packet, as opposed to a diagnostic packet) to identify the housekeeping parameters report and to interpret its content. The SID shall be

unique to a given service implementation and packet nature (i.e. housekeeping or diagnostic), however different instances of the service within different application processes can use the same values of SID.

#### **8.2.1.2 Data collection**

Each reporting definition has an associated data collection interval, which is the time interval over which the housekeeping parameters are sampled.

Parameters within a reporting definition can be sampled once per collection interval or more than once. Parameters sampled more than once are sampled regularly in time at a rate which shall be determined on-board from the combination of the data collection interval and the number of samples of the parameter in the report.

#### **8.2.1.3 Parameter report generation**

Two modes of generating housekeeping parameter reports shall exist:

- a. Periodic mode, where a housekeeping parameter report is generated once for each collection interval.
- b. “Filtered” mode, where a housekeeping parameter report is only generated at the end of the collection interval when the value of one or more parameters in the report has changed by more than a given threshold since the previous collection interval. A subset of parameters to be filtered in this manner can be selected; this subset and the thresholds to be used are specified by the ground system. In the case of parameters sampled once per collection interval, the value from the current collection interval is compared with the value from the previous collection interval. In the case of parameters sampled more than once, the change of value of one or more samples of the parameter since the last collection interval constitutes an event for this purpose.

In addition, the data sampling, data collection and parameter-report-generation activities for a housekeeping parameter report may be temporarily disabled (e.g. to reduce the on-board processing load).

The requests for selecting the mode of housekeeping parameter report generation shall be part of this service, but the requests for disabling and enabling the forwarding of housekeeping parameter reports to the ground system shall be part of the generic packet forwarding control service (see clause 17).

Filtered mode is expected to be useful primarily for status-type parameters, where a report is only transmitted to the ground system when a significant change of status has occurred. However, in principle, any parameter within the reporting definition can be used to trigger a filtered parameter report. Those parameters that are not used for this purpose shall be “filtered out” from the comparison process whilst all unfiltered parameters shall be used. To limit the circumstances under which the parameter report is generated, a threshold for the change shall be defined, on a parameter basis, which is expressed either as a percentage or an (absolute) delta change in parameter value.

Finally, for filtered mode, the concept of parameter-report time-out is introduced whereby if no change has been detected, and hence no report generated, for a pre-specified time, a one-shot report shall be generated. This measure ensures that extensive periods of time do not elapse without the occurrence of a given (enabled) housekeeping parameter report.

#### 8.2.1.4 Parameter sampling times

Absolute (as well as relative) sampling times of parameters in housekeeping parameter reports shall be determinable to a given accuracy (<PARAM\_ABS\_SAMPL\_TIME> and <PARAM\_REL\_SAMPL\_TIME>) on the ground. Three alternative methods can be used to satisfy these requirements:

- a. The parameter sampling times (e.g. time offsets with respect to packet time) can be deduced from knowledge of the on-board parameter sampling mechanism.
- b. Telemetry parameters can be explicitly time-stamped in the telemetry.

EXAMPLE An on-board time register can be provided, which is sampled and telemetered along with each sample of the parameter.

Alternatively, the application process responsible for sampling can measure the sampling time and insert this time in the telemetry. However, both of these solutions are costly in terms of packet overhead.

- c. The sampling time offsets (with respect to packet time) can be measured (or otherwise evaluated) on-board over a given collection interval and reported to the ground system. The sampling “pattern” and hence the time offsets are then assumed to hold true for all housekeeping parameter reports.

For case c., explicit reporting (on request) of the measured time offsets shall be supported.

#### 8.2.2 Diagnostic service concept

The diagnostic data reporting sub-service shall be functionally identical to the housekeeping data reporting sub-service. Different service subtypes shall be used, however, primarily for the purposes of distinguishing the diagnostic parameter reports for routing and (ground) processing.

A means to disable the generation of certain diagnostic parameter reports (whose definitions can remain on-board for intermittent use, for example, when a particular anomaly occurs) shall be provided. Because of the nature of diagnostic mode, it is anticipated that the parameter reports contain a predominance of fixed-length arrays corresponding to parameters sampled at very high rates, many times per report.

## 8.3 Service requests and reports

### 8.3.1 Defining new housekeeping or diagnostic parameter reports

The requests which provide the definition of a new housekeeping or diagnostic parameter report shall be:

#### Define New Housekeeping Parameter Report (3,1)

Telecommand packet, application data:

#### Define New Diagnostic Parameter Report (3,2)

Telecommand packet, application data:

SID	Collection Interval	NPAR1	Parameter#
Enumerated	Unsigned Integer	Unsigned Integer	Enumerated

← Optional →

← Repeated NPAR1 times →

NFA	NREP	NPAR2	Parameter#
Unsigned Integer	Unsigned Integer	Unsigned Integer	Enumerated

← Repeated NPAR2 times →

← Repeated NFA times →

SID:

The structure identification which corresponds to this reporting definition.

Collection Interval:

The data collection interval for this housekeeping or diagnostic parameter report definition, expressed in units of <DIAG\_MIN\_INTERV>.

This field shall be systematically omitted if the service uses a default value for the collection interval.

NPAR1:

The number of parameters in the definition that are sampled once per collection interval.

Parameter#:

The parameter number to be sampled. A “parameter number” is used on-board, for optimization purposes. It has a unique correspondence with the “Parameter ID” which is used on the ground for identification purposes.

NFA:

The number of fixed-length arrays (see subclause 23.6.2 in this definition).

NREP:

The number of values to be sampled for each parameter within this fixed-length array.

NPAR2:

The number of different parameters within this fixed-length array, each of which shall be sampled “NREP” times per collection interval.



When this request is received, the new housekeeping or diagnostic parameter report definition is recorded, a corresponding “Report Generation Flag” is created and this flag is set to “Disabled”. The generation mode for this parameter report shall also be set to “Periodic” by default.

### 8.3.2 Clearing housekeeping or diagnostic parameter report definitions

The requests to clear one or more housekeeping or diagnostic parameter report definitions shall be:

#### Clear Housekeeping Parameter Report Definitions (3,3)

Telecommand packet, application data:

#### Clear Diagnostic Parameter Report Definitions (3,4)

Telecommand packet, application data:

NSID	SID
Unsigned Integer	Enumerated

|← Optional →| |← Repeated NSID times →|

NSID:

This field shall be systematically omitted if the service is only able to clear one parameter report definition at a time (i.e. NSID=1).

When this request is received, the entries for the indicated housekeeping or diagnostic parameter report definitions and the corresponding parameter report generation mode and report generation flags shall be cleared (released).

### 8.3.3 Controlling the generation of housekeeping or diagnostic parameter reports

The requests to enable and disable the generation of one or more housekeeping or diagnostic parameters report definitions shall be:

#### Enable Housekeeping Parameter Report Generation (3,5)

Telecommand packet, application data:

#### Disable Housekeeping Parameter Report Generation (3,6)

Telecommand packet, application data:

#### Enable Diagnostic Parameter Report Generation (3,7)

Telecommand packet, application data:

#### Disable Diagnostic Parameter Report Generation (3,8)

Telecommand packet, application data:

NSID	SID
Unsigned Integer	Enumerated

|← Optional →| |← Repeated NSID times →|

NSID:

This field shall be systematically omitted if the service can only enable or disable the generation of one parameter report definition at a time (i.e. NSID=1).

When this request is received, the activities for the generation of the indicated housekeeping or diagnostic parameter reports shall be started (stopped).

### 8.3.4 Reporting housekeeping or diagnostic parameter report definitions

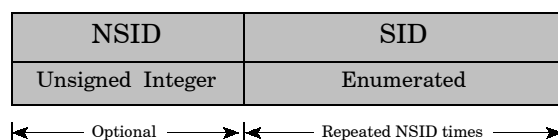
The requests for a report of one or more housekeeping or diagnostic parameter report definitions shall be:

#### Report Housekeeping Parameter Report Definitions (3,9)

Telecommand packet, application data:

#### Report Diagnostic Parameter Report Definitions (3,11)

Telecommand packet, application data:



NSID:

This field shall be systematically omitted if the service can only report one parameter report definition at a time (i.e. NSID=1).

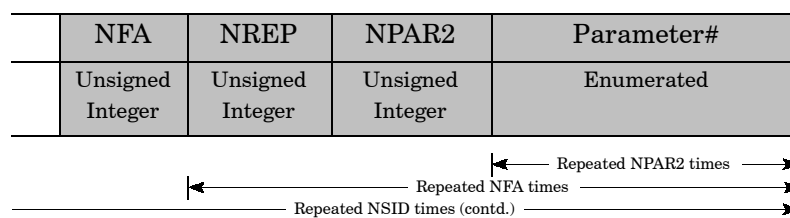
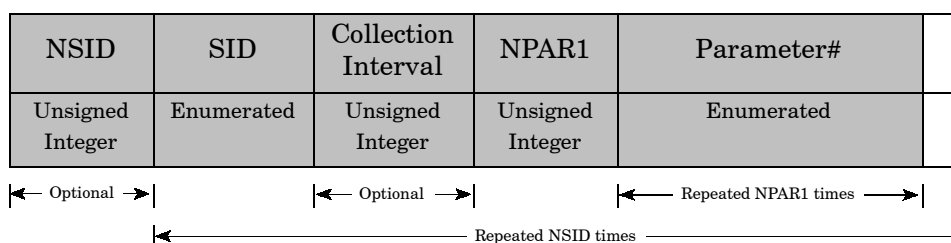
When this request is received, the following response shall be generated containing the requested housekeeping or diagnostic parameter report definitions.

#### Housekeeping Parameter Report Definitions Report (3,10)

Telemetry source packet, source data:

#### Diagnostic Parameter Report Definitions Report (3,12)

Telemetry source packet, source data:



NSID:

This field shall be systematically omitted if the service can only report one parameter report definition at a time (i.e. NSID=1).

The definitions (SIDs) shall be reported in the same sequence as in the corresponding request (subtype 9 or 11 above).

### 8.3.5 Reporting housekeeping or diagnostic parameter sampling-time offsets

The requests for a report of housekeeping or diagnostic parameter sampling-time offsets for a given housekeeping or diagnostic parameter report shall be:

#### Report Housekeeping Parameter Sampling-Time Offsets (3,13)

Telecommand packet, application data:

#### Report Diagnostic Parameter Sampling-Time Offsets (3,14)

Telecommand packet, application data:

SID
Enumerated

When this request is received, the sampling-time offsets for the next housekeeping or diagnostic parameters report of this reporting definition (structure ID) shall be measured (or evaluated) and reported as follows. This report shall also be automatically generated (as a one-shot report) whenever a new structure ID is defined and before the first transmission of a corresponding housekeeping or diagnostic parameter report.

#### Housekeeping Parameter Sampling-Time Offset Report (3,15)

Telemetry source packet, source data:

#### Diagnostic Parameter Sampling-Time Offset Report (3,16)

Telemetry source packet, source data:

SID	Time Offset 1st Parameter	...	Time Offset Last Parameter
Enumerated	Relative Time	...	Relative Time

The (NPAR1 + NPAR2) time offsets correspond to the order of the parameters in the corresponding housekeeping or diagnostic parameter report. For a fixed-length array (see subclause 23.6.2, only the time offset of the first occurrence of a parameter within the array shall be reported (the time offsets of subsequent samples are derivable from a knowledge of the parameter sampling interval).

### 8.3.6 Selecting housekeeping or diagnostic parameter report generation mode

The requests to select the periodic generation mode for a given housekeeping or diagnostic reporting definition shall be:

#### Select Periodic Housekeeping Parameter Report Generation Mode (3,17)

Telecommand packet, application data:

#### Select Periodic Diagnostic Parameter Report Generation Mode (3,18)

Telecommand packet, application data:

SID
Enumerated

When this request is received, the parameter report generation mode for the indicated housekeeping or diagnostic reporting definition shall be changed to periodic mode. If the reporting definition is already in periodic mode, an error report shall be generated.

The requests to select the filtered generation mode for a given housekeeping or diagnostic reporting definition shall be:


**Select Filtered Housekeeping Parameter Report Generation Mode (3,19)**

Telecommand packet, application data:

**Select Filtered Diagnostic Parameter Report Generation Mode (3,20)**

Telecommand packet, application data:

SID	Timeout	N	Parameter#	Threshold Type	Threshold
Enumerated	Unsigned Integer	Unsigned Integer	Enumerated	Enumerated	Unsigned Integer



Timeout:

The timeout for generation of a one-shot housekeeping or diagnostic parameter report expressed as a multiple of the data collection interval for this housekeeping or diagnostic reporting definition.

This field shall be systematically omitted if the service uses a default value for the timeout for the given housekeeping or diagnostic reporting definition.

N:

The number of parameters and their threshold attributes which follow.

This field shall be systematically omitted if the service does not support the concept of filtered parameters or always has knowledge of the list of unfiltered parameters to use for the given housekeeping or diagnostic reporting definition.

Parameter#:

The parameters which are unfiltered (i.e. used) when the values of samples between successive collection intervals are compared.

Threshold Type:

This indicates the type (i.e. interpretation) of the threshold value that follows. This shall be either:

value = 0: a percentage of the value of the parameter, or

value = 1: an absolute delta value.

This field shall be systematically omitted if the service uses a default type for the threshold for the given housekeeping or diagnostic reporting definition or a default threshold value.

Threshold:

The minimum change to be detected between parameter samples in order to declare an event for the purpose of parameters report generation.

This field shall be systematically omitted if the service uses a default value for the threshold for the given housekeeping or diagnostic reporting definition.

When this request is received, the parameter report generation mode for the indicated housekeeping or diagnostic reporting definition shall be changed to filtered mode and the new service parameters supplied in the request shall be used with immediate effect. If the reporting definition is already in filtered mode, an error report shall be generated.

### 8.3.7 Reporting unfiltered housekeeping or diagnostic parameters

The request to report the parameters which are unfiltered (i.e. used) for a given housekeeping or diagnostic reporting definition when the values of samples between successive collection intervals are compared shall be:

#### Report Unfiltered Housekeeping Parameters (3,21)

Telecommand packet, application data:

#### Report Unfiltered Diagnostic Parameters (3,22)

Telecommand packet, application data:

SID
Enumerated

When this request is received, a report shall be generated as follows:

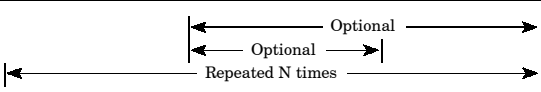
#### Unfiltered Housekeeping Parameters Report (3,23)

Telemetry source packet, source data:

#### Unfiltered Diagnostic Parameters Report (3,24)

Telemetry source packet, source data:

SID	N	Parameter#	Threshold Type	Threshold
Enumerated	Unsigned Integer	Enumerated	Enumerated	Unsigned integer



### 8.3.8 Reporting housekeeping or diagnostic data

The provider-initiated reports of the values of a set of housekeeping or diagnostic parameters shall be:

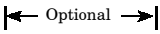
#### Housekeeping Parameter Report (3,25)

Telemetry source packet, source data:

#### Diagnostic Parameter Report (3,26)

Telemetry source packet, source data:

SID	Mode	Parameters
Enumerated	Enumerated	Any



Mode:

Indicates the mode of packet generation and, if filtered, the reason for generation of this packet:

value = 0: Periodic mode;

value = 1: Filtered mode, parameter change exceeded threshold;

value = 2: Filtered mode, but packet generated as a result of timeout.

This field shall be systematically omitted if the service does not support filtered reporting of housekeeping or diagnostic parameters.

**Parameters:**

This field consists of a sequence of values of housekeeping or diagnostic parameters that are sampled once per collection interval followed by a sequence of fixed-length arrays of records. The sequence of parameter values and arrays shall be the same as in the housekeeping or diagnostic parameter report definition request (see subtypes 1 and 2 above).

The only authorized parameter types are those described in clause 23.

For ground system processing purposes, the SID, together with the application process ID and the nature of the packet (housekeeping or diagnostic) implicitly identifies the structure of the parameter field.

## 8.4 Capability sets

### 8.4.1 Minimum capability sets

The minimum capability set for a service implementing the housekeeping sub-service shall be as given in Table 7.

**Table 7: Summary of housekeeping sub-service minimum capabilities**

Subtype	Service request, report or capability
25	Housekeeping Parameter Report

The minimum capability set for a service implementing the diagnostic sub-service shall consist of the service requests and reports given in Table 8.

**Table 8: Summary of diagnostic sub-service minimum capabilities**

Subtype	Service request, report or capability	Requires subtype
2	Define New Diagnostic Parameter Report	4
4	Clear Diagnostic Parameter Report Definitions	2
7	Enable Diagnostic Parameter Report Generation	8
8	Disable Diagnostic Parameter Report Generation	7
26	Diagnostic Parameter Report	

### 8.4.2 Additional capability sets

A housekeeping and diagnostic data reporting service can also implement the following additional capability sets:

- housekeeping report definitions control;
- report definitions reporting;
- sampling time offset reporting;
- filtered mode support.

A housekeeping and diagnostic data reporting service providing the housekeeping report definitions control capability set shall implement all the capabilities given in Table 9.

**Table 9: Summary of report definitions control additional capabilities**

Subtype	Service request, report or capability	Requires subtype
1	Define New Housekeeping Parameter Report	3
3	Clear Housekeeping Parameter Report Definitions	1
5	Enable Housekeeping Parameter Report Generation	6
6	Disable Housekeeping Parameter Report Generation	5

A housekeeping and diagnostic data reporting service providing the report definitions reporting capability set shall implement at least one of the request/report pairs given in Table 10.

**Table 10: Summary of report definitions reporting additional capabilities**

Subtype	Service request, report or capability	Requires subtype
9	Report Housekeeping Parameter Report Definitions	10
11	Report Diagnostic Parameter Report Definitions	12
10	Housekeeping Parameter Report Definitions Report	9
12	Diagnostic Parameter Report Definitions Report	11

A housekeeping and diagnostic data reporting service providing the sampling time offset reporting capability set shall implement at least one of the request/report pairs given in Table 11.

**Table 11: Summary of sampling time offset reporting additional capabilities**

Subtype	Service request, report or capability	Requires subtype
13	Report Housekeeping Parameter Sampling-Time Offsets	15
14	Report Diagnostic Parameter Sampling-Time Offsets	16
15	Housekeeping Parameter Sampling-Time Offsets Report	13
16	Diagnostic Parameter Sampling-Time Offsets Report	14

Whilst the measurement and reporting of parameter sampling time offsets is optional, if not provided it implies that the mission timing accuracy requirements can be fulfilled by one of the other indicated methods.

A housekeeping and diagnostic data reporting service providing filtered mode support shall implement at least one of the related request pairs given in Table 12.

**Table 12: Summary of filtered mode minimum capabilities**

Subtype	Service request, report or capability	Requires subtype
17	Select Periodic Housekeeping Parameter Report Generation Mode	19
18	Select Periodic Diagnostic Parameter Report Generation Mode	20
19	Select Filtered Housekeeping Parameter Report Generation Mode	17
20	Select Filtered Diagnostic Parameter Report Generation Mode	18

A housekeeping and diagnostic data reporting service providing filtered mode support can implement at least one of the request/report pairs given in Table 13.

**Table 13: Summary of filtered mode additional capabilities**

Subtype	Service request, report or capability	Requires subtype
21	Report Unfiltered Housekeeping Parameters	23
22	Report Unfiltered Diagnostic Parameters	24
23	Unfiltered Housekeeping Parameters Report	21
24	Unfiltered Diagnostic Parameters Report	22



---

## Parameter statistics reporting service

### 9.1 Scope

This service provides for the reporting to the ground system of maximum, minimum, mean and standard deviation values of on-board parameters during a time interval. The parameters can be sampled at different frequencies.

This service can be used during periods of ground non-coverage, for example, as an alternative to housekeeping data reporting if the on-board storage capacity is limited.

This service satisfies the concepts described in subclause 4.5.3 of this Standard.

Any number of on-board application processes may provide a single instance of the parameter statistics reporting service.

Telemetry source packets and telecommand packets relating to the parameter statistics reporting service shall be denoted by Service Type = 4.

### 9.2 Service concept

The service shall provide the capability for reporting the maximum, minimum, mean and (optionally) the standard deviation values of a list of parameters. The ground system may add (to a maximum of <PSLIST\_MAX\_PARAMS>) or delete parameters from this list at any time or clear it completely.

The parameters for which these statistics are reported can be sampled at quite different frequencies, depending on the particular characteristics of the parameter (e.g. slowly varying analogue parameter such as a temperature may be sampled at a very low frequency). This sampling interval (per parameter basis) shall be specified when the parameter is added to the list.

The concept for reporting the parameter statistics shall be as follows. The statistics shall be evaluated continuously on the basis of successive samples of each parameter. However, the corresponding reports of the statistics shall be generated either upon request from the ground or periodically. The periodic reporting interval can be specified through ground request.

The evaluation of the parameter statistics can be reset (if requested) when such a report is generated. It shall be systematically reset when the report is periodically generated and shall also be reset if an explicit request to do so is received from the ground at any time.

If there is a requirement to evaluate these parameter statistics during a period of ground non-coverage, the ground system shall reset the evaluation process just prior to loss of signal (LOS) and request the parameter statistics report shortly after acquisition of signal (AOS). This report will then cover the non-coverage period.

Because of the potentially long interval between reports, the time field in the packet header (if present) should indicate the report generation time and not the beginning of the evaluation interval.

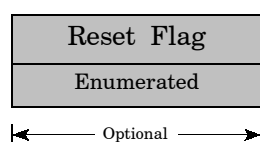
## 9.3 Service requests and reports

### 9.3.1 Reporting the parameter statistics results

The request shall be:

#### Report Parameter Statistics (4,1)

Telecommand packet, application data:



Reset Flag:

This indicates whether the evaluation of the parameter statistics shall be reset or not. Its values are “No” (value=0) and “Yes” (value=1).

This field shall be systematically omitted if the service is designed always to reset the evaluation of the parameter statistics after the report is generated.

When this request is received, a report shall be generated which contains the current parameter statistics values. The evaluation of the parameter statistics shall be reset after the report is generated, if the Reset Flag is “Yes”.

The report shall also be automatically generated at regular intervals by the service if periodic reporting is currently selected. In this case, the evaluation of the parameter statistics shall be systematically reset after the report is generated.

#### Parameter Statistics Report (4,2)

Telemetry source packet, source data:

$t_{\text{start}}$	NPAR	Parameter#	Maxval	$t_{\text{max}}$	Minval	$t_{\text{min}}$	Meanval	Stddevval
Absolute Time	Unsigned Integer	Enumerated	Deduced	Absolute Time	Deduced	Absolute Time	Deduced	Deduced



$t_{\text{start}}$ :

The time at which the evaluation of the parameter statistics started (i.e. the last time the parameter statistics list was reset).

NPAR:

The number of parameters in the parameter statistics list which have been sampled at least once since the list was last reset.

Maxval:

The maximum value of the corresponding parameter number.

$t_{\max}$ :

The time at which the maximum value was attained.

Minval:

The minimum value of the corresponding parameter number.

$t_{\min}$ :

The time at which the minimum value was attained.

Meanval:

The mean value of the corresponding parameter number.

Stddevval:

The standard deviation of the corresponding parameter number, where the service supports the evaluation of this attribute.

This field shall be systematically omitted if the service does not support the evaluation of standard deviations.

### 9.3.2 Resetting the parameter statistics reporting

The request shall be:

#### Reset Parameter Statistics Reporting (4,3)

Telecommand packet, application data: None.

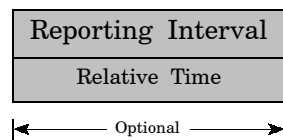
When this request is received, the evaluation of the parameter statistics shall be reset immediately, i.e. the current set of values is discarded and the evaluation shall start again from scratch.

### 9.3.3 Selecting the parameter statistics reporting mode

The requests to enable and disable the periodic reporting of the parameter statistics shall be:

#### Enable Periodic Parameter Statistics Reporting (4,4)

Telecommand packet, application data:



Reporting Interval:

The interval of time for the periodic reporting and resetting of the parameter statistics. This shall be greater than the sampling interval of any parameter currently in the parameter statistics list.

This field shall be systematically omitted if the service uses a default value for the reporting interval.

When this request is received, the service shall start the periodic reporting and resetting of the parameter statistics using the specified reporting interval (if any). When in periodic reporting mode, the ground system may still explicitly request a report at any time.

#### Disable Periodic Parameter Statistics Reporting (4,5)

Telecommand packet, application data: None.

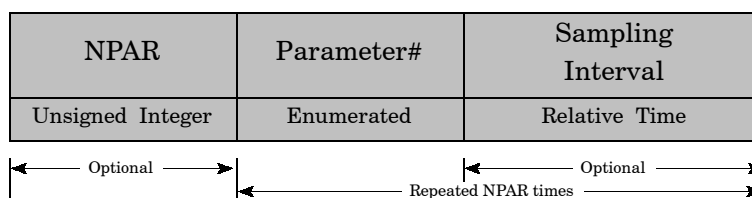
When this request is received, the periodic reporting and resetting of the parameter statistics shall be disabled and the parameter statistics evaluation shall continue.

### 9.3.4 Adding parameters to the parameter statistics list

The request shall be:

#### Add Parameters to Parameter Statistics List (4,6)

Telecommand packet, application data:



NPAR:

This field shall be systematically omitted if the service can only add one parameter at a time.

Sampling Interval:

The sampling interval to use for the associated parameter. If the parameter statistics reporting mode is currently periodic, the sampling interval shall be smaller than the reporting interval.

This field shall be systematically omitted if the service knows which value to use.

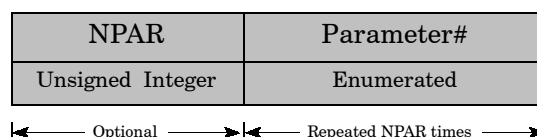
When this request is received, the indicated parameters shall be added to the statistics list and the evaluation of their statistics shall be started immediately. Within the next report (and only that report), parameters which were added to the list during the previous reporting interval shall be reported over a shorter interval than parameters which were already in the list.

### 9.3.5 Deleting parameters from the parameter statistics list

The request shall be:

#### Delete Parameters from Parameter Statistics List (4,7)

Telecommand packet, application data:



NPAR:

This field shall be systematically omitted if the service can only delete one parameter at a time.

When this request is received, the indicated parameters shall be removed from the list and the evaluation of their statistics shall be stopped immediately. These parameters shall not be reported in the succeeding report, even though their statistics have been evaluated over a part of the reporting interval.

### 9.3.6 Reporting the parameter statistics list

The request shall be:

#### Report Parameter Statistics List (4,8)


Telecommand packet, application data: None.

When this request is received, a report shall be generated as follows:

#### Parameter Statistics List Report (4,9)

Telemetry source packet, source data:

Reporting Interval	NPAR	Parameter#	Sampling Interval
Relative Time	Unsigned Integer	Enumerated	Relative Time



Reporting Interval:

The interval of time for the periodic reporting and resetting of the parameter statistics. If the current reporting mode is not periodic, its value shall be 0 s.

This field shall be systematically omitted if either:

- (a) the service does not support the concept of periodic reporting; or
- (b) the service uses a default value for the reporting interval.

### 9.3.7 Clearing the parameter statistics list

The request shall be:

#### Clear Parameter Statistics List (4,10)

Telecommand packet, application data: None.

When this request is received, the statistics list shall be cleared immediately.

## 9.4 Capability sets

### 9.4.1 Minimum capability set

The minimum capability set shall consist of the service requests and reports given in Table 14.

**Table 14: Summary of parameter statistics reporting service minimum capabilities**

Subtype	Service request, report or capability
1	Report Parameter Statistics
2	Parameter Statistics Report
3	Reset Parameter Statistics Reporting

### 9.4.2 Additional capability sets

The parameter statistics reporting service can also implement the additional capabilities as given in Table 15.

**Table 15: Summary of parameter statistics reporting service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
	Evaluation of standard deviation	"Stddevval" in subtype 2
4	Enable Periodic Parameter Statistics Reporting	5
5	Disable Periodic Parameter Statistics Reporting	4
6	Add Parameters to Parameter Statistics List	7 and/or 10
7	Delete Parameters from Parameter Statistics List	6
8	Report Parameter Statistics List	9
9	Parameter Statistics List Report	8
10	Clear Parameter Statistics List	6

---

## Event reporting service

### 10.1 Scope

This service provides for the reporting to the service user of information of operational significance which is not explicitly provided within the provider-initiated reports of another service. The service covers the requirements for event reporting, i.e.:

- a. reporting of failures or anomalies detected on-board;
- b. reporting of autonomous on-board actions;
- c. reporting of normal progress of operations and activities, e.g. detection of events which are not anomalous (such as payload events), reaching of predefined steps in an operation.

Some reports can combine more than one of these events.

EXAMPLE A report can declare that “Unit X has been switched off because its temperature was detected as 31 °C, where the currently defined limit is 30 °C”.

This service satisfies the concepts described in subclause 4.5.2 of this Standard.

Any number of on-board application processes may provide a single instance of the event reporting service.

Telemetry source packets and telecommand packets relating to the event reporting service shall be denoted by Service Type = 5.

### 10.2 Service concept

This service provides the capability for the generation of reports by any on-board function to notify the ground system of an event of operational significance. Four distinct levels of event report shall be provided:

- a. Normal/progress reports e.g. to notify the ground system of an on-board autonomous action, which does not relate to a fault condition.
- b. Error/anomaly report – low severity.
- c. Error/anomaly report – medium severity.
- d. Error/anomaly report – high severity.

In addition, the generation of specified event reports may be temporarily disabled, for example, to reduce the on-board processing load.

## 10.3 Service requests and reports

### 10.3.1 Reporting events

The different levels of provider-initiated event report shall be allocated different subtypes, to facilitate routing and ground processing. All reports shall have the same structure, as follows.

#### Normal/Progress Report (5,1)

Telemetry source packet, source data:

#### Error/Anomaly Report - Low Severity (5,2)

Telemetry source packet, source data:

#### Error/Anomaly Report - Medium Severity (5,3)

Telemetry source packet, source data:

#### Error/Anomaly Report - High Severity (5,4)

Telemetry source packet, source data:

RID	Parameters
Enumerated	Any

|← Optional →|

RID:

The Report ID (RID), together with the application process ID, implicitly defines the presence, structure and interpretation of the associated parameters field.

Parameters:

The parameters field shall provide complementary information relating to the particular value of the report ID. For these four subtypes, the parameters field shall conform to structure rules set #1 and each parameter shall be one of the standard data types (see clause 23).

### 10.3.2 Controlling the generation of event reports

The requests to enable and disable the generation of one or more event reports shall be:

#### Enable Event Report Generation (5,5)

Telecommand packet, application data:

#### Disable Event Report Generation (5,6)

Telecommand packet, application data:

NRID	RID
Unsigned Integer	Enumerated

|← Optional →|← Repeated NRID times →|

NRID:

The number of RIDs that follow.

This field shall be systematically omitted if the service can only enable or disable the generation of one event report definition at a time (i.e. NRID=1).



RID:

The identifier(s) of the event report(s) to be enabled or disabled.

When this request is received, the activities for the generation of the indicated event report(s) shall be started or stopped.

## 10.4 Capability sets

### 10.4.1 Minimum capability set

The minimum capability set shall consist of the service reports given in Table 16.

**Table 16: Summary of event reporting service minimum capabilities**

Subtype	Service request, report or capability
1	Normal/Progress Report
2	Error/Anomaly Report - Low Severity
3	Error/Anomaly Report - Medium Severity
4	Error/Anomaly Report - High Severity

### 10.4.2 Additional capability sets

The event reporting service may also implement the additional capabilities given in Table 17.

**Table 17: Summary of event reporting service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
5	Enable Event Report Generation	6
6	Disable Event Report Generation	5

*(This page is intentionally left blank)*

---

## Memory management service

### 11.1 Scope

This service relates to the management of the various memory areas (e.g. RAM or mass memory unit) which exist on-board the satellite. The service provides the capability for loading, dumping and checking the contents of either a contiguous memory area or of several non-contiguous memory areas.

This service satisfies the concepts described in subclause 4.13 of this Standard.

Any number of on-board application processes may provide a single instance of the memory management service; however, the number of instances shall ensure that all on-board changeable memory areas can be loaded and that all on-board memory areas can be dumped.

Telemetry source packets and telecommand packets relating to the memory management service shall be denoted by Service Type = 6.

### 11.2 Service concept

The memory management service provides basic dump, load and check facilities without the maintenance of service state information. It shall only read from, write to and checksum areas of, the memory blocks. The service need not know which memory areas are changeable, what the sizes of the memory blocks are or what types of information reside in particular parts of the memory blocks.

Optionally, the service shall confirm the integrity of the data to be loaded by checksumming the data and comparing the result against the checksum that is uplinked in the packet. Only if the two values correspond shall the data be loaded into memory.

A “Memory ID” shall uniquely identify each on-board memory block. It denotes, for example, a physical on-board memory of an on-board processor or the virtual memory of an application process (which can be mapped onto physical on-board memories and mass memory units).

Two distinct addressing techniques shall be available for memory load, dump and check requests and reports, namely: a base reference plus an offset or absolute addressing. In general, it is expected that a given memory block uses only one of these addressing schemes, however it is not excluded that different schemes may be used by different application processes that access the same memory block.

The memory management service can maintain several “symbolic base references” for a given memory block. A symbolic base reference denotes a well-defined

address inside the memory block which can be used in memory load, dump and check requests and reports.

The memory management service may make use of the large data transfer service (see clause 16) when the transaction for dumping (or loading) memory areas exceeds the maximum size of a single telemetry source packet (or telecommand packet) defined for the mission.

Different on-board processors may have different addressing capabilities. Some may not be capable of addressing a single octet, but instead may have a <SMALLEST\_ADDRESSABLE\_UNIT> (SAU) which is a 16-, 24-, 32- or 64-bit word. The base for the memory management service is thus referred to this SAU, whose actual value is implementation-dependent.

## 11.3 Service requests and reports

### 11.3.1 Loading data in memory using base plus offsets

The request to load data into one or more areas of a memory block defined by means of a base reference plus offsets shall be:

#### Load Memory using Base plus Offsets (6,1)

Telecommand packet, application data:

Memory ID	Base	N	Offset	Length	Data	Checksum
Fixed OctetString	Deduced	Unsigned Integer	Signed Integer	Variable OctetString		Fixed BitString (16 bits)

← Optional →      ← Optional →      ← Repeated N times →      ← Optional →

Memory ID:

This identifies the destination memory block (it can be interpreted as a character string if it always consists of ASCII codes). The meaning and internal structure of the Memory ID are beyond the scope of this Standard.

This field shall be systematically omitted if the service only manages one on-board memory block.

Base:

This is a base reference which gives (explicitly or implicitly) the address (in SAUs, with the count starting from zero) within the memory block which shall be used as the zero reference for the offset addresses.

The type of the Base field shall be one of the following:

- (a) an unsigned integer, or
- (b) a symbolic base reference which is a fixed length character string.

The type and length can be deduced from the ID of the memory to be loaded, from the APID of the destination application process of the request or from the combination of the two (see also clause 23).

N:

The number of data blocks to be loaded.

This field shall be systematically omitted if the service can only load a single continuous area of memory (i.e. N=1).

Offset:

This specifies the offset (in SAUs, with the count starting from zero) from the base reference of the start address for loading the data which follows.

Length:

The number of SAUs to be loaded.

**Data:**

A data block to be loaded (in increasing order of SAU).

**Checksum:**

An ISO standard 16-bit checksum (see annex A.2) that is used by the service provider to verify the integrity of the data being loaded. The checksum is calculated across the data words only.

This field shall be systematically omitted if checksumming at data level is not supported.

When the service provider receives this request:

- a. If a checksum is provided with the memory load, it shall calculate the checksum of the data being loaded and compare the result with the value in the checksum field. In the case of a scatter-load, this shall be repeated for each data block.

NOTE The service can be designed to skip steps b. and c. (below) if the checksum comparison fails.

- b. It shall write each data block into the memory, at the specified offset from the base reference.
- c. If the completion of execution of the request is reported, the memory area just written to shall be re-read, a checksum shall be calculated and compared with the value uplinked in the checksum field. In the case of a scatter-load, this shall be repeated for each data block. The results shall be reported.


### 11.3.2 Loading data in memory using absolute addresses

The request to load data to one or more areas of a memory block defined using absolute addresses shall be:

#### Load Memory using Absolute Addresses (6,2)

Telecommand packet, application data:

Memory ID	N	Start Address	Length	Data	Checksum
Fixed OctetString	Unsigned Integer	Unsigned Integer	Variable OctetString		Fixed BitString (16 bits)



**Memory ID:**

This field shall be systematically omitted if the service only manages one on-board memory block.

**N:**

This field shall be systematically omitted if the service can only load a single continuous area of memory (i.e. N=1).

**Start Address:**

This gives the start address (in SAUs, with the count starting from zero) within the memory block for loading the data.

The length of this field can be deduced in a similar way to the "Base" field of subtype 1 above.

When the service provider receives this request, it shall perform steps a, b and c as described for subtype 1 above.

### 11.3.3 Dumping memory using base plus offsets

The request to dump the contents of one or more areas of a memory block defined using a base reference plus offsets shall be:

#### Dump Memory using Base plus Offsets (6,3)

Telecommand packet, application data:

Memory ID	Base	N	Offset	Length
Fixed OctetString	Deduced	Unsigned Integer	Signed Integer	Unsigned Integer

← Optional → | ← Optional → | ← Repeated N times →

Memory ID:

This field shall be systematically omitted if the service only manages one on-board memory block.

N:

This field shall be systematically omitted if the service can only dump a single continuous area of memory (i.e. N=1).

Length:

The number of SAUs to be dumped.

When the service provider receives this request it shall read each indicated area of the memory block, generate a report containing the contents of these areas and downlink it (e.g. using the large data transfer service).

#### Memory Dump using Base plus Offsets Report (6,4)

Telemetry source packet, source data:

Memory ID	Base	N	Offset	Length	Data	Checksum
Fixed OctetString	Deduced	Unsigned Integer	Signed Integer	Variable	OctetString	Fixed BitString (16 bits)

← Optional → | ← Optional → | ← Repeated N times → | ← Optional →

Memory ID:

This field shall be systematically omitted if the service only manages one on-board memory block.

N:

This field shall be systematically omitted if the service can only dump a single continuous area of memory (i.e. N=1).

Length:

The number of SAUs of data that follow.

Data:

These are the values of the memory locations being dumped, starting from the SAU position indicated by the initial request (in increasing order of SAU).

Checksum:

The service shall calculate an ISO standard 16-bit checksum for each data block being dumped and place the result in this field.

This field shall be systematically omitted if checksumming at data level is not supported.

### 11.3.4 Dumping memory using absolute addresses

The request to dump the contents of one or more areas of a memory block defined using absolute addresses shall be:

#### Dump Memory using Absolute Addresses (6,5)

Telecommand packet, application data:

Memory ID	N	Start Address	Length
Fixed OctetString	Unsigned Integer	Unsigned Integer	Unsigned Integer

← Optional → | ← Optional → | ← Repeated N times →

Memory ID:

This field shall be systematically omitted if the service only manages one on-board memory block.

N:

This field shall be systematically omitted if the service can only dump a single continuous area of memory (i.e. N=1).

When the service provider receives this request it shall read each indicated area of the memory block, generate a report containing the contents of these areas and downlink it (e.g. using the large data transfer service).

#### Memory Dump using Absolute Addresses Report (6,6)

Telemetry source packet, source data:

Memory ID	N	Start Address	Length	Data	Checksum
Fixed OctetString	Unsigned Integer	Unsigned Integer	Variable OctetString		Fixed BitString (16 bits)

← Optional → | ← Optional → | ← Repeated N times → | ← Optional →

Memory ID:

This field shall be systematically omitted if the service only manages one on-board memory block.

N:

This field shall be systematically omitted if the service can only dump a single continuous area of memory (i.e. N=1).

Checksum:

The service shall calculate an ISO standard 16-bit checksum for each data block being dumped and place the result in this field.

This field shall be systematically omitted if checksumming at data level is not supported.

### 11.3.5 Checking memory using base plus offsets

The request to check the contents of one or more areas of a memory block defined using a base reference plus offsets shall be:

#### Check Memory using Base plus Offsets (6,7)

Telecommand packet, application data:

Memory ID	Base	N	Offset	Length
Fixed OctetString	Deduced	Unsigned Integer	Signed Integer	Unsigned Integer

← Optional → |      ← Optional → |      Repeated N times →

**Memory ID:**

This field shall be systematically omitted if the service only manages one on-board memory block.

**N:**

This field shall be systematically omitted if the service can only check a single continuous area of memory (i.e. N=1).

When the service provider receives this request it shall read and compute the checksum value of each indicated area of the memory block using the ISO standard 16-bit checksum algorithm defined in annex A.2. It shall then generate a report containing the checksum values computed.

**Memory Check using Base plus Offsets Report (6,8)**

Telemetry source packet, source data:

Memory ID	Base	N	Offset	Length	Checksum
Fixed OctetString	Deduced	Unsigned Integer	Signed Integer	Unsigned Integer	Fixed BitString (16 bits)

← Optional → |      ← Optional → |      Repeated N times →

**Memory ID:**

This field shall be systematically omitted if the service only manages one on-board memory block.

**N:**

This field shall be systematically omitted if the service can only check a single continuous area of memory (i.e. N=1).

**Checksum:**

The value obtained by computing the ISO checksum over the relevant memory locations.

**11.3.6 Checking memory using absolute addresses**

The request to check the contents of one or more areas of a memory block defined with absolute addresses shall be:

**Check Memory using Absolute Addresses (6,9)**

Telecommand packet, application data:

Memory ID	N	Start Address	Length
Fixed OctetString	Unsigned Integer	Unsigned Integer	Unsigned Integer

← Optional → |      ← Optional → |      Repeated N times →

**Memory ID:**

This field shall be systematically omitted if the service only manages one on-board memory block.



N:

This field shall be systematically omitted if the service can only check a single continuous area of memory (i.e. N=1).

When the service provider receives this request it shall read and compute the checksum value of each indicated area of the memory block using the ISO standard 16-bit checksum algorithm defined in annex A.2. It shall then generate a report containing the checksum values computed.

#### Memory Check using Absolute Addresses Report (6,10)

Telemetry source packet, source data:

Memory ID	N	Start Address	Length	Checksum
Fixed OctetString	Unsigned Integer	Unsigned Integer	Unsigned Integer	Fixed BitString (16 bits)

← Optional → ← Optional → ← Repeated N times →

Memory ID:

This field shall be systematically omitted if the service only manages one on-board memory block.

N:

This field shall be systematically omitted if the service can only check a single continuous area of memory (i.e. N=1).

## 11.4 Capability sets

### 11.4.1 Minimum capability set

The service shall provide the capability to load and to dump memory areas using the addressing technique of the memory block(s) it manages. This implies that one of the “Load and Dump Memory using Base plus Offsets” or “Load and Dump Memory using Absolute Addresses” sub-services shall be provided, i.e.:

- Load Memory (6,1), Dump Memory (6,3) and Memory Dump (6,4); or
- Load Memory (6,2), Dump Memory (6,5) and Memory Dump (6,6).

### 11.4.2 Additional capability sets

In addition to a minimum sub-service, the additional capabilities given in Table 18 can be implemented.

**Table 18: Summary of memory management service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
1	Load Memory using Base plus Offsets	3 and 4
2	Load Memory using Absolute Addresses	5 and 6
3	Dump Memory using Base plus Offsets	1 and 4
4	Memory Dump using Base plus Offsets Report	3
5	Dump Memory using Absolute Addresses	2 and 6
6	Memory Dump using Absolute Addresses Report	5
7	Check Memory using Base plus Offsets	8
8	Memory Check using Base plus Offsets Report	7
9	Check Memory using Absolute Addresses	10
10	Memory Check using Absolute Addresses Report	9
	Support of symbolic base references	

---

# 12

---

## Function management service

### 12.1 Scope

An application process may be designed to support software functions that are not implemented as standard or mission-specific services but whose execution may nevertheless be controlled from the ground. Examples of such functions include control of the operation of a payload instrument or a platform subsystem.

The function management service provides a standard service request for performing the functions of the application process. This service satisfies the software management concept described in subclause 4.6 of this Standard.

Any number of on-board application processes may provide a single instance of the function management service.

Telemetry source packets and telecommand packets relating to the function management service shall be denoted by Service Type = 8.

### 12.2 Service concept

Each function within an application process shall be uniquely identified by a "Function ID". The Function ID shall be assigned at the level of the individual controls for the corresponding function, e.g. load manoeuvre parameters, start manoeuvre, stop manoeuvre, configure instrument to mode A, switch off instrument etc. Where appropriate, a set of parameters pertaining to the given function may also be provided.

### 12.3 Service requests and reports

#### 12.3.1 General

There are currently no service reports defined.

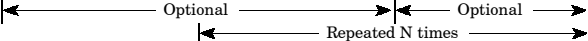
### 12.3.2 Perform function

The request shall be:

#### Perform Function (8,1)

Telecommand packet, application data:

Function ID	N	Parameter#	Value
Fixed CharString	Unsigned Integer	Enumerated	Deduced



Function ID:

The Function ID, together with the APID in the packet header, implicitly defines the presence of, and the structure of, the fields which follow. The length of the character string may be defined on a mission basis or per application process.

The internal structure of the Function ID is beyond the scope of this Standard.

N:

The number of (Parameter#, Value) pairs which follow.

Parameter#:

The identification of the parameter whose value follows.

Value:

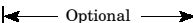
The value of the corresponding parameter (its type shall be deduced from the Parameter#).

The parameters shall be used to configure the specific instance of execution of the application function. Two options are available for supplying parameters (the option is at Function ID-level, rather than at packet instance):

- Only a subset of parameters is loaded. In this case, the packet shall contain an array of the corresponding parameter numbers and values. Any parameters not loaded shall retain their current values.
- The full set of parameters is loaded. In this case, the packet shall contain a data structure (of parameter values) conforming to structure rules set #1 (see subclause 23.6.2), where the structure is known implicitly.

In the latter case, the packet structure becomes simply:

Function ID	Parameters
Fixed CharString	Any



When this request is received, the specified function shall be performed using (or passing) the specified parameters.

## 12.4 Capability sets

### 12.4.1 Minimum capability set

The minimum capability set shall consist of the service request given in Table 19.

**Table 19: Summary of function management service  
minimum capabilities**

Subtype	Service request, report or capability
1	Perform Function

### 12.4.2 Additional capability sets

There are no additional capability sets.

*(This page is intentionally left blank)*

---

# 13

---

## Time management service

### 13.1 Scope

All satellites maintain a satellite time reference which can be downlinked by means of the “Time Report” defined in this clause. The ground segment can perform a correlation between satellite time and UTC (universal time coordinated) using, for example, the procedures described in annex C.

The time management service provides the (optional) capability for the control of the rate of generation of the time reports. This service option shall be used when a mission has varying requirements for time correlation accuracy.

This service satisfies the concepts described in subclause 4.5.4 of this Standard.

The two sub-services (rate control, time reporting) may be provided by different application processes, however there shall be, at most, a single instance of each. The time reporting sub-service shall be provided by application process ID = 0 (see annex C).

Telemetry source packets and telecommand packets relating to the time management service shall be denoted by Service Type = 9.

### 13.2 Service concept

The rate control sub-service shall maintain the generation rate (1, 2, 4, 8, 16, 32, 64, 128 or 256) of the time report. It shall also have means of communicating this generation rate to the time reporting sub-service.

The time reporting sub-service shall have access to the satellite time reference which is a free running counter from a given epoch (i.e. absolute time). The satellite time reference shall be sampled at the instant of occurrence of the leading edge of the first bit of the attached synchronization marker of the telemetry transfer frame of Virtual Channel 0 that has a virtual channel frame count of “0” and for each subsequent frame for which:

$$\text{virtual channel frame count modulo (generation rate)} = 0.$$

The time reporting sub-service shall then downlink this satellite time reference in a spacecraft time source packet at any time before the satellite time reference is next sampled.

When a new generation rate is requested, the time reporting sub-service shall use this new generation rate from the next telemetry transfer frame that meets the above criterion.

## 13.3 Service requests and reports

### 13.3.1 Controlling the time report generation rate

The request for changing the rate of generation of the time report shall be:

#### Change Time Report Generation Rate (9,1)

Telecommand packet, application data:

Rate
Unsigned Integer (1 octet)

Rate:

This parameter determines the generation rate used to sample and down-link the satellite time. Its value shall be in the range 0 to 8 inclusive. The corresponding generation rate is equal to once every  $2^{\text{Rate}}$  telemetry transfer frames.

When this request is received, the rate control sub-service shall inform the time reporting sub-service that a new time report generation rate shall be used.

### 13.3.2 Time reporting

The time report is the only provider-initiated service report defined for the time reporting sub-service.

As described in annex C, the spacecraft time source packet containing the time report has no data field header.

#### Time Report (9,2)

Telemetry source packet, source data:

Rate	Satellite Time	Status
Unsigned Integer (1 octet)	Absolute Time	Deduced
← Optional →		← Optional →

Rate:

The rate field shall be systematically omitted if a mission has a fixed generation rate.

Satellite Time:

This is the satellite time reference sampled according to the procedure described above. The format of this parameter shall be the CCSDS Unsegmented Code (CUC) format with implicit or explicit P-Field (see CCSDS 301.0-B-2). If the P-field is implicit, its value shall be given by the mission constant <MISSION\_TIME\_CODE>.

Status:

This shall give the status of the time reporting sub-service. The type and values of the Status parameter are not specified in this Standard. The type and physical format are deduced from other information (see clause 23).

The Status field may be systematically omitted if a mission implements a basic time reporting sub-service.



## 13.4 Capability sets

### 13.4.1 Minimum capability sets

The rate control sub-service is an optional capability, but a pre-requisite is that the time reporting sub-service is also provided. The minimum capability set for the rate control sub-service shall consist of the service request given in Table 20.

**Table 20: Summary of rate control sub-service minimum capabilities**

Subtype	Service request, report or capability
1	Change Time Report Generation Rate

As noted earlier, the time reporting sub-service may only be provided by APID = 0. The minimum capability set for the time reporting sub-service shall consist of the service report given in Table 21.

**Table 21: Summary of time reporting sub-service minimum capabilities**

Subtype	Service request, report or capability
2	Time Report

NOTE Transmission of the spacecraft time source packet to the ground is enabled and disabled using the packet forwarding control service (see clause 17).

### 13.4.2 Additional capability sets

There are no additional capability sets.

*(This page is intentionally left blank)*

---

## On-board operations scheduling service

### 14.1 Scope

The on-board operations scheduling service provides the capability to command on-board application processes using telecommands pre-loaded on-board the satellite and released at their due time. To achieve this, the service maintains an on-board command schedule and ensures the timely execution of telecommands contained therein.

This service satisfies the concepts described in subclause 4.7 of this Standard.

Any number of on-board application processes may provide a single instance of the on-board operations scheduling service.

Telemetry source packets and telecommand packets relating to the on-board operations scheduling service shall be denoted by Service Type = 11.

### 14.2 Service concept

#### 14.2.1 General

The on-board operations scheduling service shall maintain a command schedule which contains telecommand packets and their associated scheduling information.

The service user(s) can request the following activities:

- a. Enable the scheduling of all, or a subset of, the telecommands in the command schedule (e.g. those to be sent to specified application processes).
- b. Disable the scheduling of all, or a subset of, the telecommands in the command schedule.
- c. Add telecommands to the command schedule.
- d. Delete all, or a subset of, the telecommands in the command schedule (e.g. the telecommands becoming due for release within a specified time period).
- e. Time shift all, or a subset of, the telecommands in the command schedule.
- f. Report on all, or a subset of, the telecommands in the command schedule.
- g. Report the status of the command schedule.

Inconsistencies shall not exist as a direct result of the processing of a service user request. The on-board operations scheduling service shall refuse to perform a request in its entirety if this creates an inconsistency in the command schedule.

The service user(s) shall be responsible for ensuring that inconsistencies which can only be detected during the scheduling activity shall never occur.

### 14.2.2 The command schedule

The on-board operations scheduling service maintains a command schedule consisting of telecommand packets together with their scheduling attributes. The scheduling attributes of a telecommand indicate the following:

- a. The sub-schedule with which the telecommand is associated. A sub-schedule is a grouping mechanism for telecommands that enables them to be controlled together or interlocked to others in the same group (see point b. below).
- b. The number of the interlock to be set by this telecommand, if any. The success or failure of execution of the telecommand is associated with the specified interlock. Interlocking is restricted to commands belonging to the same sub-schedule.
- c. The number of the interlock on which the release of this telecommand is dependent, if any.
- d. Whether the release of this telecommand is dependent on the success or on the failure of the telecommand with which it is interlocked.
- e. Either the absolute on-board time at which the telecommand packet is released to its destination application process or a relative time (if supported).

Absolute times and relative times shall be expressed in CUC format for the on-board operations scheduling service.

Relative time shall be defined with respect to a “Scheduling Event” which can be the starting of the command schedule, the starting of the sub-schedule containing the telecommand or the setting of the interlock with which the telecommand is interlocked (i.e. when the service is notified of the success or failure of the telecommand which sets the interlock).

The scheduling event for relative time can also be the occurrence of a mission-specific event.

The on-board operations scheduling service shall know, therefore, if an interlocking telecommand defines a scheduling event for other subsequent interlocked telecommands. In this case the actual time of completion of command execution shall be used for the derivation of the release times of the subsequent telecommands.

### 14.2.3 Telecommand release status

The on-board operations scheduling service shall maintain appropriate information to determine whether a telecommand should be released or not at its due time.

The release status of a telecommand shall be affected by the user requests to enable or disable the release of all or a subset of the telecommands in the command schedule. The telecommand release status shall be either “disabled” or “enabled”.

The release status of a telecommand shall be “enabled” if the release of telecommands is enabled from the command schedule, from the sub-schedule to which the telecommand belongs, and to the destination application process of the telecommand.

The release status shall be “disabled” in all other cases.

Conceptually, this is as if each telecommand has three independent controlling attributes (at schedule level, at sub-schedule level and at destination application process level) whose values determine the release status of the telecommand in accordance with Table 22.

**Table 22: Decision table for the release status of a telecommand**

<b>Schedule</b>	<b>Sub-schedule</b>	<b>Destination application process</b>	<b>Release status</b>
D(isabled)	E(nabled)	E	D
D	D	E	D
D	E	D	D
D	D	D	D
E	E	E	E
E	D	E	D
E	E	D	D
E	D	D	D

#### 14.2.4 Telecommand interlock status

The execution result of a telecommand which sets an interlock shall be used in order to determine the interlock status of the telecommands which are due for release.

The execution result of a telecommand can be “success” or “failure”.

When a telecommand which sets an interlock is due for release, then:

- if its release status is “disabled” or its interlock status is “locked”, the telecommand shall not be released and its execution result shall be set to “failure”;
- otherwise the telecommand shall be released and its execution result are unknown until the execution of the telecommand is completed.

The on-board operations scheduling service shall explicitly indicate to the destination application process that the released telecommand sets an interlock so that, when the telecommand execution is complete, an execution report shall be passed back to the service.

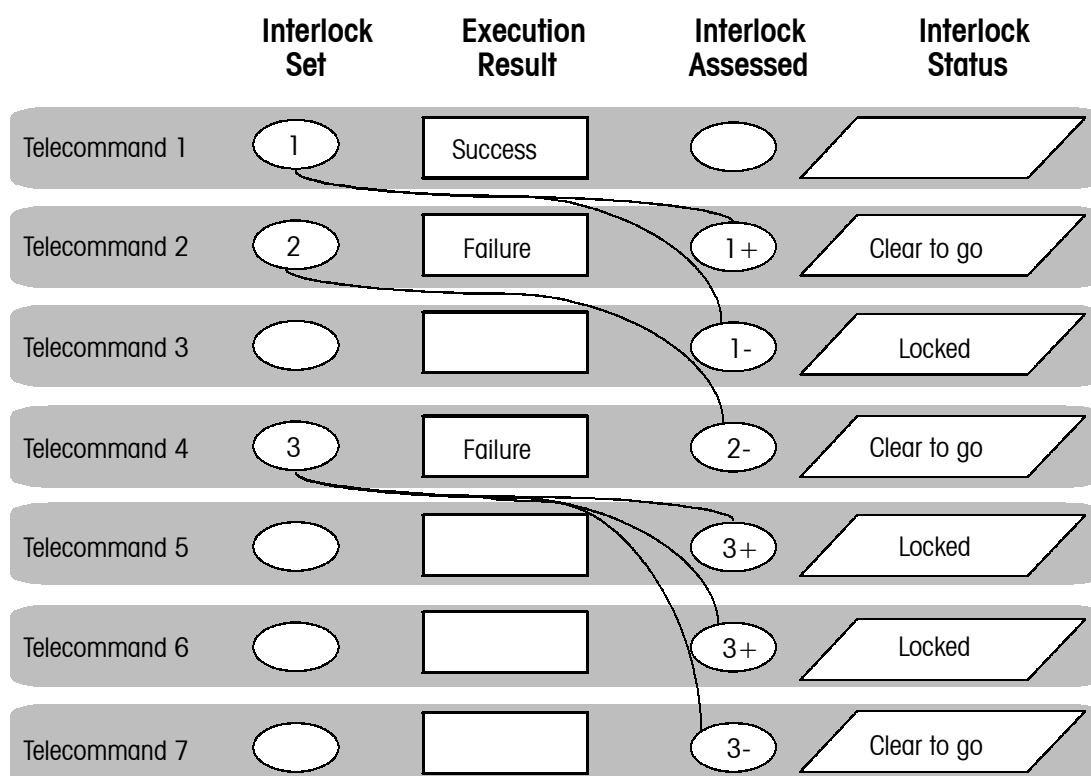
On reception of this report, the service shall set the telecommand execution result accordingly (“success” or “failure”).

If the execution completion report is not received after the maximum execution duration of the telecommand has elapsed, then the telecommand execution result shall be set to “failure”.

When a telecommand is due for release and its release depends on an interlock, its interlock status is evaluated as follows:

- If the release of the telecommand depends on the success of the telecommand setting the interlock and the telecommand execution result attached to the interlock is “failure”, then it shall be “locked”.
- If the release of the telecommand depends on the failure of the telecommand setting the interlock and the telecommand execution result attached to the interlock is “success”, then it shall be “locked”.

This is illustrated in Figure 5 in which “1+” means “release depends on the successful execution of the telecommand setting interlock number 1” and “1-” means the opposite.



**Figure 5: The relation between execution result and interlock status**

#### 14.2.5 Scheduling events

The on-board operations scheduling service shall determine the times of occurrence of the various scheduling events used as the basis for relative release times, namely:

- The on-board CUC time at which the command schedule was enabled.
- The on-board CUC time at which a sub-schedule was enabled, for each sub-schedule which contains telecommands whose release time is relative to the sub-schedule enable time.
- The on-board CUC time of reception of the execution completion report (or of occurrence of the execution completion timeout), for each interlocking telecommand whose execution completion is a scheduling event for subsequent telecommands.

**NOTE** As soon as a scheduling event has occurred, all telecommands whose scheduled time is relative to that scheduling event have a known absolute schedule time. If the same scheduling event recurs, this does not affect the absolute times of these telecommands.

The on-board operations scheduling service shall refuse to add or time-shift telecommands if this results in an interlock-dependent telecommand appearing before the execution completion timeout defined for its interlocking telecommand. This ensures that the execution result of the interlocking telecommand is known when the release of the interlock-dependent telecommand is due.

The service user(s) should also ensure that this situation never occurs in the case where it cannot be detected at the time of processing of a service request (e.g. by determining the maximum execution duration of all telecommand paths leading to an interlock-dependent telecommand with absolute time).

#### **14.2.6 Auxiliary information**

The on-board operations scheduling service shall also have access to other information needed for the proper execution of its activities. This includes:

- a. The maximum number of entries or maximum size of the command schedule.
- b. The maximum number of sub-schedules which can be simultaneously managed.
- c. The maximum number of interlocks which can be simultaneously managed.
- d. The list of sources from which the service can receive telecommand packets to be scheduled.
- e. The list of on-board application processes to which the service can release telecommand packets.

The service shall use this information for error detection and reporting.

#### **14.2.7 The scheduling activity**

The processing of a telecommand packet whose release time is due shall always be performed (even if the command schedule is disabled).

The corresponding service activity shall be:

- a. The telecommand shall not be released if the telecommand release status is “disabled” or the telecommand interlock status is “locked”. If the telecommand defines a scheduling event, the scheduling event time shall be the current time. If the telecommand sets an interlock, the execution result shall be set to “failure”.
- b. Otherwise, the telecommand shall be released. Where applicable, its execution result shall then be determined as indicated in subclause 14.2.4 and the time of its scheduling event shall be determined as indicated in subclause 14.2.5.

In addition to this primary activity, any on-board CUC time jumps shall be detected and reported.

On detection of a time jump, the service shall suspend its execution as soon as it has processed (including releasing) all the telecommands which are interlocked with telecommands which have already been released (if any).

### **14.3 Service requests and reports**

#### **14.3.1 Controlling the command schedule**

##### **14.3.1.1 General**

There are several ways of enabling or disabling the release of a subset of the telecommands in the command schedule. The subsets are specified according to selection criteria.

EXAMPLE Telecommands with specified destination application processes.

If an error is detected during the processing of a request, it shall not affect the processing of the remainder of the request.

### 14.3.1.2 Controlling the release of telecommands

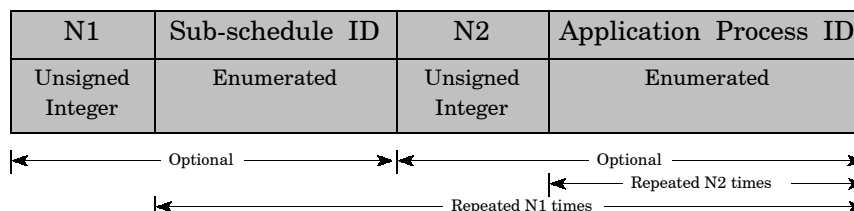
The service requests to enable or disable the release of selected telecommands shall be:

#### Enable Release of Telecommands (11,1)

Telecommand packet, application data:

#### Disable Release of Telecommands (11,2)

Telecommand packet, application data:



N1:

This field shall be systematically omitted if the service does not support the concept of sub-schedules (which is equivalent to having N1=1). In this case, the Sub-schedule ID field shall also be omitted and the application data is simply an array of N2 APIDs.

Sub-schedule ID:

The identification of the sub-schedule(s) to be enabled or disabled. By convention, the value 0 for Sub-schedule ID shall mean "all sub-schedules".

This field shall be systematically omitted if the service does not support the concept of sub-schedules.

N2:

This field shall be systematically omitted if the service does not support selection at application process level (which is equivalent to having N2=0).

Application Process ID:

The identification of the destination application process(es) to be enabled or disabled.

When the service provider receives this request, then:

- If N1 = 0, the schedule level controlling attribute of all telecommands (see subclause 14.2.3) shall be set according to the request type. If it is an enable request, then the command schedule scheduling event time shall also be set.
- If N1 > 0 and N2 = 0, the sub-schedule level controlling attribute of the telecommands from the specified sub-schedules shall be set according to the request type. If it is an enable request, then the scheduling event times of the sub-schedules shall also be set.
- If N1 > 0 and N2 > 0, the application process level controlling attribute of the telecommands with the specified destination application processes and with the specified sub-schedules shall be set according to the request type.

NOTE If N1 > 1 then there can be a mixture of empty arrays (N2 = 0) and non-empty arrays (N2 > 0).



### 14.3.1.3 Resetting the command schedule

The request shall be:

#### Reset Command Schedule (11,3)

Telecommand packet, application data: None.

When the service provider receives this request, it shall reset the command schedule as follows (this is considered to be the “default” starting state):

- It shall clear all entries in the command schedule. The command schedule shall be disabled, all sub-schedules shall be disabled and commanding to all application processes shall be enabled.
- It shall reset the interlock information (no interlock defined).
- It shall reset the scheduling event information (all scheduling event times shall be unknown).

### 14.3.2 Inserting telecommands in the command schedule

The request to insert (e.g. add) one or more telecommands in the Command Schedule shall be:

#### Insert Telecommands in Command Schedule (11,4)

Telecommand packet, application data:

Sub-schedule ID	N
Enumerated	Unsigned Integer

← Optional → | ← Optional → |

Interlock Set ID	Interlock Assessed ID	Assessment Type	Scheduling Event	Abs/Rel Time Tag	Execution Timeout	Telecommand Packet
Enumerated	Enumerated	Enumerated	Enumerated	Absolute or Relative Time	Relative Time	Any TC

← Optional → | ← Optional → | ← Optional → | ← Optional → |  
Repeated N times

Sub-schedule ID:

The identification of the sub-schedule with which the following telecommands are associated. The Sub-schedule ID cannot take the value 0.

This field shall be systematically omitted if the service does not support the concept of sub-schedules.

N:

The number of telecommands to be inserted in the schedule.

This field shall be systematically omitted if the service only supports the insertion of a single telecommand at a time.

Interlock Set ID:

The identification of the interlock to be set by this telecommand (0 if no interlock is set). The status of this interlock shall be determined by the success or failure of the telecommand execution.

This field shall be systematically omitted if the service does not support the concept of interlocking.

Interlock Assessed ID:

The identification of the interlock on which the release of this telecommand is dependent (0 if no interlock is assessed).

This field shall be systematically omitted if the service does not support the concept of interlocking.

#### Assessment Type:

Whether the release of this telecommand is dependent on the success or failure of the telecommand with which it is interlocked, viz.:

“Success” (value = 1): release if interlocking telecommand was successful;

“Failure” (value = 0): release if interlocking telecommand failed execution.

This parameter shall not be present if the telecommand is not interlock dependent (or if the service does not support the concept of interlocking).

#### Scheduling Event:

This determines whether the release time of this telecommand shall be an absolute on-board CUC time (Scheduling Event = “Absolute”, value = 0) or a relative time and, in the latter case, this parameter indicates the type of scheduling event for the relative time.

If the Scheduling Event = “Schedule” (value = 1), the telecommand release time shall be relative to the time at which the command schedule is enabled.

If the Scheduling Event = “Sub-Schedule” (value = 2), the telecommand release time shall be relative to the time at which the sub-schedule containing the telecommand is enabled.

If the Scheduling Event = “Interlock” (value = 3), then the telecommand release time shall be relative to the time of notification to the service of the success or failure of the telecommand which sets the interlock.

The Scheduling Event can take other mission-specific values.

This field shall be systematically omitted if the service does not support the concept of relative time.

#### Abs/Rel Time Tag:

If Scheduling Event = “Absolute”, then this is the on-board CUC time at which the telecommand packet shall be sent to its destination application process. The format and length of this field shall be uniquely defined for the on-board application process which provides the service and shall be used whenever an absolute time tag is provided as a parameter of a service request or report.

If the Scheduling Event indicates a relative time, then this shall be an on-board positive delta time which, when added to the time of occurrence of the event identified by the Scheduling Event, shall determine the absolute time at which the telecommand packet shall be sent to its destination application process. The format and length shall be the same as for the absolute time tag and shall be uniquely defined for the on-board application process which provides the service.

#### Execution Timeout:

This shall be an on-board positive delta time which, when added to the time of release of the telecommand, shall determine the latest time at which the execution of the telecommand is expected to complete. This parameter shall only be present if the telecommand sets an interlock.

If an execution completion report is not received by the service within the timeout window, then the telecommand shall be deemed to have failed for the purposes of handling the interlock. Also, if the completion of execution of the telecommand constitutes a scheduling event, then the scheduling event time shall be set to the timeout window upper bound. The absolute times of the telecommands linked to this scheduling event shall then become known.

The format and length of this field shall be the same as for the absolute time tag defined for the on-board application process which provides the service.

#### Telecommand Packet:

This is a standard telecommand packet of any service type and subtype, (see subclause 5.3 for the structure of a telecommand). This can be a telecommand destined for an on-board operations scheduling service, either this one or another instance of the same service implemented in another application process (e.g. a request to enable or disable sub-schedules within the command schedule).

The source of the telecommand packet shall be indicated in the Source ID field (when present in the data field header). This can be the ground system or another on-board application process.

If a telecommand to be added has a relative time with respect to an interlock, it shall be “linked” to the latest telecommand added to the command schedule which sets that interlock.

When this request is received, each telecommand in the request shall be processed in turn and, if no error is detected during its processing, it shall be added to the command schedule.

An error shall occur, if:

- a. the command schedule is full;
- b. the Sub-schedule ID or one of the Interlock IDs exceeds its maximum value;
- c. the Interlock IDs are equal;
- d. the destination application process of the telecommand is invalid;
- e. the time specification refers to the past;
- f. the time specification is not supported by the service;
- g. the telecommand is interlock dependent and its release time falls within the execution window of its interlocking telecommand;
- h. the telecommand has an “Interlock” relative time and no telecommand added since the last resetting of the command schedule sets the interlock.

### 14.3.3 Deleting telecommands from the command schedule

#### 14.3.3.1 General

There are several ways of deleting a subset of the telecommands from the command schedule. The subsets are specified according to selection criteria.

EXAMPLE Telecommands whose absolute schedule time falls within a specified time period.

The scheduling service shall refuse to delete an interlocking telecommand unless all its (directly and indirectly) interlocked telecommands have either already been deleted or are deleted in the same deletion request.

A service user can delete telecommands originating from different sources, however in practice there can be restrictions on which source(s) request this.

If an error is detected during the processing of a request, nothing shall be deleted.

### 14.3.3.2 Deleting telecommands

The request to delete sets of telecommands from the command schedule shall be:

#### Delete Telecommands (11,5)

Telecommand packet, application data:

N	Application Process ID	Sequence Count	Number of Telecommands
Unsigned Integer	Enumerated	Unsigned Integer	Unsigned Integer



N:

This field shall be systematically omitted if the service does not support the concept of “scatter delete” (i.e. N=1).

Sequence Count:

The sequence count of the first telecommand packet to be sent to the specified destination application process which shall be deleted.

The (APID, Sequence Count) pair uniquely identifies a telecommand packet. These parameters shall correspond to the packet header fields of each telecommand packet as defined in subclause 5.3.

Number of Telecommands:

The number of successive telecommand packets sent by the source to the specified destination application process which shall be deleted.

When this request is received, all telecommands which satisfy the selection criteria defined by the APID, Sequence Count and the Number of Telecommands shall be deleted. An error shall occur if the first telecommand to be deleted is not found in the command schedule. If the Number of Telecommands exceeds the total number of commands that satisfy the selection criteria, then all commands that satisfy the selection criteria shall be deleted.

The deletion of telecommands which have times relative to scheduling events which have not yet occurred can only be performed by means of this type of request.

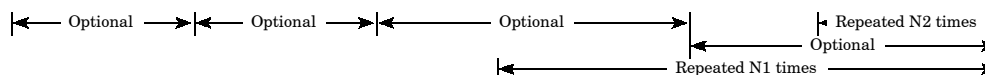
### 14.3.3.3 Deleting telecommands over a time period

The request shall be:

#### Delete Telecommands over Time Period (11,6)

Telecommand packet, application data:

Range	Time Tag 1	Time Tag 2	N1	Sub-schedule ID	N2	Application Process ID
Enumerated	Absolute Time	Absolute Time	Unsigned Integer	Enumerated	Unsigned Integer	Enumerated



Range:

This indicates that the time period is:

- from the beginning to the end of the command schedule if Range is “All” (value = 0); or
- between Time Tag 1 and Time Tag 2 inclusive if Range is “Between” (value = 1); or

- (c) less than or equal to Time Tag 1 if Range is "Before" (value = 2); or
- (d) greater than or equal to Time Tag 1 if Range is "After" (value = 3).

The service may only support limited values of Range.

#### Time Tag 1:

The earliest absolute time if Range is "Between" or "After". The latest absolute time if Range is "Before". This parameter shall not be present if Range is "All".

#### Time Tag 2:

The latest absolute time if Range is "Between". This parameter shall not be present if Range is not "Between".

#### N1:

This field shall be systematically omitted if the service does not support the concept of sub-schedule or the concept of selection at sub-schedule level. In this case, the Sub-schedule ID field shall also be omitted.

#### Sub-schedule ID:

The identification of the sub-schedule(s) from which telecommands shall be deleted. By convention, the value 0 for Sub-schedule ID shall mean "all sub-schedules".

#### N2:

This field shall be systematically omitted if the service does not support the concept of selection at application process level. In this case, the APID field shall also be omitted.

#### Application Process ID:

The identification of the destination application process(es) from which telecommands shall be deleted.

When this request is received, the following telecommands shall be deleted if they have release times falling in the specified absolute time period:

- a. if  $N1 = 0$ , all telecommands;
- b. if  $N1 > 0$  and  $N2 = 0$ , those telecommands which belong to the specified sub-schedules;
- c. if  $N1 > 0$  and  $N2 > 0$ , those telecommands which have the specified destination application processes and belong to the specified sub-schedules.

Those telecommands whose absolute release times are not yet known shall not be deleted from the command schedule. A telecommand has an unknown release time if it has a time relative to (directly or indirectly) a scheduling event which has not yet occurred.

However, if telecommand B has a time relative to the completion of execution of telecommand A which itself has not yet completed execution but which has a known release time then the release time of telecommand B shall be evaluated under the assumption that the execution duration of telecommand A is zero.

### 14.3.4 Time-shifting of telecommands in the command schedule

#### 14.3.4.1 General

There are several ways of time-shifting a subset of the telecommands in the command schedule. The time-shift request shall contain the time offset to be added (which can be a positive or negative value) and shall specify the selection of telecommands to which this time-offset is applied.

The scheduling service shall refuse to time-shift a telecommand if its new absolute time would fall in the past or before the end of the execution window of its

interlocking telecommand (if it is interlock dependent) or if its new relative time would become negative.

A service user can time-shift telecommands originating from different sources, however in practice there can be restrictions on which source(s) request this.

If an error is detected during the processing of a request, nothing shall be time-shifted.

#### 14.3.4.2 Time-shifting all telecommands

The request to time-shift all telecommands in the command schedule shall be:

##### Time-Shift All Telecommands (11,15)

Telecommand packet, application data:

Time Offset
Relative Time

Time Offset:

A positive or negative interval of time expressed in the length and format of relative time defined for the service or mission (since it is the relative time between the new and the old values of release time).

When this request is received, the release times for all telecommands in the command schedule which have an absolute release time shall be modified by adding the specified time offset.

Those telecommands whose absolute release times are not yet known are not time-shifted. A telecommand has an unknown release time if it has a time relative to (directly or indirectly) a scheduling event which has not yet occurred.

#### 14.3.4.3 Time-shifting selected telecommands

The request to time-shift a selected subset of telecommands in the command schedule shall be:

##### Time-Shift Selected Telecommands (11,7)

Telecommand packet, application data:

Time Offset	N	Application Process ID	Sequence Count	Number of Telecommands
Relative Time	Unsigned Integer	Enumerated	Unsigned Integer	Unsigned Integer

← Optional → ← Repeated N times →

N:

This field shall be systematically omitted if the service does not support the concept of “scatter time-shift” (i.e. N=1).

When this request is received the release times in the command schedule shall be time-shifted for those telecommands which meet the selection criteria defined by the specified APID, Sequence Count and Number of Telecommands. An error shall occur if the first telecommand to be time-shifted is not found in the command schedule. Those telecommands whose absolute release times are not yet known are not time-shifted. A telecommand has an unknown release time if it has a time relative to (directly or indirectly) a scheduling event which has not yet occurred.

#### 14.3.4.4 Time-shifting selected telecommands over a time period

The request shall be:

##### Time-Shift Selected Telecommands over Time Period (11,8)

Telecommand packet, application data:

Range	Time Tag 1	Time Tag 2	Time Offset
Enumerated	Absolute Time	Absolute Time	Relative Time

← Optional → ← Optional →

N1	Sub-schedule ID	N2	Application Process ID
Unsigned Integer	Enumerated	Unsigned Integer	Enumerated

← Optional → ← Optional →  
 ← Repeated N1 times → ← Repeated N2 times →

Range:

The service may only support particular values of Range.

N1:

This field shall be systematically omitted if the service does not support the concept of sub-schedule (which is equivalent to having N1=1). In this case, the Sub-schedule ID field shall also be omitted.

N2:

This field shall be systematically omitted if the service does not support the selective time-shifting of telecommands in a time range (which is equivalent to having N2=0).

By convention, the value 0 for Sub-schedule ID shall mean “all sub-schedules”.

When this request is received, the release times of the following telecommands shall be time-shifted if they have release times falling in the specified absolute time period:

- if N1 = 0, all telecommands;
- if N1 > 0 and N2 = 0, those telecommands which belong to the specified sub-schedules;
- if N1 > 0 and N2 > 0, those telecommands which have the specified destination application processes and belong to the specified sub-schedules.

Those telecommands whose absolute release times are not yet known are not time-shifted. A telecommand has an unknown release time if it has a time relative to (directly or indirectly) a scheduling event which has not yet occurred.

### 14.3.5 Reporting of the command schedule contents

#### 14.3.5.1 General

There are several ways of reporting a subset of the telecommands in the command schedule. The requests shall indicate whether a summary or detailed report is produced.

The reports shall contain information on telecommands originating from all sources. The information shall be ordered according to the predicted times of telecommand release.

Only those telecommands shall be reported for which the time of release has not yet expired.

Telecommands can be released between the reception of the service request and the completion of the service report. However, the report shall contain a consistent view which reflects the situation at the report packet time.

#### 14.3.5.2 Detailed reporting of the command schedule

The request to obtain a detailed report of all telecommands in the command schedule shall be:

##### Report Command Schedule in Detailed Form (11,16)

Telecommand packet, application data: None

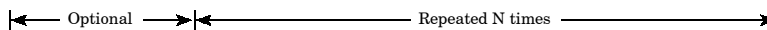
When this request is received, a detailed schedule report (see below) shall be generated containing all telecommands in the command schedule.

The request to obtain a detailed report of a selected subset of telecommands in the command schedule shall be:

##### Report Subset of Command Schedule in Detailed Form (11,9)

Telecommand packet, application data:

N	Application Process ID	Sequence Count	Number of Telecommands
Unsigned Integer	Enumerated	Unsigned Integer	Unsigned Integer



N:

This field shall be systematically omitted if the service does not support the concept of “scatter report” (i.e. N=1).

When this request is received, a detailed schedule report shall be generated containing those telecommands in the command schedule which meet the selection criteria defined by the combination of APID, Sequence Count and Number of Telecommands. An error shall occur if the first telecommand to be reported is not found in the command schedule.

The detailed schedule report shall contain all the static scheduling attributes for the selected telecommands. Telecommands whose release time is not yet known can only be selectively reported by means of this type of request.



**Detailed Schedule Report (11,10)**

Telemetry source packet, source data:

N	Sub-schedule ID	Interlock Set ID	Interlock Assessed ID	Assessment Type
Unsigned Integer	Enumerated	Enumerated	Enumerated	Enumerated

Optional      Optional      Optional  
 Repeated N times

Scheduling Event	Abs/Rel Time Tag	Execution Timeout	Telecommand Packet
Enumerated	Absolute or Relative Time	Relative Time	Any TC

Optional      Optional  
 Repeated N times (contd.)

Sub-schedule ID:

This field shall be systematically omitted if the service does not support the concept of sub-schedules.

Interlock Set ID, Interlock Assessed ID:

These fields shall be systematically omitted if the service does not support the concept of interlocking.

Scheduling Event:

This field shall be systematically omitted if the service does not support the concept of relative time.

Relative time shall be placed in the report when the absolute release time of the telecommand is still unknown (e.g. even if its release is relative to completion of execution of another telecommand whose release time is known but whose execution has not yet started or is not yet complete). The corollary is that the report can indicate an "Absolute" time even though a relative time was originally loaded for the corresponding telecommand. This happens if the event to which the relative time relates has occurred in the meantime. The service shall then elaborate the relative time to an absolute time and it can report it as such.

**14.3.5.3 Summary reporting of the command schedule**

The request to obtain a summary report of all telecommands in the command schedule shall be:

**Report Command Schedule in Summary Form (11,17)**

Telecommand packet, application data: None

When this request is received, a summary schedule report (see below) shall be generated containing all telecommands in the command schedule.

The request to obtain a summary report of a selected subset of telecommands in the command schedule shall be:

**Report Subset of Command Schedule in Summary Form (11,12)**

Telecommand packet, application data: Same as for the "Report Command Schedule in Detailed Form" service request.

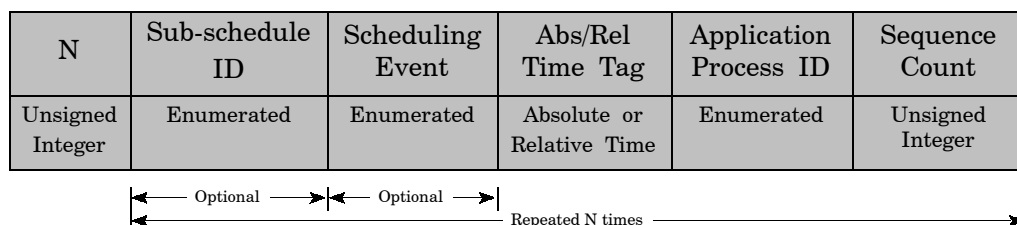
When this request is received, a report shall be generated containing those telecommands in the command schedule which meet the selection criteria defined by the combination of APID, Sequence Count and Number of Telecommands. An

error occurs if the first telecommand to be reported is not found in the command schedule.

The summary schedule report shall contain only the identifications for the selected telecommands. Telecommands whose release time is not yet known can only be selectively reported by means of this type of request.

#### Summary Schedule Report (11,13)

Telemetry source packet, source data:

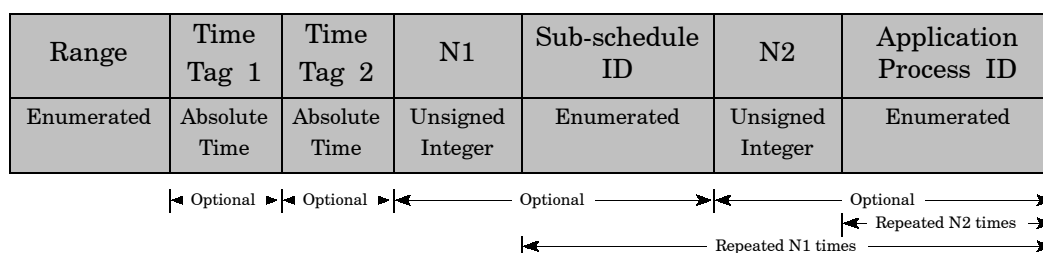


#### 14.3.5.4 Detailed reporting of the command schedule over a time period

The request for a detailed report of a selected subset of the command schedule over an absolute time period shall be:

#### Report Subset of Command Schedule in Detailed Form over Time Period (11,11)

Telecommand packet, application data:



Range:

The service may only support particular values of Range.

N1:

This field shall be systematically omitted if the service does not support the concept of sub-schedule (which is equivalent to having N1=1). In this case, the Sub-schedule ID field shall also be omitted.

N2:

This field shall be systematically omitted if the service does not support the selective reporting of telecommands in a time range (which is equivalent to having N2=0).

When this request is received, a Detailed Schedule Report (as defined in subclause 14.3.5.2) shall be generated. The report shall cover the following telecommands if they have release times falling within the specified absolute time period:

- if N1 = 0, all telecommands;
- if N1 > 0 and N2 = 0, those telecommands which belong to the specified sub-schedules;
- if N1 > 0 and N2 > 0, those telecommands which have the specified destination application processes and belong to the specified sub-schedules.

Telecommands whose absolute release times are not yet known shall not be included in the report. A telecommand has an unknown release time if it has a time relative to (directly or indirectly) a scheduling event which has not yet occurred.

However, if telecommand B has a time relative to the completion of execution of telecommand A which itself has not yet been executed but which has a known release time, then the release time of telecommand B shall be evaluated under the assumption that the execution duration of telecommand A is zero.

#### 14.3.5.5 Summary reporting of the command schedule over a time period

The request for a summary report of a selected subset of the command schedule over an absolute time period shall be:

##### Report Subset of Command Schedule in Summary Form over Time Period (11,14)

Telecommand packet, application data: Same as for the "Report Command Schedule in Detailed Form over Time Period" service request.

When this request is received, a summary schedule report (as defined in subclause 14.3.5.3) shall be generated. The report shall cover the telecommands which meet the same selection criteria as defined in subclause 14.3.5.4.

#### 14.3.6 Reporting of the status of the command schedule

The request to report the status of the command schedule shall be:

##### Report Status of Command Schedule (11,18)

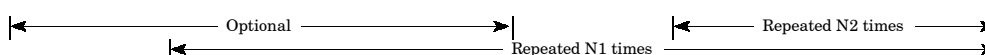
Telecommand packet, application data: None

When this request is received, a report shall be generated containing the status (enabled or disabled) of the individual sub-schedules and application processes within each sub-schedule.

##### Command Schedule Status Report (11,19)

Telemetry source packet, source data:

N1	Sub-schedule ID	Status	N2	Application Process ID	Status
Unsigned Integer	Enumerated	Enumerated	Unsigned Integer	Enumerated	Enumerated



N1:

This field shall be systematically omitted if the service does not support the concept of sub-schedules (which is equivalent to having N1=1). In this case, the Sub-schedule ID field and the Status fields shall also be omitted and the application data is simply an array of N2 APIDs + Statuses.

Sub-schedule ID:

The identification of the sub-schedule being reported.

Status:

The status of the corresponding sub-schedule, as follows:

value = 0 (Disabled)

value = 1 (Enabled).

Application Process ID:

The identification of the application process within the corresponding sub-schedule being reported.

Status:

The status of the corresponding application process, as follows:

value = 0 (Disabled)

value = 1 (Enabled).

## 14.4 Capability sets

### 14.4.1 Minimum capability set

The minimum capability set shall consist of the service requests and reports given in Table 23, but some of them only with particular combinations of parameters as specified in the subclauses above.

**Table 23: Summary of on-board operations scheduling service minimum capabilities**

Subtype	Service request, report or capability
1	Enable Release of Telecommands
2	Disable Release of Telecommands
3	Reset Command Schedule
4	Insert Telecommands in Command Schedule

### 14.4.2 Additional capability sets

An on-board operations scheduling service can implement the additional capabilities given in Table 24.

**Table 24: Summary of on-board operations scheduling service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
	Support of sub-schedules	“Sub-schedule ID” in subtypes 4, 10 and 13 N1 and corresponding array in subtypes 1 and 2
	Support of one or more types of relative times	“Scheduling Event” in subtypes 4, 10 and 13
	Support of the interlocking command concept	“Interlock Set ID” and “Interlock Assessed ID” in subtypes 4 and 10
	Support of “scatter” delete, time-shift and report	N and corresponding array in subtypes 5, 7, 9 and 12
	Support of telecommand selection strategy at sub-schedule level	“Sub-schedule ID” in subtypes 6, 8, 11 and 14

**Table 24: Summary of on-board operations scheduling service additional capabilities (*continued*)**

Subtype	Service request, report or capability	Requires subtype
	Support of telecommand selection strategy at application process level	N2 and corresponding array in subtypes 1, 2, 6, 8, 11 and 14
5	Delete Telecommands	
6	Delete Telecommands over Time Period	5
15	Time-Shift All Telecommands	
7	Time-Shift Selected Telecommands	15
8	Time-Shift Telecommands over Time Period	7 and 15
16	Report Command Schedule in Detailed Form	10
9	Report Subset of Command Schedule in Detailed Form	10 and 16
10	Detailed Schedule Report	16 and/or 9 and/or 11
11	Report Subset of Command Schedule in Detailed Form over Time Period	9, 10 and 16
17	Report Command Schedule in Summary Form	13
12	Report Subset of Command Schedule in Summary Form	13 and 17
13	Summary Schedule Report	17 and/or 12 and/or 14
14	Report Subset of Command Schedule in Summary Form over Time Period	12, 13 and 17
18	Report Status of Command Schedule	19
19	Command Schedule Status Report	18

*(This page is intentionally left blank)*

---

# 15

---

## On-board monitoring service

### 15.1 Scope

The on-board monitoring service provides the capability to monitor on-board parameters with respect to checks defined by the ground system and reports any check transitions to the service user. Optionally, an event report may be generated as the result of a given monitoring violation. To achieve this, the service maintains a monitoring list and checks parameter samples according to the information contained therein.

This service satisfies the concepts described in subclause 4.8 of this Standard.

Any number of on-board application processes may provide a single instance of the on-board monitoring service.

Telemetry source packets and telecommand packets relating to the on-board monitoring service shall be denoted by Service Type = 12.

### 15.2 Service concept

#### 15.2.1 General

A monitoring list shall be maintained which contains the parameter monitoring information, drives the parameter monitoring activity and the generation of check transition reports.

The ground segment can modify or report the contents of the monitoring list using service requests to:

- a. reset the monitoring list;
- b. add parameters to, or delete parameters from, the monitoring list;
- c. modify the monitoring information of parameters in the monitoring list;
- d. enable or disable the monitoring of parameters in the monitoring list;
- e. report the monitoring information for all parameters in the monitoring list;
- f. report the set of parameters which are currently out-of-limits.

The ground system can also modify attributes of the on-board monitoring service which determine:

- a. Whether the monitoring of parameters is enabled or disabled at service level.
- b. The maximum reporting delay for the check transition report. A check transition report should be issued with no greater delay than this after a new check transition has occurred.

The value of this parameter has an impact both on the average number of check transitions reported in a given check transition report and on the resolution with which the times of reported check transitions are known on the ground (if the reported check transitions are not individually time-stamped).

### 15.2.2 The monitoring list

The on-board monitoring service shall maintain static monitoring information for each parameter to be monitored, which is provided by the ground system by means of service requests. The parameter monitoring information shall specify:

- a. the identification of the on-board parameter to be monitored;
- b. whether the monitoring of the parameter is enabled or disabled;
- c. the associated validity parameter (if any); this is a Boolean parameter whose value determines whether the parameter is monitored;
- d. the monitoring interval for the parameter, expressed in units of `<DIAG_MIN_INTERV>`.

The parameter monitoring information shall also include a set of check definitions. A check definition shall provide the information to check a sample of the parameter against either one pair of limits, one expected value or one pair of delta thresholds. More than one check definition can be associated with a given parameter. Both limit and delta checks can be defined for a given parameter.

A check definition shall indicate:

- The nature of the check to be performed. This can be a limit-check, a delta-check or an expected-value-check.
  - For a limit-check, a low-limit value and a high-limit value shall be specified.
  - For a delta-check, a minimum-delta value and a maximum-delta value shall be specified.
  - For an expected-value-check, an expected value shall be specified.
- A check selection parameter, which is a Boolean on-board parameter whose value determines whether the check against the limit pair (or expected value or delta threshold pair) shall be applied. If no selection parameter is provided, the check shall always be applied.
- A “number of repetitions (#REP)”.
  - For a limit-check or an expected-value-check, this indicates the number of successive samples of the parameter that fail (or succeed) the check before establishing a new checking status for the parameter.
  - For a delta-check, this is the number of consecutive delta values to be used to evaluate an average delta value which is checked against the delta check definition. A delta value is the arithmetic difference between successive samples of a parameter.
- The identifier (RID) of an event report that shall be generated if the corresponding check fails.

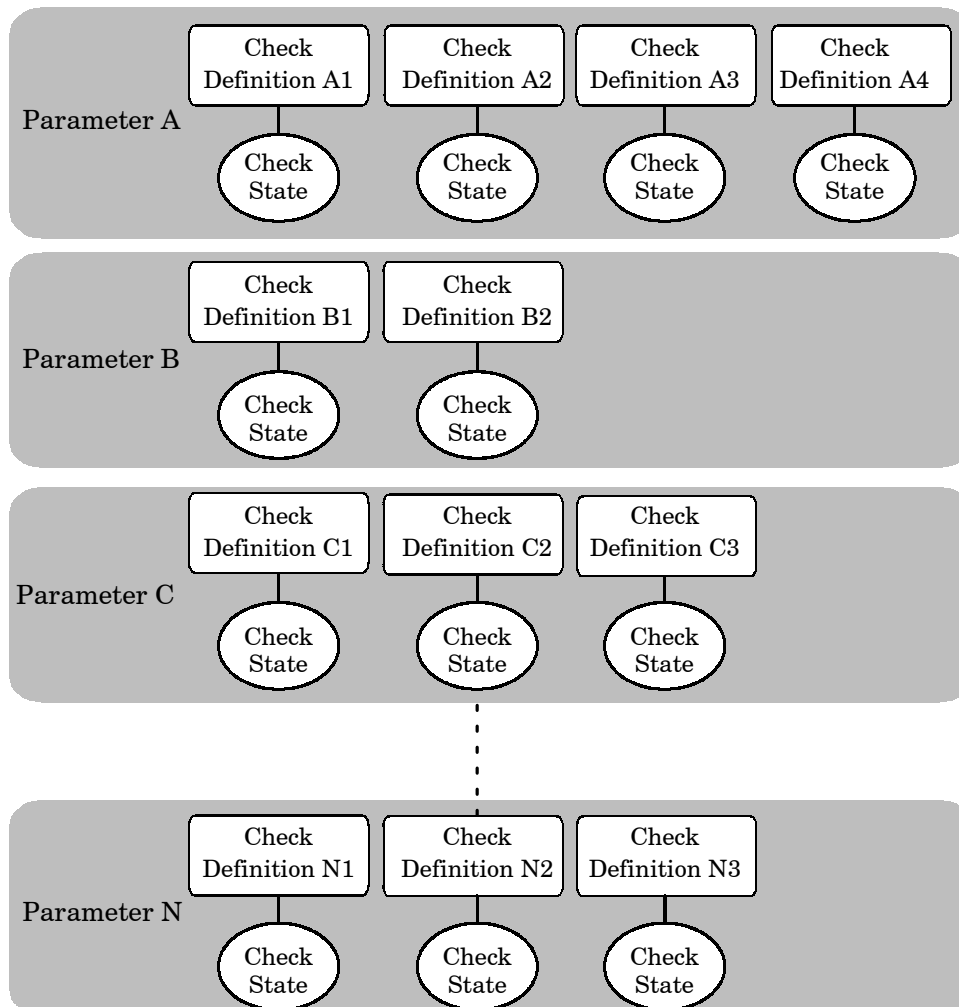


### 15.2.3 The checking activity and the check state

The on-board monitoring service shall maintain a check state corresponding to each check definition for each parameter to be monitored.

The check state shall include information about the previous and current checking status of the parameter for the given check definition and the time at which the transition to that checking status occurred. This information shall be downlinked when the ground system requests a report of the parameters which are currently out-of-limit.

**Monitoring List**



**Figure 6: Parameter check definitions and check status**

A check definition shall be “enabled” and used for checking a parameter when all the following conditions are set:

- the monitoring of parameters is enabled at service-level,
- the monitoring of the parameter is enabled,
- the parameter is valid (check validity parameter value = “TRUE”), and
- the check definition is selected for checking the sample (its check selection parameter value = “TRUE”).

Otherwise the check definition shall be “disabled” and shall not be used for checking the parameter.

Whenever a sample of the parameter is available for checking, the service shall perform the following checking activity independently for each parameter check definition (and update its check state accordingly):

- a. If the check definition is “disabled” then the new checking status shall immediately become either “Unchecked”, “Invalid” or “Unselected” depending on whether the checking of the parameter is disabled, the parameter is invalid or the check definition is not selected for checking.

By default, the initial current checking status of a parameter with respect to the check definition shall be “Unchecked” when the parameter is added to the monitoring list or when a new check definition for the parameter is added at a later time.

- b. If the check definition is “enabled” then the parameter sample is a valid sample for checking. It shall be checked against the limit pair (or expected value, or used to elaborate a new average delta value which shall be checked against the delta threshold pair) if sufficient consecutive valid samples have been accumulated.

For a limit-check or expected-value-check, if the last #REP successive valid samples of the parameter (including the current one) have consistently failed (or consistently passed) the check, then the parameter shall be assigned a new checking status. The new checking status shall be equal to the result of the check of the current sample, i.e. either “Below low limit”, “Above high limit”, “Within limits”, “Unexpected value” or “Expected value”.

When the previously determined checking status of a parameter with respect to a limit-check was “Within limits”, and when successive samples are alternately “Below low limit” and “Above high limit”, these earn the parameter a new checking status. However, when the last known checking status was “Above high limit”, then a sequence of consecutive “Below low limit” samples shall be available before a new checking status is assigned.

For a delta-check, once #REP successive valid samples of the parameter have been accumulated, a new mean delta value shall be evaluated and checked. The new checking status shall be set to the result of the check which shall be either “Below low threshold”, “Above high threshold” or “Within thresholds”.

- c. Having elaborated a new checking status for the parameter, a comparison between the previous and new checking statuses shall be performed. If they differ, then a check transition shall be recorded (conceptually this is recorded in a transition reporting list, see subclause 15.2.4).
- d. When a check transition is detected, the transition time shall be recorded in the corresponding check state. This shall be the sampling time of the first parameter sample which was used to establish the new checking status.

If a check transition occurs for which the check definition identifies an associated event report, a telemetry packet of type 5, subtype 4 “high severity” shall be generated, containing the specified report identifier (RID) but with no auxiliary parameters - see subclause 10.3. The check transitions concerned are those where the checking status changes to “Below low limit”, “Above high limit”, “Unexpected value”, “Below low threshold” or “Above high threshold (depending on the parameter and check type) where it was previously something different.

The current checking status and associated transition times can be reported to the ground system on request.

#### 15.2.4 The transition reporting list

During the course of the monitoring activity, an ordered list of checking status transitions shall be established. Within this list, there can be more than one checking status transition for a given parameter (e.g. transitions relating to different check definitions; transitions corresponding to different samples of the parameter).

Each checking status transition in the list shall be characterized by:

- the parameter for which the checking status transition was detected;
- the value of the parameter at the time the checking status transition was detected;
- the type of the transition, defined by the previous and the new checking statuses;
- the value of the limit, delta threshold or expected value which was crossed or violated.

The transition reporting list shall be downlinked via a check transition report no later than the maximum reporting delay after the time of the first transition in the list. The list shall be cleared after downlink.

### 15.2.5 Auxiliary information

It is assumed that the on-board monitoring service has access to other information used for the detection of errors in the processing of service requests. This includes the following:

- The maximum number of entries of the monitoring list.
- The list of parameters which can be accessed, and can thus be monitored, by the application process.
- The type(s) of check for each parameter which can be monitored (delta-check, limit-check or expected-value-check).
- The list of Boolean on-board parameters which can be accessed by the application process and can be used as validity parameter or check definition selection parameters.

## 15.3 Service requests and reports

### 15.3.1 Controlling the on-board monitoring

The capability shall exist to enable or disable the monitoring of parameters globally or to enable or disable the monitoring of a specified subset of parameters. The requests shall be:

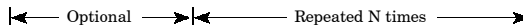
#### Enable Monitoring of Parameters (12,1)

Telecommand packet, application data:

#### Disable Monitoring of Parameters (12,2)

Telecommand packet, application data:

N	Parameter#
Unsigned Integer	Enumerated



N:

The number of parameters whose monitoring shall be enabled or disabled. By convention, N = 0 shall mean "Enable/disable monitoring at service-level".

This field shall be systematically omitted if the service only supports the control of one parameter at a time (i.e. N=1).

**Parameter#:**

The identification of a parameter.

When the service provider receives this request:

- a. If  $N = 0$ , it shall set the service level monitoring status to “Enabled” or “Disabled”, depending on the request subtype.

If “Enable” is requested, the parameters in the monitoring list whose parameter level monitoring status is “Enabled” shall start to be monitored.

If “Disable” is requested, none of the parameters in the monitoring list shall be monitored any more. Also, the current checking status for all check definitions in the monitoring list shall be set to “Unchecked” and their times of transition shall be set to the current time. The current content of the transition reporting list shall not be affected by these activities. If the list is not empty, its content shall be reported as usual via a check transition report.

- b. If  $N > 0$ , each parameter in the request shall be processed in turn and its parameter level monitoring status shall be set to “Enabled” or “Disabled”, depending on the request subtype.

If the monitoring of parameters is enabled at service-level and “Enable” is requested, the monitoring of the parameters specified in the request shall start immediately after the processing of the request.

An error shall be flagged if the parameter is not in the list. However, the processing of the remaining parameters shall not be affected.

### 15.3.2 Changing the maximum reporting delay

The request shall be:

#### Change Maximum Reporting Delay (12,3)

Telecommand packet, application data:

Max Reporting Delay
Unsigned Integer

Max Reporting Delay:

The maximum reporting delay for the check transition report, expressed in units of <DIAG\_MIN\_INTERV>.

When the service provider receives this request, the maximum reporting delay shall be recorded and used to determine when to downlink the transition reporting list.

### 15.3.3 Clearing the monitoring list

The request shall be:

#### Clear Monitoring List (12,4)

Telecommand packet, application data: None.

When the service provider receives this request, it shall set the service monitoring status to “Disabled” and clears all entries in the monitoring list and in the transition reporting list.

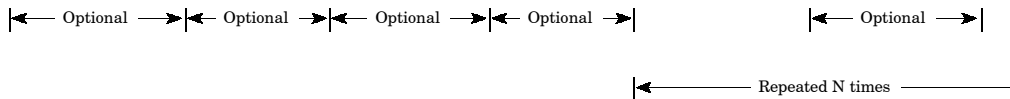
### 15.3.4 Adding parameters to the monitoring list

The request shall be:

#### Add Parameters to Monitoring List (12,5)

Telecommand packet, application data:

Parameter Monitoring Interval	Value #REP	Delta #REP	N	Parameter#	Validity Parameter#
Unsigned Integer	Unsigned Integer	Unsigned Integer	Unsigned Integer	Enumerated	Enumerated



NOL	Check Selection Parameter#	Low Limit	RID	High Limit	RID
Unsigned Integer	Enumerated	Deduced	Enumerated	Deduced	Enumerated

Diagram showing the structure of the Add Parameters to Monitoring List (12,5) telecommand packet. The packet consists of six optional fields, each indicated by a double-headed arrow labeled "Optional". The fields are: NOL, Check Selection Parameter#, Low Limit, RID, High Limit, and RID. The entire structure is repeated NOL times, as indicated by a double-headed arrow labeled "Repeated NOL times" below the fields. The entire structure is repeated N times (contd.), as indicated by a double-headed arrow labeled "Repeated N times (contd.)" below the fields.

NOD	Check Selection Parameter#	Low Delta Threshold	RID	High Delta Threshold	RID
Unsigned Integer	Enumerated	Deduced	Enumerated	Deduced	Enumerated

Diagram showing the structure of the Add Parameters to Monitoring List (12,5) telecommand packet. The packet consists of six optional fields, each indicated by a double-headed arrow labeled "Optional". The fields are: NOD, Check Selection Parameter#, Low Delta Threshold, RID, High Delta Threshold, and RID. The entire structure is repeated NOD times, as indicated by a double-headed arrow labeled "Repeated NOD times" below the fields. The entire structure is repeated N times (contd.), as indicated by a double-headed arrow labeled "Repeated N times (contd.)" below the fields.

NOE	Check Selection Parameter#	Expected Value	RID
Unsigned Integer	Enumerated	Deduced	Enumerated

Diagram showing the structure of the Add Parameters to Monitoring List (12,5) telecommand packet. The packet consists of four optional fields, each indicated by a double-headed arrow labeled "Optional". The fields are: NOE, Check Selection Parameter#, Expected Value, and RID. The entire structure is repeated NOE times, as indicated by a double-headed arrow labeled "Repeated NOE times" below the fields. The entire structure is repeated N times (contd.), as indicated by a double-headed arrow labeled "Repeated N times (contd.)" below the fields.

#### Parameter Monitoring Interval:

The monitoring interval for the parameters, in units of <DIAG\_MIN\_INTERV>.

This field shall be systematically omitted if the service knows which value to use.

**Value #REP:**

The number of successive samples of the parameters to establish a new checking status for an expected-value-check or a limit-check.

This field shall be systematically omitted if the service knows which value to use for the number of successive samples, or if limit and expected-value-checks are not supported by the service.

**Delta #REP:**

The number of successive samples of the parameters to establish a new checking status for a delta-check.

This field shall be systematically omitted if the service knows which value to use for the number of successive samples or if delta-checks are not supported by the service.

**N:**

The number of parameters to be added to the monitoring list.

This field shall be systematically omitted if the service only supports the addition of one parameter at a time.

**Parameter#:**

The identification of a parameter to be monitored.

**Validity Parameter#:**

A Boolean parameter whose value determines whether a parameter is valid or not. By convention, if the validity parameter# is 0, the corresponding parameter is always valid (i.e. it shall always be checked).

This field shall be systematically omitted if the service does not support the concept of parameter validity.

**NOL, NOD, NOE:**

The number of limit-check definitions (or delta-check definitions or expected-value-check definitions) which follow.

These fields shall be systematically omitted if limit-checks, delta-checks or expected-value-checks are not supported by the service (i.e. NOL=0, NOD=0 or NOE=0 and the corresponding array is empty) or if the service supports only one check definition of this type per parameter. The service shall support at least one type of check.

**Check Selection Parameter#:**

The Boolean parameter whose value determines whether the associated check definition is applied. By convention, if the parameter# is 0, the associated check shall always be applied.

This field shall be systematically omitted, for a given type of check, if the service only supports one check definition of this type per parameter.

**RID:**

The identifier of the event report to be generated in the event of a monitoring violation.

Event reports may be defined for only a subset of the checks defined for a given parameter, for example, an event report may be defined for a high limit transgression but not for the corresponding low limit.

This field shall be systematically omitted if the service does not support the generation of event reports.

By convention, the value 0 for RID shall mean "no event report is generated".

The type and format of the Low Limit, High Limit, Low Delta Threshold, High Delta Threshold or Expected Value shall be the same as the type and format of the parameter to be monitored.

The number of parameters to be added to the monitoring list shall be limited to a maximum of <MONLIST\_MAX\_PARAMS>.

Where more than one check definition is defined for a given parameter, each has a “check position” which is determined by its position in the request (however, the check position may be changed by a subsequent “modify parameter checking information” request).

When the service provider receives this request, it shall add the parameter monitoring information to the monitoring list, shall initialize the check state of the check definitions and shall set the parameter monitoring status to “Disabled”.

If an error is detected during the processing of the monitoring information for a given parameter, this parameter shall not be added to the monitoring list. This shall not affect the processing of the remaining parameters.

A standard error occurs, if:

- the monitoring list is full,
- the parameter is already in the list,
- the parameter is not accessible<sup>5)</sup>,
- the validity parameter or check selection parameter is not accessible or is not Boolean.

### 15.3.5 Deleting parameters from the monitoring list

The request to delete specified parameters from the monitoring list shall be:

#### Delete Parameters from Monitoring List (12,6)

Telecommand packet, application data:

N	Parameter#
Unsigned Integer	Enumerated

◀ Optional ▶      ◀ Repeated N times ▶

N:

This field shall be systematically omitted if the service only supports the deletion of one parameter at a time (i.e. N=1).

When the service provider receives this request, it shall process each parameter in turn and shall remove its corresponding monitoring information, if any, from the monitoring list (the entry becomes free).

An error shall occur if the parameter is not in the monitoring list. This shall not affect the deletion of the parameters which have an entry in the monitoring list.

5) The application process has access to a given set of parameters. If this parameter does not lie within this set, it is deemed “not accessible”.

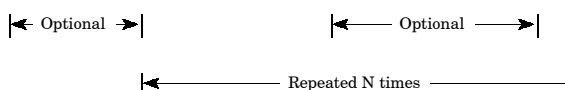
### 15.3.6 Modifying the parameter checking information

The request to modify the checking information for specified parameters shall be:

#### Modify Parameter Checking Information (12,7)

Telecommand packet, application data:

N	Parameter#	Validity Parameter#
Unsigned Integer	Enumerated	Enumerated



NOL	Check Position	Check Selection Parameter#	Low Limit	RID	High Limit	RID
Unsigned Integer	Signed Integer	Enumerated	Deduced	Enumerated	Deduced	Enumerated

Diagram showing the structure of the Modify Parameter Checking Information (12,7) packet. The packet consists of a sequence of fields, each preceded by an "Optional" flag. The fields are: NOL, Check Position, Check Selection Parameter#, Low Limit, RID, High Limit, and RID. The entire sequence is repeated NOL times. The sequence is repeated N times (contd.).

NOD	Check Position	Check Selection Parameter#	Low Delta Threshold	RID	High Delta Threshold	RID
Unsigned Integer	Signed Integer	Enumerated	Deduced	Enumerated	Deduced	Enumerated

Diagram showing the structure of the Modify Parameter Checking Information (12,7) packet. The packet consists of a sequence of fields, each preceded by an "Optional" flag. The fields are: NOD, Check Position, Check Selection Parameter#, Low Delta Threshold, RID, High Delta Threshold, and RID. The entire sequence is repeated NOD times. The sequence is repeated N times (contd.).

NOE	Check Position	Check Selection Parameter#	Expected Value	RID
Unsigned Integer	Signed Integer	Enumerated	Deduced	Enumerated

Diagram showing the structure of the Modify Parameter Checking Information (12,7) packet. The packet consists of a sequence of fields, each preceded by an "Optional" flag. The fields are: NOE, Check Position, Check Selection Parameter#, Expected Value, and RID. The entire sequence is repeated NOE times. The sequence is repeated N times (contd.).

N:

This field shall be systematically omitted if the service only supports modification of the monitoring information for one parameter at a time.

Validity Parameter#:

This field shall be systematically omitted if the service does not support the concept of parameter validity.



**NOL, NOD, NOE:**

These fields shall be systematically omitted if limit-checks, delta-checks or expected-value-checks are not supported by the service (i.e. NOL=0, NOD=0 or NOE=0 and the corresponding array is empty) or if the service supports only one check definition of this type per parameter.

**Check Position:**

This indicates which check definition (for the given parameter) is deleted, added or replaced with a new check definition.

A positive Check Position value indicates that the corresponding check definition for the parameter shall be replaced by the new check definition which follows.

A negative Check Position value indicates that the corresponding check definition in the positive range shall be deleted (the positions of succeeding check definitions are decremented). No check definition shall follow in this case. The service shall refuse to delete the last remaining check definition of a parameter and shall report the error.

If the Check Position value is 0, this indicates that the check definition which follows should be added to the monitoring list.

This field shall be systematically omitted, for a given type of check, if the service only supports one check definition of this type per parameter (equivalent to having Check Position=1).

**Check Selection Parameter#:**

This field shall be systematically omitted, for a given type of check, if the service only supports one check definition of this type per parameter.

**RID:**

This field shall be systematically omitted if the service does not support the generation of event reports.

The number of parameters whose entry in the monitoring list can be modified shall be limited to a maximum of <MONLIST\_MAX\_PARAMS>.

When the service provider receives this request, it shall process the checking information for each parameter in turn and (if no error is detected) shall replace, add or delete the specified check definitions. The check state for a new or modified check definition shall be initialized.

An error shall occur if the parameter is not in the list, if a position to be modified or deleted does not contain a check definition, if a check definition to be added cannot be added (too many check definitions for the parameter) or if a check selection parameter is not accessible or not Boolean. The parameter entry in the monitoring list shall be unchanged if any error is detected but the processing of the remaining parameters shall be unaffected.

### 15.3.7 Reporting the current monitoring list contents

The request shall be:

#### Report Current Monitoring List (12,8)

Telecommand packet, application data: None.

When the service provider receives this request, it shall issue a report with the current static contents of the monitoring list.

**Current Monitoring List Report (12,9)**

Telemetry source packet, source data:

Monitoring Status	Maximum Reporting Delay	N	Parameter#	Validity Parameter#	Parameter Monitoring Interval	Parameter Monitoring Status
Enumerated	Unsigned Integer	Unsigned Integer	Enumerated	Enumerated	Unsigned Integer	Enumerated

|← Optional →|

|← Repeated N times →|

Value #REP	Delta #REP	NOL	Check Selection Parameter#	Low Limit	RID	High Limit	RID
Unsigned Integer	Unsigned Integer	Unsigned Integer	Enumerated	Deduced	Enumerated	Deduced	Enumerated

|← Optional →| |← Optional →| |← Optional →| |← Optional →| |← Optional →|

|← Repeated NOL times →|

Repeated N times (contd.)

NOD	Check Selection Parameter#	Low Delta Threshold	RID	High Delta Threshold	RID
Unsigned Integer	Enumerated	Deduced	Enumerated	Deduced	Enumerated

|← Optional →| |← Optional →| |← Optional →|

|← Repeated NOD times →|

Repeated N times (contd.)

NOE	Check Selection Parameter#	Expected Value	RID
Unsigned Integer	Enumerated	Deduced	Enumerated

|← Optional →| |← Optional →|

|← Repeated NOE times →|

Repeated N times (contd.)

**Monitoring Status:**

This indicates whether the overall monitoring is “enabled” (value = 1) or “disabled” (value = 0).

**Validity Parameter#:**

This field shall be systematically omitted if the service does not support the concept of parameter validity.

**Parameter Monitoring Status:**

This indicates whether the monitoring of the corresponding parameter is “enabled” (value = 1) or “disabled” (value = 0).

**Value #REP:**

This field shall be systematically omitted if expected-value and limit-checks are not supported by the service.

**Delta #REP:**

This field shall be systematically omitted if delta-checks are not supported by the service.

**NOL, NOD, NOE:**

These fields shall be systematically omitted if limit-checks, delta-checks or expected-value-checks are not supported by the service (i.e. NOL=0, NOD=0 or NOE=0 and the corresponding array is empty) or if the service only supports one check definition of this type per parameter.

**Check Selection Parameter#:**

This field shall be systematically omitted, for a given type of check, if the service only supports one check definition of this type per parameter.

**RID:**

This field shall be systematically omitted if the service does not support the generation of event reports.

### 15.3.8 Reporting the current parameters out-of-limit list

The request shall be:

#### Report Current Parameters Out-of-limit List (12,10)

Telecommand packet, application data: None.

When the service provider receives this request, it shall issue a report containing the check states for those check definitions which indicate that the current checking status of the parameter is equal to “Below low limit”, “Above high limit”, “Below low threshold”, “Above high threshold” or “Unexpected Value”. A parameter can appear more than once in the report (e.g. if it violates two of its check definitions).

#### Current Parameters Out-of-limit List Report (12,11)

Telemetry source packet, source data:

N	Parameter#	Parameter Value	Limit Crossed	Previous Checking Status	Current Checking Status	Transition Time
Unsigned Integer	Enumerated	Deduced	Deduced	Enumerated	Enumerated	Absolute Time

← Repeated N times →

**Parameter value:**

This shall give the value of the telemetry parameter at the time the last checking status transition was detected. The format and length shall be uniquely defined for the parameter.

**Limit Crossed:**

This shall be the value of the Low Limit, High Limit, Low Delta Threshold, High Delta Threshold or Expected Value which has been crossed or violated. It shall have the same format and length as the value of the parameter itself.

**Previous Checking Status:**

This indicates the checking status of the parameter before the transition to the current checking status. The possible values are defined below.

**Current Checking Status:**

This indicates the current checking status of the parameter. The possible values are defined below.

**Transition Time:**

The time at which the transition occurred, i.e. the time of the first sample used to elaborate the current checking status.

The possible values for the Previous Checking Status and the Current Checking Status shall be:

"Expected Value", "Within Limits", "Within Thresholds":	value = 0
"Unchecked":	value = 1
"Invalid":	value = 2
"Unselected":	value = 3
"Unexpected Value", "Below Low Limit", "Below Low Threshold":	value = 4
"Above High Limit", "Above High Threshold":	value = 5

Only certain combinations of values for the two fields are meaningful for a given type of check as specified below with the (X→Y) convention in which "X" is a "Previous Checking Status" field value and "Y" is a "Current Checking Status" field value.

For an expected-value-check, the only reported transition types shall be (U→UV, I→UV, US→UV, EV→UV), where the mnemonics appearing in the transition types mean:

EV:	Expected Value	US:	Unselected
U:	Unchecked	UV:	Unexpected Value
I:	Invalid		

For a limit-check, the only reported transition types shall be (U→BL, U→AL, I→BL, I→AL, US→BL, US→AL, WL→BL, WL→AL, BL→AL, AL→BL), where the mnemonics appearing in the transition types mean:

WL:	Within Limits	US:	Unselected
U:	Unchecked	BL:	Below Low Limit
I:	Invalid	AL:	Above High Limit

For a delta-check, the only reported transition types shall be (U→BT, U→AT, I→BT, I→AT, US→BT, US→AT, WT→BT, WT→AT, BT→AT, AT→BT), where the mnemonics appearing in the transition types mean:

WT:	Within Thresholds	US:	Unselected
U:	Unchecked	BT:	Below Low Threshold
I:	Invalid	AT:	Above High Threshold

### 15.3.9 Reporting the check transitions

The check transition report is the only provider-initiated report and shall contain the contents of the transition reporting list established since the last time a check transition report was issued. It shall consist of:

#### Check Transition Report (12,12)

Telemetry source packet, source data:

N	Parameter#	Parameter Value	Limit Crossed	Previous Checking Status	Current Checking Status	Transition Time
Unsigned Integer	Enumerated	Deduced	Deduced	Enumerated	Enumerated	Absolute Time



For an expected-value-check, the Previous Checking Status and Current Checking Status field values can be any of the following values (but shall be different):

Unchecked, Invalid, Unselected, Expected Value or Unexpected Value

For a limit-check, the Previous Checking Status and Current Checking Status field values can be any of the following values (but shall be different):

Unchecked, Invalid, Unselected, Within Limits, Below Low Limit or Above High Limit

For a delta-check, the Previous Checking Status and Current Checking Status field values can be any of the following values (but shall be different):

Unchecked, Invalid, Unselected, Within Thresholds, Below Low Threshold or Above High Threshold

The values used for encoding the Previous Checking Status and Current Checking Status fields shall be as defined for the Current Parameter Out-of-limit List report in the previous subclause (e.g. value = 0 for "Expected Value", "Within Limits" and "Within Thresholds").

The same Parameter# can appear several times in a report.

The Transition Time field shall be systematically omitted if the maximum reporting delay supported by the service is always smaller than the precision with which the transition time shall be known (it may be omitted unless the maximum reporting delay is large compared with the parameter monitoring intervals for some parameters).

The check transition report shall be generated when at least one transition has been entered in the transition reporting list and no later than the maximum reporting delay after the time of the first transition in the transition reporting list.

The transition reporting list shall be cleared immediately after the report has been generated.

## 15.4 Capability sets

### 15.4.1 Minimum capability set

The minimum capability set shall consist of the service requests and reports given in Table 25.

**Table 25: Summary of on-board monitoring service minimum capabilities**

Subtype	Service request, report or capability
1	Enable Monitoring of Parameters
2	Disable Monitoring of Parameters
12	Check Transition Report

### 15.4.2 Additional capability sets

The on-board monitoring service can implement the additional capabilities given in Table 26.

**Table 26: Summary of on-board monitoring service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
	Support of limit, expected-value or delta-checks (minimum one type to be supported)	NOL, NOE or NOD and corresponding array in subtypes 5, 7 and 9
	Support of validity parameter concept	"Validity Parameter#" in subtypes 5, 7 and 9
	Support of several check definitions of a given type per parameter	NOL, NOE or NOD, "Check Position" and "Check Selection Parameter#" in subtypes 5, 7 and 9
3	Change Maximum Reporting Delay	
4	Clear Monitoring List	5
5	Add Parameters to Monitoring List	4 and/or 6
6	Delete Parameters from Monitoring List	5
7	Modify Parameter Checking Information	
8	Report Current Monitoring List	9
9	Current Monitoring List Report	8
10	Report Current Parameters Out-of-limit List	11
11	Current Parameters Out-of-limit List Report	10

---

## Large data transfer service

### 16.1 Scope

The large data transfer service is a supporting service used by the ground system or other services to transfer large service data units in a controlled manner. The choice to use the large data transfer service is made by the initiator of a given service data unit.

Some scenarios in which the large data transfer service can be used include:

- the on-board operations scheduling service is requested to report to the ground the current contents of the command schedule, whose size exceeds the maximum size of a single telemetry source packet;
- a very large area of on-board memory is loaded using telecommand packets with a mission-specific maximum size.

The large data transfer service provides a common transfer mechanism for all services and avoids the proliferation of service-specific solutions. A service data unit is passed to the large data transfer service which splits it into parts and transmits each part within a single telemetry source packet (on-board to ground) or a single telecommand packet (ground to on-board).

Different operational requirements can apply to the data transfer, including:

- a. the capability to determine when the transfer is complete;
- b. the capability to identify and selectively re-transmit those parts of the data which are lost during a transfer;
- c. the capability to pass the data parts in order, and without duplication, to the recipient.

This service satisfies the concepts described in subclause of this Standard.

Any number of application processes may provide a single instance of the large data transfer service.

Telemetry source packet and telecommand packet relating to the large data transfer service shall be denoted by Service Type = 13.

## 16.2 Service concept

### 16.2.1 Introduction

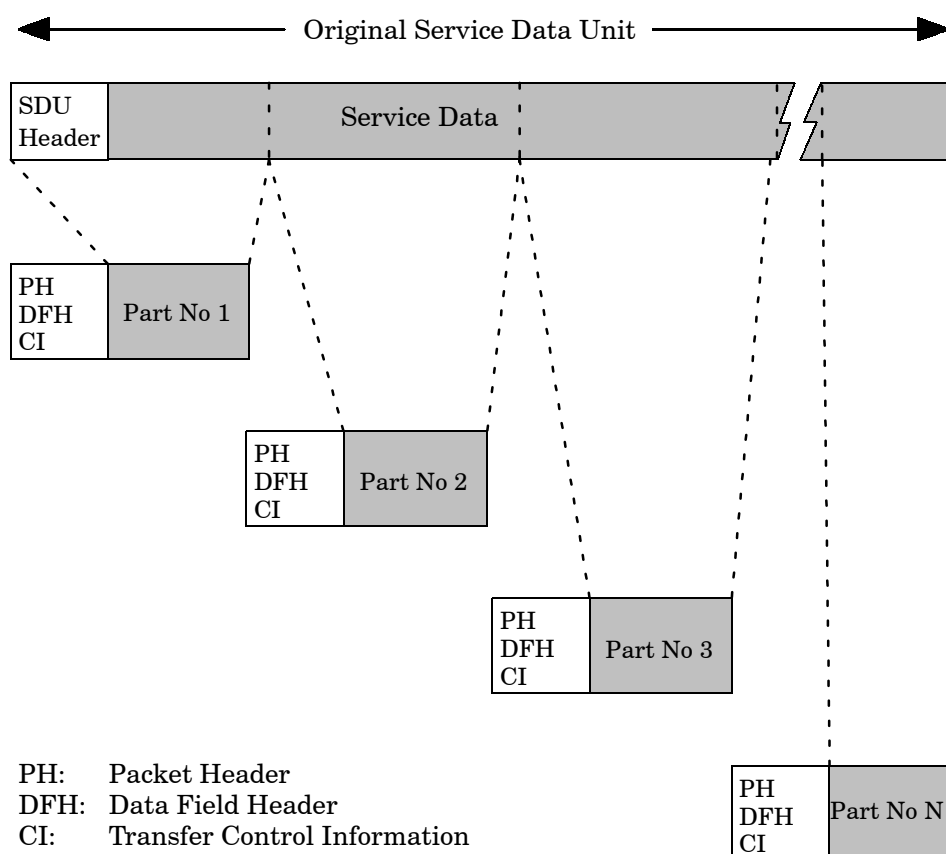
The large data transfer service shall provide two distinct but symmetrical operations: a large data downlink and a large data uplink.

There are similarities between the two operations; the transfer of a large data unit implies the simultaneous use of a sending sub-service on-board and of a receiving sub-service on the ground or vice-versa. However, the emphasis of the service specification in this Standard is on the sub-service to be provided on-board.

In the remainder of this subclause, the text relates to the downlink of a service data unit and the text specific to the uplink case is put in parentheses.

The basic concept for the large data downlink (large data uplink) protocol shall be as follows: when a service data unit or portion of it is passed to the sending sub-service provider of the large data transfer service, it shall perform the downlink (uplink) in the following manner:

- a. It shall split the original Service Data Unit (SDU) in fixed size parts and send the parts one by one, and in order. Each part shall be transmitted within a single telemetry source packet (telecommand packet) as illustrated in Figure 7.



**Figure 7: The splitting of a service data unit in parts to be downlinked (uplinked)**

- b. The sending and receiving sub-service providers shall use the same definition of the size of the parts (Part Size). The parts shall have equal size (except possibly the last part), which shall be less than the maximum size of the source data field of a telemetry source packet (application data field of a telecommand packet).



### 16.2.2 Service data unit to be downlinked (uplinked)

For the large data downlink, the service data unit which an on-board service provider passes to the sending sub-service provider shall consist of either a telemetry source packet or an “extended” telemetry source packet.

For the large data uplink, the service data unit which the ground segment passes to the on-board receiving sub-service provider shall consist of either a telecommand packet or an “extended” telecommand packet.

In the remainder of this subclause, depending on the type of transfer, downlink or uplink:

- the term “packet” denotes either a telemetry source packet or a telecommand packet;
- the term “extended packet” denotes either an “extended” telemetry source packet or an “extended” telecommand packet;
- the term “packet data” denotes the source data or the application data of a packet or extended packet.

The format of any service data unit to be downlinked (uplinked) shall be as follows:

Unit Type	Packet Header	Data Field Header	Packet Data
Enumerated	6 octets	Deduced	Any

#### Unit Type:

This parameter indicates whether what follows is a standard packet (value = 0) or an extended packet (value = 1).

An extended packet shall have the same structure as a packet, but the size of its packet data field can exceed the maximum size defined for a packet (i.e. if the total length of the packet data plus the data field header exceeds 65 535 octets).

For an extended packet, the value of the packet length field in the packet header shall be 0.

#### Packet Header:

This is a standard packet header, but the interpretation of the packet length field value shall differ for an extended packet (see above).

#### Data Field Header:

This is a data field header as defined in this Standard for a telemetry source packet or a telecommand packet.

#### Packet Data:

This is the source data of a large service report or the application data of a large service request.

In addition, the sending sub-service provider shall manage a “Large Data Unit ID”, which unambiguously identifies the service data unit to be transferred. This Large Data Unit ID is used by the sub-service providers of the large data transfer service on-board and on the ground, to distinguish the parts of this service data unit from the parts of other service data units for the case where there are simultaneous, overlapping, large data transfers originating from, or received by, the same on-board application process.

The detailed specification of the mechanism used to pass large data units between an on-board service provider and a large data transfer service provider is beyond the scope of this Standard. However, depending on the storage capacity available

to the large data transfer service provider or on design constraints, the following approaches can be used.

- a. For a downlink operation, the data may be passed by the sending service provider to the on-board large data transfer service provider either as a complete service data unit or as a set of data blocks (in which case the large data transfer service provider informs the sending service provider when the next block may be passed or when a downlink problem occurs). The service data unit can even be available in a storage or memory area directly accessible by the large data transfer service provider.
- b. For an uplink operation, the data received by the on-board large data transfer service provider may be passed to the destination service provider either as a complete data unit or as a set of data blocks or it can even be written directly into a storage or memory area accessible by the destination service provider.

### 16.2.3 Large data downlink (uplink) protocol

The following downlink (uplink) protocol aspects complement the basic protocol concept defined in subclause 16.2.1.

- a. The sending sub-service provider can have an attribute (window size) defining the number of parts which may be downlinked prior to receiving an acknowledgement from the receiving sub-service provider.

If this attribute is defined, then at no time should the sequence number of the next part to be downlinked (uplinked) exceed the sequence number of the last downlinked part which has been successfully acknowledged by the receiving sub-service provider by more than the value of "Window Size". In other words, a part may only be sent if it falls within the sliding window defined by the last acknowledged part and the window size.

Several window sizes can be used by a given large data transfer service provider.

**EXAMPLE** A large window size can be used for the memory management service and a smaller window size for other services.

If a sliding window is not used, then the sending sub-service provider shall send the parts of the service data unit at the highest frequency supported under the prevailing operational constraints.

- b. When a receiving sub-service provider receives an error-free part, this shall become the "last successfully received part" if there is no discontinuity between this part and the previous "last successfully received part".

If a sliding window is used, then no later than when the sequence counter of the last received part reaches the boundary of the sliding downlink window shall the receiving end send a notification of the "last successfully received part". This enables the sending sub-service provider to slide the downlink window.

If a sliding window is not used, the notification of successful reception shall only be provided on completion of the service data unit reception.

In both situations, this ensures that the sending sub-service provider can inform the originating service provider (the ground system) of the success or failure of the downlink (uplink). Only when the destination application on the ground (destination service provider on-board) has received the complete service data unit should notification of successful reception of the final part be uplinked (downlinked) by the receiving end of the large data transfer service.

- c. When the sending sub-service provider has sent the last part which can be sent in the current sliding window or the last part of the service data unit, it shall initiate an "acknowledgement timer". If it does not receive a notification of "last successfully received part" or an abort notification within a specified

timeout interval, then it shall assume that the downlink (uplink) has failed and shall notify the originating service provider (the ground system) accordingly.

- d. When the first, or an intermediate, part (which can be a re-transmitted part) is received by a receiving sub-service provider, a “reception timer” shall be initiated. If the receiving sub-service provider does not receive a subsequent part within a specified timeout interval, then it shall assume that the downlink (uplink) has failed and the reception activity shall be locally aborted. The destination application on the ground (destination service provider on-board) shall be notified accordingly.
- e. If the receiving sub-service provider receives an erroneous part or detects a gap in the reception sequence it shall add the erroneous or missing part(s) to the set of “failed parts”. Depending on the implementation, this information may be used by the receiving sub-service provider to automatically request the re-transmission of these parts or simply to inform the ground system (the destination service provider) of the errors in the transferred data.
- f. If a sliding window is used, then no later than when the sequence counter of the last received part reaches the boundary of the sliding window shall the receiving sub-service provider send notifications of the failed parts.

The sending sub-service provider shall then re-transmit (though not necessarily immediately) the notified failed parts. The successful re-transmission of a failed part shall update the set of failed parts and can determine a new “last successfully received part”.

If a sliding window is not used, notification of failed parts is unnecessary. However, if the receiving sub-service provider does send such a notification, then the sending sub-service provider shall re-transmit the failed parts.

- g. At any time during the downlink (uplink) activity, the originating (destination) service provider or the ground segment can abort the large data transfer operation.

If the sending sub-service provider can perform several overlapping data transfers, there shall be no restriction on the order in which it sends or re-sends the parts of the different data transfers other than the restrictions defined above.

## 16.3 Service requests and reports

### 16.3.1 Introduction

For each data downlink service request, there is a corresponding data uplink service report with an identical structure. For each data downlink service report, there is a corresponding data uplink service request with an identical structure.

### 16.3.2 Transferring the first part of a service data unit

The report to downlink and the request to uplink the first part of a large service data unit shall be:

#### First Downlink Part Report (13,1)

Telemetry source packet, source data:

#### Accept First Uplink Part (13,9)

Telecommand packet, application data:

Large Data Unit ID	Sequence Number	Service Data Unit Part
Enumerated	Unsigned Integer	Fixed OctetString

← Optional →

**Large Data Unit ID:**

This uniquely identifies the service data unit which is the subject of the transfer.

This field shall be systematically omitted if the service provider only supports the downlink (or uplink) of one service data unit at a time. This applies to all requests and reports in this subclause.

**Sequence Number:**

This is a unique identification for this part of the service data unit. By convention, the sequence number of the first part shall be 1. The sequence number shall then be incremented by one for each part which is subsequently transferred.

**Service Data Unit Part:**

This is the first part of the service data unit. Its size shall always be equal to the part size in use.

**Downlink:** The on-board sending sub-service provider shall use this report to send the first part of the data to be downlinked.

**Uplink:** The ground segment shall use this request to send the first part of the data to be uplinked.

On reception of the part, the receiving sub-service provider shall update its context and initiate a reception timer.

**16.3.3 Transferring an intermediate part of a service data unit**

The report to downlink and the request to uplink an intermediate part of a large service data unit shall be:

**Intermediate Downlink Part Report (13,2)**

Telemetry source packet, source data:

**Accept Intermediate Uplink Part (13,10)**

Telecommand packet, application data:

Same as for (13,1) and (13,9).

The size of an intermediate part of the service data unit shall always be equal to the part size in use.

**Downlink:** The sending sub-service provider shall use this report to send an intermediate part of the data to be downlinked.

**Uplink:** The ground segment shall use this request to send an intermediate part of the data to be uplinked.

The sending sub-service provider shall initiate an acknowledgement timer if this part is the last part which can be sent in the current sliding window.

On reception of the part, the receiving sub-service provider shall update its context and initiate a reception timer.

**16.3.4 Transferring the last part of a service data unit**

The report to downlink and the request to uplink the last part of a large service data unit shall be:

**Last Downlink Part Report (13,3)**

Telemetry source packet, source data:

**Accept Last Uplink Part (13,11)**

Telecommand packet, application data:

Same as for (13,1) and (13,9).

The size of the last part of a service data unit shall be less than or equal to the part size in use.

Downlink: The sending sub-service provider shall use this report to send the last part of the data to be downlinked.

Uplink: The ground segment shall use this request to send the last part of the data to be uplinked.

The sending sub-service provider shall always initiate an acknowledgement timer.

On reception of the part, the receiving sub-service provider shall update its context and immediately notify the sending sub-service provider of what is the “last successfully received part” at this point.

### 16.3.5 Re-transferring a part of a service data unit

The report to re-downlink and the request to re-uplink a part of a large service data unit whose re-transmission has been requested by the receiving end shall be:

#### Repeated Part Report (13,7)

Telemetry source packet, source data:

#### Accept Repeated Part (13,12)

Telecommand packet, application data:

Same as for (13,1) and (13,9).

The sequence number shall be the same as the sequence number used for the initial transfer.

Downlink: The sending sub-service provider shall use this report to re-transmit a part of the data which was not properly received by the ground system and which was requested for re-transmission.

Uplink: The ground segment shall use this request to re-transmit a part of the data which was not properly received by the on-board receiving sub-service provider.

The sending sub-service provider shall initiate an acknowledgement timer if this part is the last part which can be sent in the current window.

On reception of the part, the receiving sub-service provider shall update its context and initiate a reception timer.

### 16.3.6 Transfer abort initiated by the sending end

The report (during a downlink operation) and the request (during an uplink operation) to notify a transfer abort initiated by the sending end shall be:

#### Downlink Abort Report (13,4)

Telemetry source packet, source data:

#### Abort Reception of Uplinked Data (13,13)

Telecommand packet, application data:

Large Data Unit ID	Reason Code
Enumerated	Enumerated
<div style="display: flex; justify-content: space-around; align-items: center;"> <span>← Optional →</span> <span>← Optional →</span> </div>	

Reason Code:

This indicates the reason for the transfer abort. The values it may take are mission-specific.

Downlink: The sending sub-service provider shall use this report to indicate to the ground system that the large data downlink operation has been aborted (e.g. by the originating service provider).

Uplink: The ground segment shall use this request to indicate to the on-board receiving sub-service provider that the large data reception activities should be aborted. The on-board receiving sub-service provider shall inform the destination service provider of the abort.

### 16.3.7 Acknowledging the successful reception up to a part

The report (during an uplink operation) and the request (during a downlink operation) to acknowledge the successful reception of the large service data unit up to a specified part shall be:

#### Uplink Reception Acknowledgement Report (13,14)

Telemetry source packet, source data:

#### Downlink Reception Acknowledgement (13,5)

Telecommand packet, application data:

Large Data Unit ID	Sequence Number
Enumerated	Unsigned Integer

|<----- Optional ----->|

The receiving sub-service provider shall use this notification to indicate to the sending sub-service provider that it has successfully received all parts of the large service data unit up to and including the part with the indicated sequence number.

If a sliding window is used, this enables it to be moved forward and the sending activity to be resumed.

### 16.3.8 Notifying which parts have not been properly received

The report (during an uplink operation) and the request (during a downlink operation) to notify the sending sub-service provider that specified parts were not received or were erroneously received shall be:

#### Unsuccessfully Received Parts Report (13,15)

Telemetry source packet, source data:

#### Repeat Parts (13,6)

Telecommand packet, application data:

Large Data Unit ID	N	Sequence Number
Enumerated	Unsigned Integer	Unsigned Integer

|<----- Optional ----->|      |<----- Repeated N times ----->|

On reception of this notification, the sending sub-service provider shall record the information. It shall use this information to re-transmit the parts if a sliding window is used or if it is an on-board sending sub-service provider. A ground system sending sub-service provider may decide not to re-transmit the parts if a sliding window is not used.

### 16.3.9 Transfer abort initiated by the receiving end

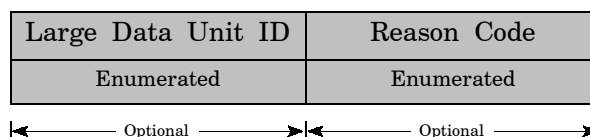
The report (during an uplink operation) and the request (during a downlink operation) shall be:

#### Reception Abort Report (13,16)

Telemetry source packet, source data:

#### Abort Downlink (13,8)

Telecommand packet, application data:



Reason Code:

This indicates the reason for the transfer abort. The values it may take are mission-specific.

On reception of this notification, the sending sub-service provider shall inform the originating service provider (the ground system) that the downlink (uplink) operation shall be aborted.

## 16.4 Capability sets

### 16.4.1 Minimum capability sets

Each sub-service (sending and receiving) is optional, but a given implementation of the large data transfer service shall support at least one sub-service.

The minimum capability set for the sending sub-service (data downlink operation) shall consist of the service requests and reports given in Table 27.

**Table 27: Summary of sending sub-service (downlink) minimum capabilities**

Subtype	Service request, report or capability
1	First Downlink Part Report
2	Intermediate Downlink Part Report
3	Last Downlink Part Report
4	Downlink Abort Report
5	Downlink Reception Acknowledgement
8	Abort Downlink

The minimum capability set for the receiving sub-service (data uplink operation) shall consist of the service requests and reports given in Table 28.

**Table 28: Summary of receiving sub-service (uplink) minimum capabilities**

Subtype	Service request, report or capability
9	Accept First Uplink Part
10	Accept Intermediate Uplink Part
11	Accept Last Uplink Part
13	Abort Reception of Uplinked Data
14	Uplink Reception Acknowledgement Report
16	Reception Abort Report

## 16.4.2 Additional capability sets

The use of a “Sliding Window” and the automatic re-transmission of lost or erroneous parts are optional.

The selective re-transmission of lost or erroneous parts without using a sliding window is optional.

The sending sub-service provider (data downlink operation) can implement the additional capabilities given in Table 29.

**Table 29: Summary of sending sub-service (downlink) additional capabilities**

Subtype	Service request, report or capability	Requires subtype
6	Repeat Parts	7
7	Repeated Part Report	6
	Use of a “Sliding Window”	7 and 6

The receiving sub-service provider (data uplink operation) can implement the additional capabilities given in Table 30.

**Table 30: Summary of receiving sub-service (uplink) additional capabilities**

Subtype	Service request, report or capability	Requires subtype
12	Accept Repeated Part	15
15	Unsuccessfully Received Parts Report	12
	Use of a “Sliding Window”	12 and 15



---

## Packet forwarding control service

### 17.1 Scope

The packet forwarding control service provides the capability to control the forwarding to the ground of telemetry source packets issued by on-board services. This service satisfies the packet forwarding concepts described in subclause 4.12 of this Standard.

Any number of on-board application processes may provide a single instance of the packet forwarding control service.

Telemetry source packets and telecommand packets relating to the packet forwarding control service shall be denoted by Service Type = 14.

### 17.2 Service concept

Two options exist for the location of this service, either it may be provided at the level of the originating application process or it may be provided by a centralised application process which is responsible for routing packets on the downlink.

The packet forwarding control service shall maintain the knowledge of which packets can be transmitted to the ground system.

For a given application process, the forwarding of packets can be “enabled” and “disabled” at the level of:

- a. a type of packet;
- b. a subtype of packet;
- c. a housekeeping packet definition, a diagnostic packet definition or an event report definition (see clauses 8 and 10).

The forwarding of packets with a given type and subtype shall be “enabled” if and only if the packet type and the packet subtype are both enabled (i.e. if the type is in the set of enabled types and the subtype is in the set of enabled subtypes for that type).

In addition, the forwarding of housekeeping (or diagnostic or event report) packets shall be “enabled” if and only if the packet type, the packet subtype and the housekeeping packet definition (or the diagnostic packet definition or the event report definition) are all enabled.

Conceptually, this is as if each such packet definition has three independent controlling attributes (at type level, at subtype level and at packet structure

identification level) whose values determine the forwarding status of the packets in accordance with Table 31.

**Table 31: Decision table for the forwarding status of a packet**

Type	Subtype	Identification (SID/RID)	Forwarding status
D(isabled)	E(nabled)	E	D
D	D	E	D
D	E	D	D
D	D	D	D
E	E	E	E
E	D	E	D
E	E	D	D
E	D	D	D

The packet forwarding control service shall maintain the forwarding status for all types and subtypes of packet (per application process, in the case of the centralised option) and shall report the list of enabled packets to the ground system on request.

Similarly, the packet forwarding control service shall maintain the forwarding status for all housekeeping packet definitions, diagnostic packet definitions and event report definitions (per application process, in the case of the centralised option) and shall report the list of enabled packets to the ground system on request.

## 17.3 Service requests and reports

### 17.3.1 Controlling the forwarding of specified telemetry source packets

The requests to enable or disable the forwarding of telemetry source packets of specified type and subtype shall be:

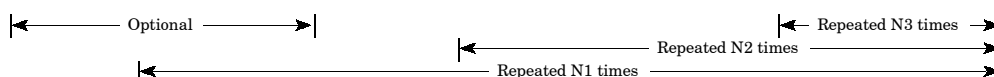
#### Enable Forwarding of Telemetry Source Packets (14,1)

Telecommand packet, application data:

#### Disable Forwarding of Telemetry Source Packets (14,2)

Telecommand packet, application data:

N1	Application Process ID	N2	Type	N3	Subtype
Unsigned Integer	Enumerated	Unsigned Integer	Enumerated	Unsigned Integer	Enumerated



N1:

The number of application processes that follow.

This field shall be systematically omitted if the packet forwarding service is provided by the originating application process.

**Application Process ID:**

The identification of the originating application process of the packets to be forwarded.

This field shall be systematically omitted if the packet forwarding service is provided by the originating application process.

**Type:**

The telemetry source packet service type.

**Subtype:**

The telemetry source packet service subtype for the specified service type.

When the service provider receives this request one of the following shall occur:

- a. If  $N2 = 0$  and “enable” is requested, all types of telemetry source packets from the corresponding application process shall be placed in the set of enabled types.
- b. If  $N2 = 0$  and “disable” is requested, all types of telemetry source packets from the corresponding application process shall be removed from the set of enabled types and all sets of enabled subtypes shall be cleared.
- c. If  $N2 > 0$  and  $N3 = 0$  and “enable” is requested, the specified types of telemetry source packets from the corresponding application process shall be added to the set of enabled types.
- d. If  $N2 > 0$  and  $N3 = 0$  and “disable” is requested, the specified types of telemetry source packets from the corresponding application process shall be removed from the set of enabled types and their sets of enabled subtypes shall be cleared.
- e. If  $N2 > 0$  and  $N3 > 0$  and “enable” is requested, the specified subtypes of telemetry source packets from the corresponding application process shall be added to the set of enabled subtypes for the specified type.
- f. If  $N2 > 0$  and  $N3 > 0$  and “disable” is requested, the specified subtypes of telemetry source packets from the corresponding application process shall be removed from the set of enabled subtypes for the specified type.

NOTE 1 If  $N2 > 1$  then there can be a mixture of empty ( $N3 = 0$ ) and non-empty ( $N3 > 0$ ) arrays.

NOTE 2 These requests do not change the forwarding status at the level of the SID/RID.

An error shall be flagged if a specified type or subtype of packet is not generated by the corresponding application process. However, the processing of the other types and subtypes in the request shall be unaffected.

### 17.3.2 Reporting the list of enabled telemetry source packets

The request to report the list of telemetry source packet types and subtypes with an “enabled” forwarding status shall be:

#### Report Enabled Telemetry Source Packets (14,3)

Telecommand packet, application data: None

When this request is received, the enabled types and subtypes of telemetry source packet shall be determined and a report shall be generated.

**Enabled Telemetry Source Packets Report (14,4)**

Telemetry source packet, source data:

N1	Application Process ID	N2	Type	N3	Subtype
Unsigned Integer	Enumerated	Unsigned Integer	Enumerated	Unsigned Integer	Enumerated



If  $N2 = 0$ , no type nor subtype of packet from the corresponding application process is enabled.

If  $N2 > 0$ , the specified types of packet from the corresponding application process are enabled.

If  $N3 = 0$  for a type of packet, none of the subtypes of this type from the corresponding application process are enabled.

If  $N3 > 0$  for a type of packet, the specified subtypes of this type from the corresponding application process are enabled.

NOTE If  $N2 > 1$  then there can be a mixture of empty ( $N3 = 0$ ) and non-empty ( $N3 > 0$ ) arrays.

**17.3.3 Controlling the forwarding of specified housekeeping packets**

The requests shall be:

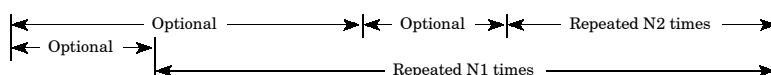
**Enable Forwarding of Housekeeping Packets (14,5)**

Telecommand packet, application data:

**Disable Forwarding of Housekeeping Packets (14,6)**

Telecommand packet, application data:

N1	Application Process ID	N2	SID
Unsigned Integer	Enumerated	Unsigned Integer	Enumerated



N2:

The number of housekeeping packet definitions to be enabled or disabled.

$N1$  and  $N2$  shall be systematically omitted if the service only supports the control of one housekeeping packet at a time.

SID:

The Structure Identifier identifying a housekeeping packet definition.

When this request is received, the specified housekeeping packet definitions from the corresponding application process shall be added to (or removed from) the set of enabled housekeeping packet definitions (depending on whether it is an "enable" or "disable" request).

An error shall be flagged if a specified housekeeping SID is not defined for the application process. However, this shall not affect the processing of the other housekeeping SIDs in the request.

### 17.3.4 Reporting the list of enabled housekeeping packets

The request to report the list of housekeeping packet definitions with an “enabled” forwarding status shall be:

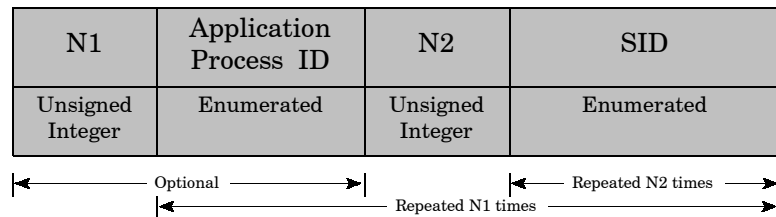
#### Report Enabled Housekeeping Packets (14,7)

Telecommand packet, application data: None.

When this request is received, the list of enabled housekeeping packet definitions shall be determined and a report shall be generated.

#### Enabled Housekeeping Packets Report (14,8)

Telemetry source packet, source data:



### 17.3.5 Controlling the forwarding of specified diagnostic packets

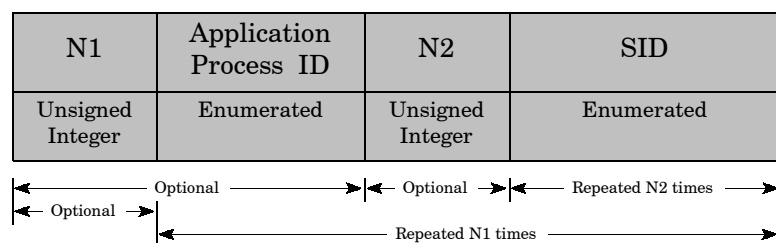
The requests shall be:

#### Enable Forwarding of Diagnostic Packets (14,9)

Telecommand packet, application data:

#### Disable Forwarding of Diagnostic Packets (14,10)

Telecommand packet, application data:



N2:

The number of diagnostic packet definitions to be enabled or disabled.

N1 and N2 shall be systematically omitted if the service only supports the control of one diagnostic packet at a time.

SID:

The Structure Identifier identifying a diagnostic packet definition.

When this request is received, the specified diagnostic packet definitions from the corresponding application process shall be added to (or removed from) the set of enabled diagnostic packet definitions (depending on whether it is an “enable” or “disable” request).

An error shall be flagged if a specified diagnostic SID is not defined for the application process,. However, this shall not affect the processing of the other diagnostic SIDs in the request.

### 17.3.6 Reporting the list of enabled diagnostic packets

The request to report the list of diagnostic packet definitions with an “enabled” forwarding status shall be:

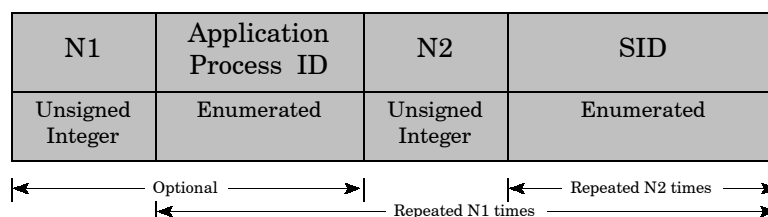
#### Report Enabled Diagnostic Packets (14,11)

Telecommand packet, application data: None.

When this request is received, the list of enabled diagnostic packet definitions shall be determined and a report shall be generated.

#### Enabled Diagnostic Packets Report (14,12)

Telemetry source packet, source data:



### 17.3.7 Controlling the forwarding of specified event report packets

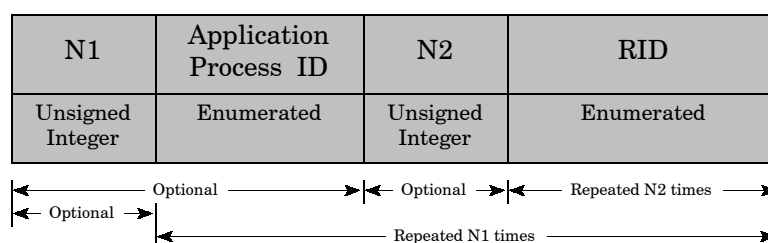
The requests to enable or disable the forwarding of specified event report packets shall be:

#### Enable Forwarding of Event Report Packets (14,13)

Telecommand packet, application data:

#### Disable Forwarding of Event Report Packets (14,14)

Telecommand packet, application data:



N2:

The number of event report packet definitions to be enabled or disabled.

N1 and N2 shall be systematically omitted if the service only supports the control of one event report packet at a time.

RID:

The report identifier identifying an event report packet definition.

When this request is received, the specified event report packet definitions from the corresponding application process shall be added to (or removed from) the set of enabled event report packet definitions (depending on whether it is an “enable” or “disable” request).

An error shall be flagged if a specified event RID is not defined for the application process. However, this shall not affect the processing of the other event RIDs in the request.

### 17.3.8 Reporting the list of enabled event report packets

The request to report the list of event report packet definitions with an “enabled” forwarding status shall be:

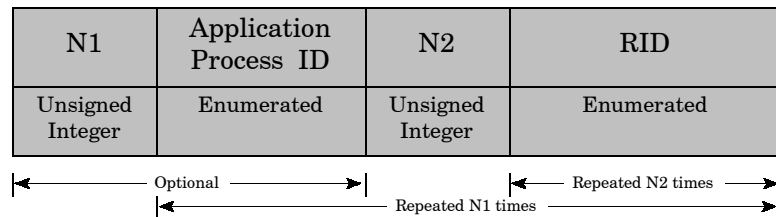
#### Report Enabled Event Report Packets (14,15)

Telecommand packet, application data: None

When this request is received, the list of enabled event report packet definitions shall be determined and a report shall be generated.

#### Enabled Event Report Packets Report (14,16)

Telemetry source packet, source data:



## 17.4 Capability sets

### 17.4.1 Minimum capability set

The capability to control the forwarding of a given type of definition of a status reporting packet (i.e. a housekeeping, diagnostic or event report packet) shall only be provided if the corresponding type of definition is generated.

Thus, the minimum capability set shall consist of the service requests and reports given in Table 32.

**Table 32: Summary of packet forwarding control service minimum capabilities**

Subtype	Service request, report or capability
1	Enable Forwarding of Telemetry Source Packets
2	Disable Forwarding of Telemetry Source Packets

### 17.4.2 Additional capability sets

A packet forwarding control service can implement the additional capabilities given in Table 33.

**Table 33: Summary of packet forwarding control service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
3	Report Enabled Telemetry Source Packets	4
4	Enabled Telemetry Source Packets Report	3
5	Enable Forwarding of Housekeeping Packets	6
6	Disable Forwarding of Housekeeping Packets	5
7	Report Enabled Housekeeping Packets	8, 5 and 6
8	Enabled Housekeeping Packets Report	7
9	Enable Forwarding of Diagnostic Packets	10
10	Disable Forwarding of Diagnostic Packets	9
11	Report Enabled Diagnostic Packets	12, 9 and 10
12	Enabled Diagnostic Packets Report	11
13	Enable Forwarding of Event Report Packets	14
14	Disable Forwarding of Event Report Packets	13
15	Report Enabled Event Report Packets	16, 13 and 14
16	Enabled Report Event Packets Report	15



---

## On-board storage and retrieval service

### 18.1 Scope

The on-board storage and retrieval service is a supporting service used by other on-board services to selectively store the service reports which they generate in order to give the ground system the capability to request the retrieval and downlink of the selectively stored data.

The on-board storage and retrieval service consists of two parts:

- a. a packet selection sub-service: selection and transfer of telemetry source packets for storage in packet stores;
- b. a storage and retrieval sub-service: storage and retrieval of telemetry source packets from packet stores.

This service satisfies the concepts described in subclause 4.11 of this Standard.

Any number of on-board application processes may provide a single instance of the on-board storage and retrieval service.

Telemetry source packets and telecommand packets relating to the on-board storage and retrieval service shall be denoted by Service Type = 15.

### 18.2 Service concept

#### 18.2.1 General

The on-board storage and retrieval service shall be used if a mission uses on-board storage of telemetry source packets with the capability to selectively store the packet types and subtypes in different packet stores.

**EXAMPLE** For missions with intermittent coverage, packets of high operational significance (anomaly report packets) which are generated during a period of non-coverage can be stored in a dedicated packet store so that they can be retrieved first during the next period of coverage.

The service can also be used by application processes to provide a “lost packet recovery” capability, protecting against temporary spacelink outages, by systematically placing event-driven service reports in short-term circular packet stores.

One or more packet types and subtypes generated by one or more application processes can be selected for storage in a given packet store managed by a given application process.

The application process that selects which telemetry source packets are sent for storage, either to another application process or to its own storage and retrieval sub-service, shall provide the packet selection sub-service.

The application process responsible for managing the packet store(s) shall provide the storage and retrieval sub-service.

The service concept includes the possibility that the packet selection sub-service is always provided by the same application process that provides the storage and retrieval sub-service (rather than by the application process that generates the packets). This leads to an implementation where all selection, storage and retrieval is performed by a centralised application process that (by design) has access to all telemetry packets.

A packet store is uniquely identified by a "Store ID". The packet selection sub-service shall know implicitly (e.g. by virtue of its design) which application process provides the storage and retrieval sub-service that manages a packet store to which it transfers packets.

The definition of the storage selection used by a given packet selection sub-service can either be predefined or changeable by the ground system.

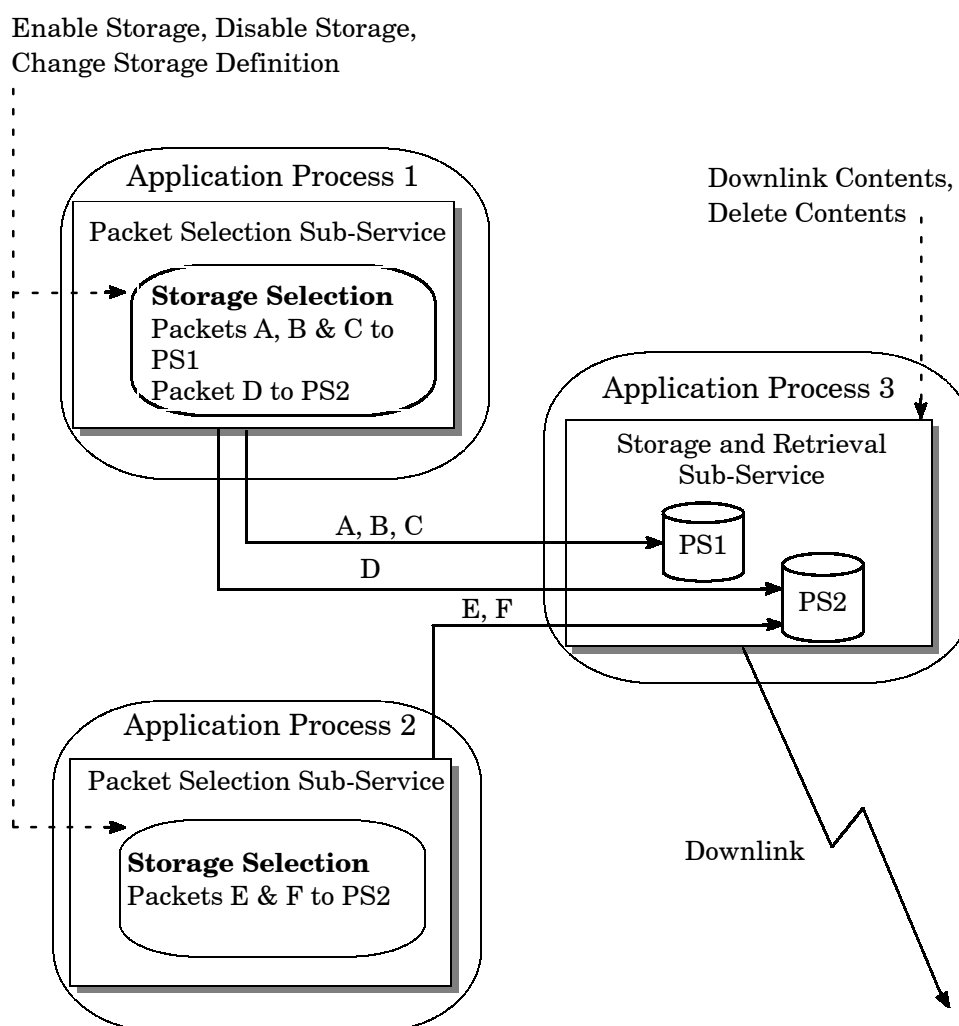
Packets shall be ordered according to their time of arrival at the storage and retrieval sub-service.

Telemetry source packets stored in a packet store shall be downlinked on request. The downlink request may specify that all, or a subset (e.g. within a specified packet range), of the stored packets are downlinked.

Two options exist for downlinking stored telemetry source packets:

- a. They can be downlinked in the same virtual channel as real-time data, in which case, they shall be wrapped in a "real-time" packet envelope, where the envelope bears the service type and subtype of the on-board storage and retrieval service.
- b. They can be downlinked in a virtual channel that is dedicated to retrieval, in which case they shall be downlinked exactly as originally received by the packet store.

Only one of these options shall exist (by design) for a given storage and retrieval service.



**Figure 8: An example of a storage and retrieval approach**

The storage and retrieval sub-service can (optionally) time stamp each telemetry source packet it receives with the time of reception/storage. Where this service is offered, the ground system can request the downlink of packets from a packet store which were stored after, before, or between specified times.

When downlinking using the real-time virtual channel, the packet storage and retrieval sub-service may use the large data transfer service to downlink the requested packets. The service data unit containing the data to be downlinked shall be constructed by retrieving from the packet store those packets which meet the selection criteria specified by the ground system. The position within the service data unit shall correspond to the packet store reception and storage order.

The specification of the mechanism by which a given storage and retrieval sub-service receives packets to be stored from the services of application processes is beyond the scope of this Standard.

### 18.2.2 Packet store types

Each packet store has attributes, which are accessed by the storage and retrieval sub-service which manages it, and which shall indicate:

- whether the storage strategy is circular or bounded;
- the maximum size of the packet store;
- the boundaries of the current set of stored packets (e.g. the oldest and newest packets).

When a circular packet store is full, any subsequently received packet shall overwrite partly or fully the oldest packet(s) in the list. A packet store used to provide a “lost packet recovery” capability shall be circular in nature.

When a bounded packet store is full, any subsequently received packet shall be ignored. The contents of the packet store, or at least the oldest part of it, shall be explicitly deleted in order to free storage space and to enable the resumption of storage of new packets. This type of deletion (i.e. erasing the contents of a packet store) can also be performed on a circular packet store.

The performance of a storage and retrieval sub-service may not be able to guarantee storage of packets without loss. Moreover, the protocol used for passing packets to the storage and retrieval sub-service is not always reliable. If packets cannot be stored, the service need not record information about the gaps in the sequence of stored packets. These gaps can be detected on the ground following the downlink of the contents of a packet store.

When the contents of a circular packet store are being downlinked, the storage and retrieval sub-service may not be able to guarantee that packets are downlinked at a higher rate than that at which new packets are arriving at the packet store. In this case, priority shall be given to the downlink activity, i.e. the overwriting pointer shall always stay behind the retrieval pointer. In other words, new packets shall not be placed in the packet store until the packets being retrieved have been downlinked.

Such a strategy is appropriate where the spacelink is being used to downlink both real-time packets and packets retrieved from the packet store. The newest packets shall be received directly by the ground system, whilst the oldest packets in the packet store shall not be overwritten until downlinked.

In addition to housekeeping data that can be reported on a regular basis, catalogue information should be maintained by the storage and retrieval sub-service for each packet store, which shall be reported to ground on request. This catalogue information shall include: identification of the oldest and newest packets in the packet store, the percentage of filling of the packet store, the percentage of the packet store contents that has not yet been downlinked.

## 18.3 Service requests and reports

### 18.3.1 Controlling the storage in specified packet stores

The requests shall be:

#### Enable Storage in Packet Stores (15,1)

Telecommand packet, application data:

#### Disable Storage in Packet Stores (15,2)

Telecommand packet, application data:

N	Store ID
Unsigned Integer	Fixed CharString
Optional	Repeated N times

N:

The number of packet stores to be controlled. By convention, N = 0 shall mean “all packet stores”.

This field shall be systematically omitted if the service only supports one packet store (this is equivalent to having N=0).

**Store ID:**

An on-board packet store is uniquely identified by a “Store ID”. This shall be a character string which can explicitly or implicitly indicate, for example, an access path to a physical on-board recording device or file. The meaning and internal structure of the Store ID are beyond the scope of this Standard.

When the packet selection sub-service provider receives this request, it shall start or stop (depending on whether it is an “enable” or “disable” request) sending the relevant packets to the application processes managing the specified packet stores.

**18.3.2 Modifying the definition of a storage selection criteria**

The storage selection definition used by an application process to send packets for storage in a given packet store shall consists of the identification of the type and subtype of the relevant packets. If the packets originate from an application process other than that which provides the packet selection sub-service, the originating application process shall also be specified. The capability shall exist to add specified packets to (or remove from) a storage selection definition.

The requests to modify the storage selection definition for a specified packet store shall be:

**Add Packets to Storage Selection Definition (15,3)**

Telecommand packet, application data:

**Remove Packets from Storage Selection Definition (15,4)**

Telecommand packet, application data:

Store ID	N1	Application Process ID	N2	Type	N3	Subtype
Fixed CharString	Unsigned Integer	Enumerated	Unsigned Integer	Enumerated	Unsigned Integer	Enumerated

Diagram illustrating the structure of the data fields:

- Store ID: Optional (indicated by a double-headed arrow below the field).
- N1: Optional (indicated by a double-headed arrow below the field).
- Application Process ID: Repeated N1 times (indicated by a double-headed arrow below the field).
- N2: Repeated N2 times (indicated by a double-headed arrow below the field).
- Type: Repeated N3 times (indicated by a double-headed arrow below the field).
- N3: Repeated N3 times (indicated by a double-headed arrow below the field).
- Subtype: Repeated N3 times (indicated by a double-headed arrow below the field).

**Store ID:**

This field shall be systematically omitted if the service only supports one packet store.

**Application Process ID:**

The identification of the originating application process of the packets to be stored.

This field shall be systematically omitted if the packet selection sub-service is provided by the originating application process.

**Type:**

A telemetry source packet service type.

**Subtype:**

A telemetry source packet service subtype of the specified service type.

When the packet selection sub-service provider receives this request one of the following shall occur:

- If  $N2 = 0$ , all types of telemetry source packet from the corresponding application process shall be added to, or removed from, the list of packet types to be stored in the specified packet store (depending on the type of request).

- b. If  $N2 > 0$  and  $N3 = 0$ , the specified types of telemetry source packet from the corresponding application process shall be added to (if not yet present), or removed from, the list of packet types to be stored in the specified packet store (depending on the type of request).
- c. If  $N2 > 0$  and  $N3 > 0$ , the specified subtypes of telemetry source packet from the corresponding application process shall be added to, or removed from, the list of packet subtypes (for the specified type) to be stored in the specified packet store (depending on the type of request).

The request shall have no effect for a packet type which is already in the list of packet types to be stored in the specified packet store (because all its subtypes are already selected for storage).

NOTE If  $N2 > 1$  then there can be a mixture of empty ( $N3 = 0$ ) and non-empty arrays ( $N3 > 0$ ).

The current content of the packet store shall not be affected by the request and, if storage is enabled, packets shall start or stop to be appended to the packet store immediately after the request is processed.

The creation, deletion or modification of packet stores, together with their attributes is beyond the scope of this Standard.

### 18.3.3 Reporting a storage selection definition

The request shall be:

#### Report Storage Selection Definition (15,5)

Telecommand packet, application data:



Store ID:

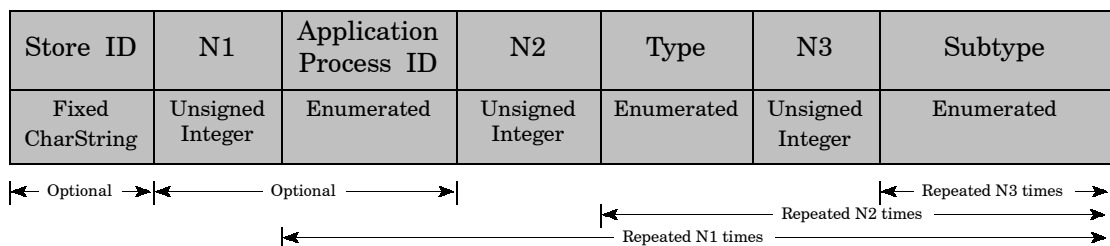
This field shall be systematically omitted if the service only supports one packet store.

When this request is received by the application process which provides the packet selection sub-service, the storage selection definition for the specified packet store shall be read and a report shall be generated.

NOTE Since the destination application process uses a local storage selection definition for selecting which of its packets to send to a packet store (which can be managed by another application process), what is downlinked might only represent a partial definition of the complete storage selection for the given packet store.

#### Storage Selection Definition Report (15,6)

Telemetry source packet, source data:



The Store ID field shall be systematically omitted if the service provider only supports one packet store.

The Application Process ID field shall be systematically omitted if the packet selection sub-service is provided by the originating application process.

If  $N2 = 0$ , no type nor subtype of packet from the corresponding application process is selected for storage.

If  $N2 > 0$ , the specified types of packet from the corresponding application process are selected for storage.

If  $N3 > 0$  for a type of packet, the specified subtypes of this type from the corresponding application process are selected for storage.

### 18.3.4 Downlinking the contents of a packet store for a specified packet range

The request shall be:

#### Downlink Packet Store Contents for Packet Range (15,7)

Telecommand packet, application data:

Store ID	Packet Set	Application Process ID 1	Source Sequence Count 1	Application Process ID 2	Source Sequence Count 2
Fixed CharString	Enumerated	Enumerated	Unsigned Integer	Enumerated	Unsigned Integer

← Optional →   ← Optional →   ← Optional →   ← Optional →

Store ID:

This field shall be systematically omitted if the service provider only supports one packet store.

Packet Set:

This indicates how the packet range is specified. If Packet Set is "All" (value = 0), the full contents of the packet store shall be downlinked, otherwise it is the set of packets received and stored:

- between the packets identified by (Application Process ID 1, Source Sequence Count 1) and (Application Process ID 2, Source Sequence Count 2) inclusive, if Packet Set is "Between" (value = 1);
- up to and including the packet identified by (Application Process ID 1, Source Sequence Count 1), if Packet Set is "Before" (value = 2);
- after and including the packet identified by (Application Process ID 1, Source Sequence Count 1), if Packet Set is "After" (value = 3).

This field shall be systematically omitted if the service provider only supports one way of specifying the packet range. The service provider may only support a subset of the Packet Set values.

Application Process ID 1, Source Sequence Count 1:

Application Process ID 2, Source Sequence Count 2:

The identifications of the packets defining the boundary(ies) of the range of packets to be downlinked. These shall be matched with the corresponding fields in the data field header of packets in the packet store.

The fields Application Process ID 1 and Source Sequence Count 1 shall be present if Packet Set is not "All". The fields Application Process ID 2 and Source Sequence Count 2 shall be present if Packet Set is "Between".

When this request is received by the application process which provides the storage and retrieval sub-service, the contents of the specified packet store falling within the specified packet range shall be downlinked. In this respect, "falling

within” refers to the time interval between the time of reception (by the storage and retrieval sub-service) of the first and last packets defining the packet set. Whatever the value of Packet Set, the retrieval shall end at the latest when the last packet stored at the time of reception of the request has been downlinked.

If a packet defining a lower boundary of the packet range is not in the packet store, the retrieval shall start with the last packet from the same application process before the missing packet (or the oldest stored packet if no earlier packet from that application process is in the packet store).

If a packet defining an upper boundary of the packet range is not in the packet store, the retrieval shall end with the first packet from the same application process after the missing packet (or the end of the packet store at the time of reception of the request).

Depending on the service design, the retrieved packets shall be either:

- downlinked directly into a dedicated virtual channel, i.e. exactly as received by the packet store; or
- placed in a report in the same sequence as their storage order and downlinked in the real-time virtual channel (using the large data transfer service, if appropriate). The format of this report shall be:

#### Packet Store Contents Report (15,8)

Telemetry source packet, source data:

N	TLM Packet
Unsigned Integer	Any TM

|← Repeated N times →|

TLM Packet:

A telemetry source packet of any type retrieved from the packet store.

### 18.3.5 Downlinking the contents of a packet store for a specified time period

The request shall be:

#### Downlink Packet Store Contents for Time Period (15,9)

Telecommand packet, application data:

Store ID	Time Span	Storage Time 1	Storage Time 2
Fixed CharString	Enumerated	Absolute Time	Absolute Time

|← Optional →| |← Optional →| |← Optional →| |← Optional →|

Store ID:

This field shall be systematically omitted if the service only supports one packet store.

Time Span:

This indicates how the packet range is specified. If Time Span is “All” (value = 0), the full contents of the packet store shall be downlinked, otherwise it is the set of packets whose storage times are:

- between Storage Time 1 and Storage Time 2 inclusive, if Time Span is “Between” (value = 1);
- less than or equal to Storage Time 1 if Time Span is “Before” (value = 2);



- (c) greater than or equal to Storage Time 1 if Time Span is “After” (value = 3).

This field shall be systematically omitted if the service provider only supports one way of specifying the packet range. The service provider may only support a subset of the Time Span values.

Storage Time 1, Storage Time 2:

The absolute time(s) defining the boundary(ies) of the range of packets to be downlinked. The format and length of these parameters shall be uniquely defined for the application process for which the request is destined.

Storage Time 1 shall be present if Time Span is not “All”. Storage Time 2 shall be present if Time Span is “Between”.

When this request is received by the application process which provides the storage and retrieval sub-service, the contents of the specified packet store falling within the specified packet range shall be downlinked. Whatever the value of Time Span, the retrieval shall end at the latest when the last packet stored at the time of reception of the request has been downlinked.

The packets shall be downlinked either directly (using a dedicated virtual channel) or by means of a Packet Store Contents Report, as described in subclause 18.3.4.

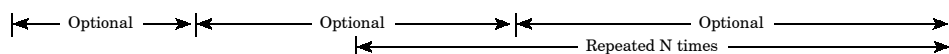
### 18.3.6 Deleting the contents of specified packet stores up to specified packets

The request shall be:

#### Delete Packet Stores Contents up to Specified Packets (15,10)

Telecommand packet, application data:

Deletion Set	N	Store ID	Application Process ID	Source Sequence Count
Enumerated	Unsigned Integer	Fixed CharString	Enumerated	Unsigned Integer



Deletion Set:

This indicates how the part of the packet store to be deleted is specified. If Deletion Set is “All” (value = 0) the full contents of the packet store shall be deleted. If Deletion Set is “Before” (value = 1), it shall be the set of packets up to the specified packet.

This field shall be systematically omitted if the service provider only supports one way of specifying the part of the packet store to be deleted.

N:

By convention, N = 0 shall mean “Delete the full contents of all packet stores”.

This field shall be systematically omitted if the service provider only supports one packet store (equivalent to having N=1).

Store ID:

This field shall be systematically omitted if the service provider only supports one packet store.

Application Process ID, Source Sequence Count:

This is the identification of the packet defining the upper boundary (inclusive) of the packet range to be deleted (only present if Deletion Set is

“Before”). These fields shall be matched with the corresponding fields in the data field header of packets in the packet store.

When this request is received by the application process which provides the storage and retrieval sub-service, the packets in the specified packet stores which have been stored before the specified packet shall be deleted. The deletion shall end at the latest when the last packets stored at the time of reception of the request have been deleted. If  $N = 0$ , the full contents of all packet stores shall be deleted.

If a packet defining the upper boundary of the packet range is not in the packet store, the deletion shall end with the last packet from the same application process before the missing packet (or nothing shall be deleted if no earlier packet from that application process is in the packet store).

While deletion from a packet store is in progress, packets sent for storage can be recorded in the freed space of the packet store.

### 18.3.7 Deleting the contents of specified packet stores up to a specified storage time

The request shall be:

#### Delete Packet Stores Contents up to Specified Storage Time (15,11)

Telecommand packet, application data:

End Time	N	Store ID
Absolute Time	Unsigned Integer	Fixed CharString
<div style="display: flex; justify-content: space-around; align-items: center;"> <span>← Optional →</span> <span>← Repeated N times →</span> </div>		

End Time:

The absolute time defining the upper boundary (inclusive) of the packet range to be deleted. The format and length of this parameter shall be uniquely defined for the application process for which the request is destined.

N:

By convention,  $N = 0$  shall mean “All packet stores”.

This field shall be systematically omitted if the service provider only supports one packet store (equivalent to having  $N=0$ ).

When this request is received by the application process which provides the storage and retrieval sub-service, the packets in the specified packet stores (all packet stores if  $N = 0$ ) which have a storage time earlier than or equal to the specified time shall be deleted. The deletion shall end at the latest when the last packets stored at the time of reception of the request have been deleted.

While deletion from a packet store is in progress, packets sent for storage can be recorded in the freed space of the packet store.

### 18.3.8 Reporting packet store catalogues

The request to report the catalogues of selected packet stores shall be:

#### Report Catalogues for Selected Packet Stores (15,12)

Telecommand packet, application data:

N	Store ID
Unsigned Integer	Fixed CharString
<div style="display: flex; justify-content: space-around; align-items: center;"> <span>← Optional →</span> <span>← Repeated N times →</span> </div>	

N:

The number of packet stores whose catalogues shall be reported. By convention, N = 0 shall mean “all packet stores”.

This field shall be systematically omitted if the service provider only supports one packet store.

When this request is received by the application process which provides the storage and retrieval sub-service, the catalogues for the specified packet stores shall be reported. The format of this report shall be:

#### Packet Store Catalogue Report (15,13)

Telemetry source packet, source data:

N	Store ID	Application Process ID 1	Source Sequence Count 1	Service Type 1	Service Subtype 1	Packet Sub-counter 1	Storage Time 1
Unsigned Integer	Fixed CharString	Enumerated	Unsigned Integer	Enumerated	Enumerated	Unsigned Integer	Absolute Time

Optional (N)      Repeated N times      Optional (Packet Sub-counter 1)      Optional (Storage Time 1)

Application Process ID 2	Source Sequence Count 2	Service Type 2	Service Subtype 2	Packet Sub-counter 2	Storage Time 2
Enumerated	Unsigned Integer	Enumerated	Enumerated	Unsigned Integer	Absolute Time

Repeated N times (contd.)      Optional (Packet Sub-counter 2)      Optional (Storage Time 2)

Percentage Filled	Percentage Downlinked	No. of Packets Stored	No. of Packets Downlinked
Unsigned Integer	Unsigned Integer	Unsigned Integer	Unsigned Integer

Repeated N times (contd.)

N, Store ID:

These fields shall be systematically omitted if the service only supports one packet store.

Application Process ID 1, Source Sequence Count 1, Service Type1, Service Subtype1, Packet Sub-counter1:

Application Process ID 2, Source Sequence Count 2, Service Type2, Service Subtype2, Packet Sub-counter2:

The corresponding fields from the packet header and data field header of the oldest and newest packets respectively within the given packet store.

Storage Time 1, Storage Time 2:

The absolute time of reception of the oldest and newest packets respectively. These fields shall be systematically omitted if the service provider does not time-stamp packets on arrival at the packet store.

**Percentage Filled:**

The percentage of the packet store occupied by stored packets.

**Percentage Downlinked:**

The percentage of the packet store that has already been downlinked.

**No. of Packets Stored:**

The total number of packets stored in the packet store.

**No. of Packets Downlinked:**

The number of packets stored in the packet store that have already been downlinked.

## 18.4 Capability sets

### 18.4.1 Minimum capability sets

The minimum capability set for the packet selection sub-service shall consist of the service requests and reports given in Table 34.

**Table 34: Summary of packet selection sub-service minimum capabilities**

Subtype	Service request, report or capability
1	Enable Storage in Packet Stores
2	Disable Storage in Packet Stores

The minimum capability set for the storage and retrieval sub-service shall consist of one of the downlinking options given in Table 35.

**Table 35: Summary of storage and retrieval sub-service minimum capabilities**

Subtype	Service request, report or capability	Requires subtype
	Downlinking into a dedicated VC	7 and/or 9
8	Packet Store Contents Report	7 and/or 9

### 18.4.2 Additional capability sets

The packet selection sub-service provider can implement the additional capabilities given in Table 36.

**Table 36: Summary of packet selection sub-service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
3	Add Packets to Storage Selection Definition	4
4	Remove Packets from Storage Selection Definition	3
5	Report Storage Selection Definition	3, 4 and 6
6	Storage Selection Definition Report	3, 4 and 5

The storage and retrieval sub-service provider can implement the additional capabilities given in Table 37.

**Table 37: Summary of storage and retrieval sub-service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
	Support of circular packet store	
	Support of bounded packet store	10 and/or 11
	Time stamping of stored packets	9 and/or 11
7	Downlink Packet Store Contents for Packet Range	
9	Downlink Packet Store Contents for Time Period	
10	Delete Packet Stores Contents up to Specified Packets	
11	Delete Packet Stores Contents up to Specified Storage Time	
12	Report Catalogues for Selected Packet Stores	13
13	Packet Store Catalogue Report	12

*(This page is intentionally left blank)*

---

# 19

---

## Test service

### 19.1 Scope

The test service provides the capability to activate test functions implemented on-board and to report the results of such tests.

This service satisfies the concepts described in subclause 4.15 of this Standard.

Any number of application processes may provide a single instance of the test service.

Telemetry source packets and telecommand packets relating to the test service shall be denoted by Service Type = 17.

### 19.2 Service concept

The majority of test functions are mission-specific in nature. The only generic test identified for this Standard is an end-to-end “connection test” between the ground system and the application process.

The function exercised by the connection test service request shall be the generation of a corresponding one-shot service report by the application process. The reception on the ground of the service report shall serve to confirm that the routes (uplink and downlink) between itself and the application process are operational and that the application process itself is performing a minimum set of functions (which includes telecommand processing).

### 19.3 Service requests and reports

The request to perform an end-to-end connection test shall be:

#### **Perform Connection Test (17,1)**

Telecommand packet, application data: None.

When this request is received, a report shall be generated as follows:

#### **Link Connection Report (17,2)**

Telemetry source packet, source data: None.

## 19.4 Capability sets

### 19.4.1 Minimum capability set

The minimum capability set shall consist of the service requests and reports given in Table 38.

**Table 38: Summary of test service minimum capabilities**

Subtype	Service request, report or capability
1	Perform Connection Test
2	Connection Test Report

### 19.4.2 Additional capability set

There are currently no additional capabilities defined.



---

## **On-board operations procedure service**

### **20.1 Scope**

The ground system can define a set of operations procedures that it can load to an application process, which then manages the on-board storage of these procedures and their subsequent execution under ground system control. In principle, such an operations procedure can also be controlled (e.g. started) autonomously on-board, e.g. as the result of detection of a specific on-board event.

The on-board operations procedure service provides standard service requests and reports for controlling the execution of these procedures and monitoring their status. This service satisfies the on-board operations procedures concept described in subclause 4.9 of this Standard. The specification of the operations language used for the procedure execution is beyond the scope of this Standard.

Any number of on-board application processes may provide a single instance of the on-board operations procedure service.

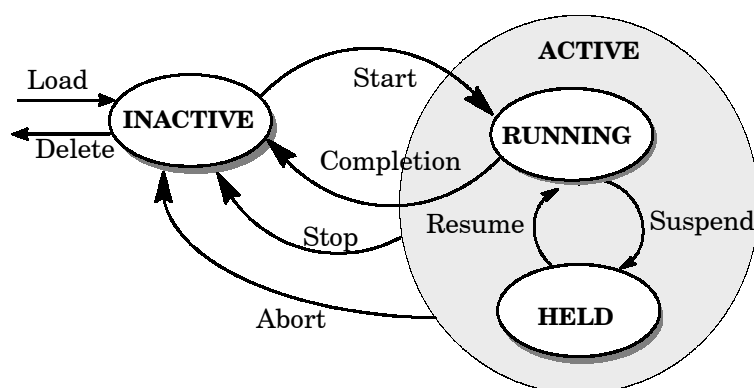
Telemetry source packets and telecommand packets relating to the on-board operations procedure service shall be denoted by Service Type = 18.

### **20.2 Service concept**

An on-board operations procedure managed by a given application process is uniquely identified by a "Procedure ID".

An on-board operations procedure shall be comprised of a number of logical elements or steps, each of which shall be identified by a unique "Step ID".

An on-board operations procedure shall have an execution status which indicates whether it is currently "inactive", "running" or "held" (the latter two states together constituting "active"). The states and the transitions between them are shown in Figure 9. The ground system can provide run-time parameters to the on-board operations procedure at the time it is started, or when it is "running".



**Figure 9: States and transitions for an on-board operations procedure**

The service shall maintain a list of the currently loaded procedures and a list of the currently active procedures and these lists shall be reported to the ground on request.

Reporting on the progress of execution of a procedure can be achieved either by the use of telecommand verification reports (see clause 6) or event reports (see clause 10).

## 20.3 Service requests and reports

### 20.3.1 Loading a procedure

The request shall be:

#### Load Procedure (18,1)

Telecommand packet, application data:

Procedure ID	Length	Procedure Code
Fixed CharString	Variable OctetString	

Procedure ID:

The unique identification of the procedure that is being loaded. The length of the character string can be defined on a mission basis or per application process.

The internal structure of the Procedure ID is beyond the scope of this Standard.

Length:

The length (in octets) of the procedure code that follows.

Procedure Code:

The code of the procedure (in increasing order of octet).

When this request is received, the application process shall store the procedure code and update its list of loaded on-board procedures accordingly. If the Procedure ID is the same as a procedure currently in the list, the procedure just loaded shall replace the one that was previously loaded. The status of the loaded procedure shall be set to “inactive”. However, if the procedure to be replaced is currently “active”, the request shall be ignored.

### 20.3.2 Deleting a procedure

The request shall be:

#### Delete Procedure (18,2)

Telecommand packet, application data:

Procedure ID
Fixed CharString

When this request is received, the specified on-board procedure shall be deleted from the list of loaded on-board procedures and the area occupied by the procedure code shall be cleared.

The request shall be ignored if the procedure status is “running” or “held”.

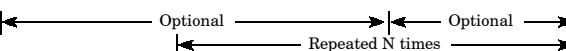
### 20.3.3 Starting a procedure

The request shall be:

#### Start Procedure (18,3)

Telecommand packet, application data:

Procedure ID	N	Parameter#	Value
Fixed CharString	Unsigned Integer	Enumerated	Deduced



N:

The number of (Parameter#, Value) pairs which follow.

Parameter#:

The identification of the parameter whose value follows.

Value:

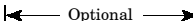
The value of the corresponding parameter (its type can be deduced from the Parameter#)

The parameters shall be used to configure the specific instance of execution of the on-board procedure. Two options are available for supplying parameters (the option is at service-level, rather than procedure-level or packet instance):

- a. Only a subset of parameters is loaded.  
In this case, the packet contains an array of the corresponding parameter numbers and values. Any parameters not loaded shall retain their current values.
- b. The full set of parameters is loaded.  
In this case, the packet shall contain a data structure (of parameter values) conforming to structure rules set #1 (see subclause 23.6.2), where the structure is known implicitly.

In this case, the packet structure shall become simply:

Procedure ID	Parameters
Fixed CharString	Any



When this request is received, the specified on-board procedure shall be started using (or passing) the specified activation parameters. The procedure status shall then be “running”.

The request shall be ignored if the status of the procedure was “running” or “held”.

The verification of execution of this “start procedure” request can report on the progress of execution of the procedure using the standard reports of the telecommand verification service, defined in clause 6 of this Standard.

### 20.3.4 Stopping a procedure

The request shall be:

#### Stop Procedure (18,4)

Telecommand packet, application data:

Procedure ID	Step ID
Fixed CharString	Enumerated

← Optional →

Step ID:

The Step ID indicates the step at the completion of which the procedure shall be stopped.

When this request is received, the specified on-board procedure shall be stopped at the completion of the indicated step. If no Step ID is provided, the procedure shall be stopped at the completion of the current step. The procedure status shall then be “inactive”.

The request shall be ignored if the procedure already has the “inactive” status.

### 20.3.5 Suspend a procedure

The request shall be:

#### Suspend Procedure (18,5)

Telecommand packet, application data:

Procedure ID	Step ID
Fixed CharString	Enumerated

← Optional →

Step ID:

The Step ID indicates the step at the completion of which the procedure shall be suspended.

When this request is received, the specified on-board procedure shall be suspended at the completion of the indicated step. If no Step ID is provided, the procedure shall be suspended at the completion of the currently executing instruction. The procedure status shall then be “held”.

The request shall be ignored if the procedure status was “inactive” or “held”.

### 20.3.6 Resume a procedure

The request shall be:

#### Resume Procedure (18,6)

Telecommand packet, application data:

Procedure ID	Step ID
Fixed CharString	Enumerated

← Optional →

Step ID:

The Step ID indicates the step from the start of which the procedure shall be resumed.

When this request is received, the specified on-board procedure shall be resumed from the start of the indicated step. If no Step ID is provided, the procedure shall be resumed from the next step after that at which it was suspended. The procedure status shall then be “running”.

The request shall be ignored if the procedure status was “inactive” or “running”.

### 20.3.7 Aborting a procedure

The request shall be:

#### Abort Procedure (18,12)

Telecommand packet, application data:

Procedure ID
Fixed CharString

When this request is received, the specified on-board procedure shall be aborted immediately. The procedure status shall then be “inactive”.

The request shall be ignored if the procedure status is “inactive”.

### 20.3.8 Communicate parameters to a procedure

The request shall be:

#### Communicate Parameters to a Procedure (18,7)

Telecommand packet, application data:

Procedure ID	N	Parameter#	Value
Fixed CharString	Unsigned Integer	Enumerated	Deduced

← Optional → ← Optional →  
← Repeated N times →

N:

The number of (Parameter#, Value) pairs which follow.

Parameter#:

The identification of the parameter whose value follows.

Value:

The value of the corresponding parameter (its type shall be deduced from the Parameter#)

Two options are available for communicating parameters, as described for subtype 1 above.

When this request is received, the application process shall pass the parameters to the procedure. The procedure status shall be unchanged by this action.

The request shall be ignored if the procedure status was “inactive” or “held”.

### 20.3.9 Reporting the list of on-board operations procedures

The request shall be:

#### Report List of on-board Operations Procedures (18,8)

Telecommand packet, application data: None.

When this request is received, a report shall be generated as follows:

#### On-board Operations Procedures List Report (18,9)

Telemetry source packet, source data:

NPROC	Procedure ID
Unsigned integer	Fixed CharString

← Repeated NPROC times →

NPROC:

The number of procedures loaded on-board that follow.

### 20.3.10 Reporting the list of active on-board operations procedures

The request shall be:

#### Report List of Active on-board Operations Procedures (18,10)

Telecommand packet, application data: None.

When this request is received, a report shall be generated as follows:

#### On-board Active Operations Procedures List Report (18,11)

Telemetry source packet, source data:

NPROC	Procedure ID	Status	Step ID
Unsigned integer	Fixed CharString	Enumerated	Enumerated

← Repeated NPROC times →

NPROC:

The number of active on-board procedures that follow.

Status:

The status of the corresponding procedure. This shall be either:

value = 0: running

value = 1: held

Step ID:

The currently executing step within the corresponding procedure (in the case of a held procedure, this is the step at which the procedure was suspended).

## 20.4 Capability sets

### 20.4.1 Minimum capability set

The minimum capability set shall consist of the service requests and reports given in Table 39.

**Table 39: Summary of on-board operations procedure service minimum capabilities**

Subtype	Service request, report or capability
1	Load Procedure
2	Delete Procedure
3	Start Procedure
4	Stop Procedure
5	Suspend Procedure
6	Resume Procedure
12	Abort Procedure

### 20.4.2 Additional capability sets

In addition to a minimum capability set, the additional capabilities given in Table 40 can be implemented.

**Table 40: Summary of on-board operations procedure service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
7	Communicate Parameters to a Procedure	
8	Report List of on-board Operations Procedures	9
9	On-board Operations Procedures List Report	8
10	Report List of Active on-board Operations Procedures	11
11	Active on-board Operations Procedures List Report	10

*(This page is intentionally left blank)*



---

## 21

---

# Event-action service

### 21.1 Scope

As an extension to the on-board capability for detecting events and reporting them asynchronously to the ground system, this service provides the capability to define an action that is executed autonomously on-board when a given event is detected.

The class of events that can give rise to an action are those that also give rise to an event report (see clause 10) and the associated action can be a telecommand of any standard type (i.e. as defined in this Standard) or any mission-specific telecommand.

This service satisfies the concept for attaching an action to an on-board event described in subclause 4.10 of this Standard.

Any number of on-board application processes may provide a single instance of the event-action service.

Telemetry source packets and telecommand packets relating to the event-action service shall be denoted by Service Type = 19.

### 21.2 Service concept

The service shall maintain a list of events to be detected that contains the following information:

- a. Application Process ID generating the event report (optional, see below);
- b. Event report ID;
- c. Associated action (telecommand packet);
- d. Status of the action - enabled or disabled (optional).

The list shall be updated in accordance with requests from ground and (optionally) the list information shall be reported to ground on request.

The service can be designed to detect event reports generated by one (e.g. its own) or more application process. On reception of an event report, the service shall scan the detection list and if a matching event report is detected and the associated action is enabled, the corresponding telecommand packet shall be sent to the destination application process.

## 21.3 Service requests and reports

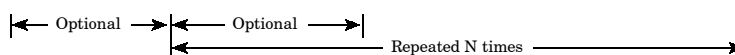
### 21.3.1 Adding events to the detection list

The request shall be:

#### Add Events to the Detection List (19,1)

Telecommand packet, application data:

N	Application Process ID	RID	Telecommand Packet
Unsigned integer	Enumerated	Enumerated	Any TC



N:

The number of events that follow.

This field shall be systematically omitted if the service can add only one event at a time to the detection list.

Application Process ID:

The identifier of the application process generating this event report.

This field shall be systematically omitted if the service is designed only to receive event reports from a single application process.

RID:

The identifier of the event report to be added to the detection list.

Telecommand Packet:

The action to be taken (i.e. telecommand to be sent) when this event report is detected.

When this request is received, the event(s) shall be added to the detection list and the corresponding action status shall be set to "disabled". If a given event is already in the detection list, the action shall replace the existing one, providing that the current action status is "disabled". Otherwise, the request to replace that event shall be ignored.

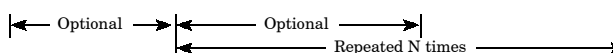
### 21.3.2 Deleting events from the detection list

The request shall be:

#### Delete Events from the Detection List (19,2)

Telecommand packet, application data:

N	Application Process ID	RID
Unsigned integer	Enumerated	Enumerated



N:

This field shall be systematically omitted if the service can delete only one event at a time from the detection list.

Application Process ID:

This field shall be systematically omitted if the service is designed only to receive event reports from a single application process.

When this request is received, the indicated event(s) shall be deleted from the detection list providing that the current action status is “disabled”. Otherwise, the request to delete that event shall be ignored.

An error shall be flagged if an event, requested for deletion, is not in the detection list, however the processing of the remainder of the request shall not be affected.

### 21.3.3 Clearing the event detection list

The request shall be:

#### Clear the Event Detection List (19,3)

Telecommand packet, application data: None.

When this request is received, all entries in the detection list shall be cleared.

### 21.3.4 Controlling the actions associated with events

The requests shall be:

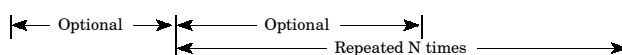
#### Enable Actions (19,4)

Telecommand packet, application data:

#### Disable Actions (19,5)

Telecommand packet, application data:

N	Application Process ID	RID
Unsigned integer	Enumerated	Enumerated



N:

This field shall be systematically omitted if the service can enable or disable only one action at a time.

Application Process ID:

This field shall be systematically omitted if the service is designed only to receive event reports from a single application process.

When this request is received, the action associated with the corresponding event(s) shall be enabled or disabled.

An error shall be flagged if an event, whose action is requested for enabling or disabling, is not in the detection list, however the processing of the remainder of the request shall not be affected.

### 21.3.5 Reporting the event detection list

The request shall be:

#### Report the Event Detection List (19,6)

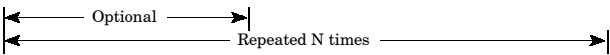
Telecommand packet, application data: None.

When this request is received, a report shall be generated as follows:

#### Event Detection List Report (19,7)

Telemetry source packet, source data:

N	Application Process ID	RID	Action Status
Unsigned integer	Enumerated	Enumerated	Enumerated



N:

The number of events in the event detection list.

Application Process ID:

This field shall be systematically omitted if the service is designed only to receive event reports from a single application process.

Action Status:

This indicates the status of the action associated with the event, as follows:

value = 0 (Disabled)

value = 1 (Enabled).

## 21.4 Capability sets

### 21.4.1 Minimum capability set

The minimum capability set shall consist of the service requests given in Table 41.

**Table 41: Summary of event-action service minimum capabilities**

Subtype	Service request, report or capability
1	Add Events to the Detection List
4	Enable Actions
5	Disable Actions

In addition, one of the requests “Delete Events from the Detection List” (19,2) or “Clear the Event Detection List” (19,3) shall be provided.

### 21.4.2 Additional capability sets

In addition to a minimum capability set, the additional capabilities given in Table 42 can be implemented.

**Table 42: Summary of event-action service additional capabilities**

Subtype	Service request, report or capability	Requires subtype
2	Delete Events from the Detection List	
3	Clear the Event Detection List	
6	Report the Event Detection List	7
7	Event Detection List Report	6

*(This page is intentionally left blank)*

## Summary of service requests and reports

Table 43 provides a summary of the requests and reports for the services defined within this Standard.

Requests appear in the left-hand column and reports in the right-hand column. Reports which are responses are placed in the same row as their corresponding requests, whilst reports which are indications appear alone in a row.

The subtype (ST) for requests and reports which constitute the minimum capability set for a given service is indicated in the table by a shaded background. For some services, the minimum capability set can be selected from a number of alternatives. In these cases, nothing is indicated in this summary table and reference should be made to the corresponding service clause of this Standard.

**Table 43: Summary of requests and reports for PUS standard services**

ST	Service requests	ST	Service reports
<b>Telecommand verification service - 1</b>			
		1	Telecommand Acceptance Report - Success
		2	Telecommand Acceptance Report - Failure
		3	Telecommand Execution Started Report - Success
		4	Telecommand Execution Started Report - Failure
		5	Telecommand Execution Progress Report - Success
		6	Telecommand Execution Progress Report - Failure
		7	Telecommand Execution Completed Report - Success
		8	Telecommand Execution Completed Report - Failure
<b>Device command distribution service - 2</b>			
1	Distribute On-Off Commands		
2	Distribute Register Load Commands		
3	Distribute CPDU Commands		

**Table 43: Summary of requests and reports for PUS standard services  
(continued)**

ST	Service requests	ST	Service reports
<b>Housekeeping and diagnostic data reporting service - 3</b>			
1	Define New Housekeeping Parameter Report		
2	Define New Diagnostic Parameter Report		
3	Clear Housekeeping Parameter Report Definitions		
4	Clear Diagnostic Parameter Report Definitions		
5	Enable Housekeeping Parameter Report Generation		
6	Disable Housekeeping Parameter Report Generation		
7	Enable Diagnostic Parameter Report Generation		
8	Disable Diagnostic Parameter Report Generation		
9	Report Housekeeping Parameter Report Definitions	10	Housekeeping Parameter Report Definitions Report
11	Report Diagnostic Parameter Report Definitions	12	Diagnostic Parameter Report Definitions Report
13	Report Housekeeping Parameter Sampling-Time Offsets	15	Housekeeping Parameter Sampling-Time Offsets Report
14	Report Diagnostic Parameter Sampling-Time Offsets	16	Diagnostic Parameter Sampling-Time Offsets Report
17	Select Periodic Housekeeping Parameter Report Generation Mode		
18	Select Periodic Diagnostic Parameter Report Generation Mode		
19	Select Filtered Housekeeping Parameter Report Generation Mode		
20	Select Filtered Diagnostic Parameter Report Generation Mode		
21	Report Unfiltered Housekeeping Parameters	23	Unfiltered Housekeeping Parameters Report
22	Report Unfiltered Diagnostic Parameters	24	Unfiltered Diagnostic Parameters Report
		25	Housekeeping Parameter Report
		26	Diagnostic Parameter Report
<b>Parameter statistics reporting service - 4</b>			
1	Report Parameter Statistics	2	Parameter Statistics Report
3	Reset Parameter Statistics Reporting		
4	Enable Periodic Parameter Statistics Reporting		
5	Disable Periodic Parameter Statistics Reporting		
6	Add Parameters to Parameter Statistics List		
7	Delete Parameters from Parameter Statistics List		
8	Report Parameter Statistics List	9	Parameter Statistics List Report
10	Clear Parameter Statistics List		
<b>Event reporting service - 5</b>			
		1	Normal/Progress Report
		2	Error/Anomaly Report - Low Severity
		3	Error/Anomaly Report - Medium Severity
		4	Error/Anomaly Report - High Severity
5	Enable Event Report Generation		
6	Disable Event Report Generation		



**Table 43: Summary of requests and reports for PUS standard services  
(continued)**

ST	Service requests	ST	Service reports
<b>Memory management service - 6</b>			
1	Load Memory using Base plus Offsets		
2	Load Memory using Absolute Addresses		
3	Dump Memory using Base plus Offsets	4	Memory Dump using Base plus Offsets Report
5	Dump Memory using Absolute Addresses	6	Memory Dump using Absolute Addresses Report
7	Check Memory using Base plus Offsets	8	Memory Check using Base plus Offsets Report
9	Check Memory using Absolute Addresses	10	Memory Check using Absolute Addresses Report
<b>Function management service - 8</b>			
1	Perform Function		
<b>Time management service - 9</b>			
<b>Rate control sub-service</b>			
1	Change Time Report Generation Rate		
<b>Time reporting sub-service</b>			
		2	Time Report
<b>On-board operations scheduling service - 11</b>			
1	Enable Release of Telecommands		
2	Disable Release of Telecommands		
3	Reset Command Schedule		
4	Insert Telecommands in Command Schedule		
5	Delete Telecommands		
6	Delete Telecommands over Time Period		
15	Time-Shift All Telecommands		
7	Time-Shift Selected Telecommands		
8	Time-Shift Telecommands over Time Period		
16	Report Command Schedule in Detailed Form	10	Detailed Schedule Report
9	Report Subset of Command Schedule in Detailed Form	(10)	
11	Report Subset of Command Schedule in Detailed Form over Time Period	(10)	
17	Report Command Schedule in Summary Form	13	Summary Schedule Report
12	Report Subset of Command Schedule in Summary Form	(13)	
14	Report Subset of Command Schedule in Summary Form over Time Period	(13)	
18	Report Status of Command Schedule	19	Command Schedule Status Report
<b>On-board monitoring service - 12</b>			
1	Enable Monitoring of Parameters		
2	Disable Monitoring of Parameters		
3	Change Maximum Reporting Delay		
4	Clear Monitoring List		
5	Add Parameters to Monitoring List		
6	Delete Parameters from Monitoring List		
7	Modify Parameter Checking Information		
8	Report Current Monitoring List	9	Current Monitoring List Report
10	Report Current Parameters Out-of-limit List	11	Current Parameters Out-of-limit List Report
		12	Check Transition Report

**Table 43: Summary of requests and reports for PUS standard services  
(continued)**

ST	Service requests	ST	Service reports
<b>Large data transfer service - 13</b>			
<b>Data downlink operation</b>			
		1	First Downlink Part Report
		2	Intermediate Downlink Part Report
		3	Last Downlink Part Report
		4	Downlink Abort Report
5	Downlink Reception Acknowledgement		
6	Repeat Parts	7	Repeated Part Report
8	Abort Downlink		
<b>Data uplink operation</b>			
9	Accept First Uplink Part		
10	Accept Intermediate Uplink Part		
11	Accept Last Uplink Part		
12	Accept Repeated Part		
13	Abort Reception of Uplinked Data		
		14	Uplink Reception Acknowledgement Report
		15	Unsuccessfully Received Parts Report
		16	Reception Abort Report
<b>Packet forwarding control service - 14</b>			
1	Enable Forwarding of Telemetry Source Packets		
2	Disable Forwarding of Telemetry Source Packets		
3	Report Enabled Telemetry Source Packets	4	Enabled Telemetry Source Packets Report
5	Enable Forwarding of Housekeeping Packets		
6	Disable Forwarding of Housekeeping Packets		
7	Report Enabled Housekeeping Packets	8	Enabled Housekeeping Packets Report
9	Enable Forwarding of Diagnostic Packets		
10	Disable Forwarding of Diagnostic Packets		
11	Report Enabled Diagnostic Packets	12	Enabled Diagnostic Packets Report
13	Enable Forwarding of Event Report Packets		
14	Disable Forwarding of Event Report Packets		
15	Report Enabled Event Report Packets	16	Enabled Event Report Packets Report
<b>On-board storage and retrieval service - 15</b>			
<b>Packet selection sub-service</b>			
1	Enable Storage in Packet Stores		
2	Disable Storage in Packet Stores		
3	Add Packets to Storage Selection Definition		
4	Remove Packets from Storage Selection Definition		
5	Report Storage Selection Definition	6	Storage Selection Definition Report
<b>Storage and retrieval sub-service</b>			
7	Downlink Packet Store Contents for Packet Range	8	Packet Store Contents Report
9	Downlink Packet Store Contents for Time Period	(8)	
10	Delete Packet Stores Contents up to Specified Packets		



**Table 43: Summary of requests and reports for PUS standard services  
(continued)**

ST	Service requests	ST	Service reports
11	Delete Packet Stores Contents up to Specified Storage Time		
12	Report Catalogues for Selected Packet Stores	13	Packet Store Catalogue Report
<b>Test service - 17</b>			
1	Perform Connection Test	2	Connection Test Report
<b>On-board operations procedure service - 18</b>			
1	Load Procedure		
2	Delete Procedure		
3	Start Procedure		
4	Stop Procedure		
5	Suspend Procedure		
6	Resume Procedure		
12	Abort Procedure		
7	Communicate Parameters to a Procedure		
8	Report List of On-board Operations Procedures	9	On-board Operations Procedures List Report
10	Report List of Active On-board Operations Procedures	11	Active On-board Operations Procedures List Report
<b>Event-action service - 19</b>			
1	Add Events to the Detection List		
2	Delete Events from the Detection List		
3	Clear the Event Detection List		
4	Enable Actions		
5	Disable Actions		
6	Report the Event Detection List	7	Event Detection List Report

*(This page is intentionally left blank)*

---

## 23

---

# Parameter types and structure rules

## 23.1 Introduction

Each field in a telecommand or telemetry source packet described in this Standard is either a parameter field containing a parameter value or a structured field containing several parameter fields organized according to a set of rules.

A parameter field has a simple type defined by the set of values which the parameter can take. The simple types (or parameter types) are defined in subclause 23.5.

A structured field has a structured type indicating the high level organization of the field and is defined by reference to one or more other types (the types of its components). The structured types are described in subclause 23.6.

The specification of the structure and content of the source data field (for telemetry source packets) or application data field (for telecommand packets) in the packet data field of the packets defined in this Standard is expressed in term of the types of its simple and structured components.

The various types described in this clause form a subset (sometimes with explicit restrictions or extensions) of the simple and structured types defined in ISO 8824 “Information Technology – Abstract Syntax Notation One (ASN.1)” (Reference [5]).

The ASN.1 notation is not used in this Standard. A simpler tabular notation is used to describe the packet structures of the service requests and reports specified in this Standard. The tabular notation shows the names and order of appearance of the fields in the packet and the type of each field.

This clause defines the physical encoding rules for each simple and structured type, their lengths and the internal format used to encode values.

These encoding rules differ from the rules defined in ISO 8825 “Information Technology – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)” (Reference [6]) for several reasons, the main reason being the mission constraints imposed by the space link bandwidth.

This Standard does not specify a particular length (and physical encoding format) for each field in the packet data field of the packets. A mission shall select for each field the length and encoding format which best suit its requirements.

## 23.2 Conventions

The following convention is used to identify each bit in an N-bit field:

The first bit in a field, starting from the left, is defined to be “Bit-0” and is represented as the leftmost justified bit on a figure. The following bit is called “Bit-1”, and so on, up to “Bit-N-1”, the bits being represented in this order from left to right of a figure.

The following nomenclature is used throughout this Standard to describe adjacent groups of bits within packets:

1 OCTET = 1 BYTE = 8 bits

## 23.3 Encoding formats of parameter types

The parameter type defines the abstract class to which the parameter values belong. For a given parameter type there can be variations in the format and length of the values.

Each combination of a parameter type and encoding format has an associated parameter code, which identifies completely a simple type and how it is physically encoded in a packet.

The parameter code shall be used whenever an explicit definition of a parameter field is requested.

EXAMPLE To specify the physical format selected by a mission; for parameter interpretation in the ground system.

The parameter codes are valid for both telecommand and telemetry data.

When contained in a packet, the parameter code should consist of two consecutive parameter fields with type “Enumerated” (as defined later in this clause):

$$PC =$$

PTC	PFC
Enumerated	Enumerated

Where:

PTC = Parameter Type Code  
PFC = Parameter Format Code

} PC = Parameter Code

The only parameter types for telemetry and telecommand parameters shall be those defined in subclauses 23.5.1 to 23.5.11. For each type of parameter, a list of standard parameter formats is defined in this Standard. Missions shall adopt these standard parameter formats, but other mission-specific parameter formats can also be defined with new parameter format codes.

NOTE 1 The parameter code only restricts the set of bit combinations encoding the parameter values. However, the set of values that the parameter may take can be further restricted on a mission basis.

NOTE 2 Some simple types have variable-length physical formats (e.g. a variable-length bit string). The actual length is encoded as an unsigned integer field which physically precedes the variable-length value.

Thus, the complete definition of the physical format of a variable-length parameter field also includes the parameter code of the field containing the actual length of the parameter value.

## 23.4 Tailoring of packet structures for a mission

The specification of the service requests and reports defined in this Standard only indicates the parameter types of the parameter fields in the corresponding packet structures. During the design phase of a mission, the actual physical formats shall be selected by specifying the parameter code of each parameter field found in each implemented packet type and subtype.

**EXAMPLE** The format of the time field in the packet data header of the telemetry source packets is selected on the basis of the time resolution applicable for the monitoring of an application process.

The actual physical format for a parameter field can be:

- a. **Mission defined:**  
The physical format of the parameter field shall be unique for the mission. The parameter field can appear in different packet types and subtypes or can appear in a packet sent to or generated by different application processes; in all cases its physical format shall be the same.
- b. **Application process defined:**  
The physical format of the parameter field shall be defined per application process. The encoding of the parameter field shall be identical in all packet types and subtypes sent to or generated by the application process.
- c. **Service instance defined:**  
The physical format of the parameter field shall be defined per service provided by an application process. The encoding of the parameter field shall be identical in all packet subtypes received or generated by a service provided by a given application process (this can reflect, for example, the fact that several organizations implement the on-board software).
- d. **Request or report defined:**  
The physical format of the parameter field in a packet shall be deduced from the value(s) of one or several preceding parameter fields in the packet (including the packet type and subtype). It can additionally depend on the value(s) of one or several mission constant(s), on the destination (or source) application process and on the service provided by the application process.

The actual type of a parameter field in a packet can also be mission defined, application process defined, service instance defined or request or report defined. The specification of the packet structures defined in this Standard highlights which parameter fields shall have their types defined in this way by indicating that their type is "Deduced".

The presence or absence of a parameter field specified as "Optional" in the structure of a packet can be mission defined, application process defined, service instance defined or request or report defined. The level at which a given field is optional is explained within the corresponding definition for that field.

A mission can specify that a parameter field starts on a byte boundary, or on a 16-bit, 24-bit, 32-bit or 64-bit word boundary. This can be achieved by ensuring that the preceding parameter field ends on a word boundary (increasing its size if necessary) or by introducing padding bits between two parameter fields. All such padding bits shall be set to zero.

When tailoring the PUS, a mission may decide to implement only a subset of the parameter types or formats defined in the following subclause.

When a mission tailors the packet structures to suit its requirements, it should take into account the impact on related data processing systems and databases.

## 23.5 Simple parameter types

### 23.5.1 Boolean parameter (PTC = 1)

PFC = 0

This is a one-bit parameter, with two distinguished values only, involved in logical operations where:

value 1 denotes "TRUE"

value 0 denotes "FALSE".

### 23.5.2 Enumerated parameter (PTC = 2)

PFC = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 24 or 32: length in bits of the parameter values

This is a parameter, with discrete integer values only, involved in logical and comparative expressions (but not in numeric and relational expressions). The values that such a parameter can take are discrete and unordered. Each value has a meaning which is interpreted as a character string value. An error code is a typical example (e.g. 0 means "unchecked", 3 means "invalid").

### 23.5.3 Unsigned integer parameter (PTC = 3)

The values that this parameter can take are positive and can be involved in arithmetical, relational and comparative expressions. An unsigned integer shall be encoded with Bit-0 being the most significant bit (MSB) and Bit-N-1 the least significant bit (LSB).

The formats of an unsigned integer shall be:

Format Code	Format Definition	Lowest Value	Highest Value
$0 \leq \text{PFC} \leq 12$	PFC +4 bits, unsigned	0	$2^{\text{PFC}+4}-1$ (15 to 65 535)
PFC = 13	3 octets, unsigned	0	$2^{24}-1$ (16 777 215)
PFC = 14	4 octets, unsigned	0	$2^{32}-1$ ( $\cong 4,3 \cdot 10^9$ )
PFC = 15	6 octets, unsigned	0	$2^{48}-1$ ( $\cong 2,8 \cdot 10^{14}$ )
PFC = 16	8 octets, unsigned	0	$2^{64}-1$ ( $\cong 18,5 \cdot 10^{18}$ )

### 23.5.4 Signed integer parameter (PTC = 4)

The values that such a parameter can take are positive or negative and can be involved in arithmetical, relational and comparative expressions.

- If Bit-0 = 0, the value shall be positive following the unsigned integer convention.
- If Bit-0 = 1, the value shall be negative and the field is the 2's complement of the positive value.



The formats of a signed integer shall be:

Format Code	Format Definition	Lowest Value	Highest Value
$0 \leq \text{PFC} \leq 12$	PFC +4 bits, signed	$-2^{\text{PFC}+3}$ (-8 to -32 768)	$2^{\text{PFC}+3}-1$ (7 to 32 767)
PFC = 13	3 octets, signed	$-2^{23}$ (-8 388 608)	$2^{23}-1$ (8 388 607)
PFC = 14	4 octets, signed	$-2^{31}$ ( $\cong -2,15 \cdot 10^9$ )	$2^{31}-1$ ( $\cong 2,15 \cdot 10^9$ )
PFC = 15	6 octets, signed	$-2^{47}$ ( $\cong -1,4 \cdot 10^{14}$ )	$2^{47}-1$ ( $\cong 1,4 \cdot 10^{14}$ )
PFC = 16	8 octets, signed	$-2^{63}$ ( $\cong -9,2 \cdot 10^{18}$ )	$2^{63}-1$ ( $\cong 9,2 \cdot 10^{18}$ )

### 23.5.5 Real parameter (PTC = 5)

PFC = 1: 4 octets simple precision format (IEEE standard).

PFC = 2: 8 octets double precision format (IEEE standard).

PFC = 3: 4 octets simple precision format (MIL standard).

PFC = 4: 6 octets extended precision format (MIL standard).

#### 23.5.5.1 IEEE standard format

The simple-precision format and the double-precision format are defined in “IEEE 754 Standard for Binary Floating-Point Arithmetic” (Reference [7]). The important features of their definitions are repeated here.

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

$$(-1)^S \times 2^E \times (b_0 \bullet b_1 b_2 \dots b_{p-1})$$

where:

$b_0 \bullet b_1 b_2 \dots b_{p-1}$  means  $b_0/2^0 + b_1/2^1 + b_2/2^2 + \dots + b_{p-1}/2^{p-1}$ ;

S = 0 or 1;

E = any integer between  $E_{\min}$  and  $E_{\max}$ , inclusive;

$b_i$  = 0 or 1;

p = number of significant bits (precision).

Each format also permits the representation of two infinities, “ $+\infty$ ” and “ $-\infty$ ” and special values which are not numbers.

Real numbers in both formats are composed of 3 subfields:

Sign	unsigned integer, contains the value S
Exponent	unsigned integer, contains the value E+127 on 8 bits (single precision) or E+1023 (double precision)
Fraction	bit string, contains the value $\bullet b_1 b_2 \dots b_{p-1}$ with p = 24 (single precision) or p = 53 (double precision)

The encoded value of a single-precision real parameter shall be constituted as follows:

Sign	Exponent	Fraction
1 bit	8 bits	23 bits

The value of a single-precision parameter shall be:

	Value
If exponent = 255 and fraction $\neq 0$	Not a Number
If exponent = 255 and fraction = 0	$(-1)^{\text{sign}} \times \infty$
If $0 < \text{exponent} < 255$	$(-1)^{\text{sign}} \times 2^{\text{exponent}-127} \times (1, \text{fraction})$
If exponent = 0 and fraction $\neq 0$	$(-1)^{\text{sign}} \times 2^{-126} \times (0, \text{fraction})$
If exponent = 0 and fraction = 0	0

The encoded value of a double-precision real parameter shall be constituted as follows:

Sign	Exponent	Fraction
1 bit	11 bits	52 bits

The value of a double-precision parameter shall be:

	Value
If exponent = 2047 and fraction $\neq 0$	Not a Number
If exponent = 2047 and fraction = 0	$(-1)^{\text{sign}} \times \infty$
If $0 < \text{exponent} < 2047$	$(-1)^{\text{sign}} \times 2^{\text{exponent}-1023} \times (1, \text{fraction})$
If exponent = 0 and fraction $\neq 0$	$(-1)^{\text{sign}} \times 2^{-1022} \times (0, \text{fraction})$
If exponent = 0 and fraction = 0	0

In the cases where Exponent = 0 and Fraction  $\neq 0$ , the values are called denormalized.

The range of possible values and precision for a real parameter are as follows:

Simple precision:

$$1,12 \times 10^{-38} \leq |\text{value}| \leq 3,40 \times 10^{38} \text{ (precision } 1,15 \times 10^{-7})$$

Double precision:

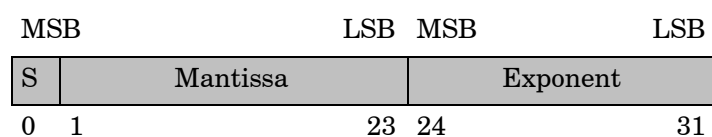
$$2,22 \times 10^{-308} \leq |\text{value}| \leq 1,79 \times 10^{308} \text{ (precision } 2,22 \times 10^{-16})$$

### 23.5.5.2 United States Air Force (USAF) military standard format

This is defined in "Military Standard Sixteen-Bit Computer Instruction Set Architecture" MIL-STD-1750a, 2<sup>nd</sup> July 1980 (Reference [8]) and this definition is repeated below:

#### Simple precision floating point data:

This shall be represented as a 32-bit quantity consisting of a 24-bit 2's complement mantissa and an 8-bit 2's complement exponent.



Floating point numbers are represented as a fractional mantissa times 2 raised to the power of the exponent. All floating point numbers are assumed normalized or floating point zero at the beginning of a floating point operation and the results

of all floating point operations are normalized (a normalized floating point number has the sign of the mantissa and the next bit of opposite value) or floating point zero. A floating point zero is defined as 0000 0000<sub>16</sub>, that is, a zero mantissa and a zero exponent (00<sub>16</sub>). An extended floating point zero is defined as 0000 0000 0000<sub>16</sub>, that is, a zero mantissa and a zero exponent. Some examples of the machine representation for 32-bit floating point numbers are:

Decimal Number	Hexadecimal Notation
	Mantissa Exp
$0,999\,999\,8 \times 2^{127}$	7FFF FF FF
$0,5 \times 2^{127}$	4000 00 7F
$0,625 \times 2^4$	5000 00 04
$0,5 \times 2^1$	4000 00 01
$0,5 \times 2^0$	4000 00 00
$0,5 \times 2^{-1}$	4000 00 FF
$0,5 \times 2^{-128}$	4000 00 80
$0,0 \times 2^0$	0000 00 00
$-1,0 \times 2^0$	8000 00 00
$-0,500\,000\,1 \times 2^{-128}$	BFFF FF 80
$-0,750\,000\,1 \times 2^4$	9FFF FF 04

#### Extended Precision Floating Point Data:

Extended floating point data shall be represented as a 48-bit quantity consisting of a 40-bit 2's complement mantissa and an 8-bit 2's complement exponent. The exponent bits 24 to 31 lie between the split mantissa bits 0 to 23 and bits 32 to 47. The most significant bit of the mantissa is the sign bit 0, and the least significant bit of the mantissa is bit 47.

S	Mantissa MS			Exponent		Mantissa LS	
0	1	23	24	31	32	47	

Some examples of the machine representation of 48-bit extended floating point numbers are:

Decimal Number	Mantissa (MS)	Exp	Mantissa (LS)
$0,5 \times 2^{127}$	400000	7F	0000
$0,5 \times 2^0$	400000	00	0000
$0,5 \times 2^{-1}$	400000	FF	0000
$0,5 \times 2^{-128}$	400000	80	0000
$-1,5 \times 2^{127}$	800000	7F	0000
$-1,0 \times 2^0$	800000	00	0000
$-1,0 \times 2^{-1}$	800000	FF	0000
$-1,0 \times 2^{-128}$	800000	80	0000
$0,0 \times 2^0$	000000	00	0000
$-0,75 \times 2^{-1}$	A00000	FF	0000

For both floating point and extended floating point numbers, an overflow is defined as an exponent overflow and an underflow is defined as an exponent underflow.

### 23.5.6 Bit-string parameter (PTC = 6)

PFC = 0: A variable-length bit string.

PFC > 0: A fixed-length bit string with a number of bits equal to PFC.

The values that such a parameter can take are variable-length or fixed-length sequences of bits, each with a value 1 or 0. The meaning and interpretation of a value is application process specific.

A variable-length bit-string parameter shall be of the form:

n	$B_1 \dots B_n$
Unsigned Integer	n bits

Where:

$B_1 \dots B_n$  are bits;

n indicates the number of bits which follows.

A fixed-length bit-string parameter shall be of the form:

$B_1 \dots B_n$
n bits

Where:

$B_1 \dots B_n$  are bits;

n is the number of bits and is equal to PFC.

### 23.5.7 Octet-string parameter (PTC = 7)

PFC = 0: A variable-length octet string.

PFC > 0: A fixed-length octet string with a number of octets equal to PFC.

The values that such a parameter can take are variable-length or fixed-length sequences of octets, each octet being an ordered sequence of eight bits. The meaning and interpretation of a value shall be application process specific.

A variable-length octet-string parameter shall be of the form:

n	O <sub>1</sub>	O <sub>2</sub>	...	O <sub>n</sub>
Unsigned Integer	octet	octet		octet

Where:

O<sub>1</sub>...O<sub>n</sub> are octets;

n indicates the number of octets which follow.

A fixed-length octet-string parameter is of the form:

O <sub>1</sub>	O <sub>2</sub>	...	O <sub>n</sub>
octet	octet		octet

Where:

O<sub>1</sub>...O<sub>n</sub> are octets;

n is the number of octets and is equal to PFC.

### 23.5.8 Character-string parameter (PTC = 8)

PFC = 0: A variable-length character string.

PFC > 0: A fixed-length character string with a number of characters equal to PFC.

The values that such a parameter can take are variable-length or fixed-length sequences of visible characters (visible characters are defined in ANSI X3.4 Reference [9]). A visible character shall be represented by its ASCII code on one octet. The meaning and interpretation of a value is application process specific.

A variable-length character-string parameter shall be of the form:

n	C <sub>1</sub>	C <sub>2</sub>	...	C <sub>n</sub>
Unsigned Integer	ASCII	ASCII		ASCII

Where:

C<sub>1</sub>...C<sub>n</sub> are ASCII character codes on 8 bits;

n indicates the number of ASCII character codes which follow.

A fixed-length character-string parameter shall be of the form:

C <sub>1</sub>	C <sub>2</sub>	...	C <sub>n</sub>
ASCII	ASCII		ASCII

Where:

C<sub>1</sub>...C<sub>n</sub> are ASCII character codes;

n is the number of ASCII character codes and is equal to PFC.

### 23.5.9 Absolute time parameter (PTC = 9)

PFC = 0: Explicit definition of time format (CUC or CDS), i.e. including the P-field.

PFC = 1: 2 octets day CDS format without a  $\mu$ s field (parameter field is 6 octets).

PFC = 2: 2 octets day CDS format with a  $\mu$ s field (parameter field is 8 octets).

$3 \leq \text{PFC} \leq 18$ :

CUC format with  $((\text{PFC} + 1) \div 4, \text{rounded down})$  octets of coarse time and  $((\text{PFC} + 1) \bmod 4)$  octets of fine time. PFC values in this range do not include the P-field (the value of the implicit P-field can be derived from the PFC).

The CUC and CDS time formats are defined in CCSDS 301.0-B-2.

The value of an Absolute Time parameter field shall be a number of seconds and fractions of second from a given Agency epoch. It shall be a positive time offset.

CDS format

The CDS Format with  $\mu$ s (PFC = 2) shall be as follows:

Day	msec of day	$\mu$ sec of msec
2 octets	4 octets	2 octets

The value of Day is an unsigned integer in the range 0 to  $2^{16}-1$ .

CUC format

The full CUC Format shall be as follows:

C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>
1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet

The time in seconds from the given Agency epoch is given by:

$$t = C_1 \cdot 256^3 + C_2 \cdot 256^2 + C_3 \cdot 256 + C_4 + F_1 \cdot 256^{-1} + F_2 \cdot 256^{-2} + F_3 \cdot 256^{-3}.$$

### 23.5.10 Relative time parameter (PTC = 10)

PFC = 0: Explicit definition of CUC time format, i.e. including the P-field

$1 \leq \text{PFC} \leq 16$ :

CUC format with  $((\text{PFC} + 3) \div 4, \text{rounded down})$  octets of coarse time and  $((\text{PFC} + 3) \bmod 4)$  octets of fine time. PFC values in this range do not include the P-field (the value of the implicit P-field can be derived from the PFC).

The value of a Relative Time parameter field shall be a number of seconds and fractions of a second from the occurrence time of an event whose identification can be derived from other parameters in the packet (e.g. identifying a type of on-board event) or a number of seconds and fractions of a second between two absolute times. It can be a positive or negative time offset.

The full CUC Format shall be as follows:

C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>
1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet

A positive time offset is given by:

$$t = C_1 \cdot 256^3 + C_2 \cdot 256^2 + C_3 \cdot 256 + C_4 + F_1 \cdot 256^{-1} + F_2 \cdot 256^{-2} + F_3 \cdot 256^{-3}$$

where C<sub>1</sub> is in the range 0 to 127.

A negative time offset is expressed as the “2’s complement” of the corresponding positive time offset.

### 23.5.11 Deduced parameter (PTC = 11)

PFC = 0

A type which is unspecified and can only be instantiated to one of the simple types defined in the previous subclauses. Its actual type and encoding format for an instance of the parameter field in a packet is deduced from the value(s) of other preceding parameter field(s) in the packet or from the value(s) of mission constants or a combination of both.

**EXAMPLE** A telemetry source packet source data field which contains two parameter fields, the first field being the explicit identification of a parameter and the second field the value of that parameter, as illustrated below:

Parameter#	Parameter Value
Enumerated	Deduced

In this example, successive packets generated by an application process can contain different identifiers of telemetry parameters having different types of parameter value, thus the type of the Parameter Value field is unspecified, whilst its actual type in a packet is deduced from the value in the Parameter# field.

## 23.6 Structured field types

### 23.6.1 Introduction

The packet data field structure and any structured field within it, in particular the source data field or the application data field, shall be defined by a structured type. The structured types of fields shall be defined through rules specifying how to organize the component fields which constitute the field body.

Within this Standard, there is only one set of valid structure rules for both telecommand and telemetry structured fields, and all specialized packet structures within this Standard follow these rules. This set is referred to as structure rules set #1 and is defined below.

### 23.6.2 Structure rules set #1

The structured types shall be:

- **Record type:** this defines a field which is composed of a fixed number of distinct components where each component has its own type (simple or structured).
- **Array type:** this defines a field which is composed of zero, one or more components where all the components shall be of an identical type (simple or structured).

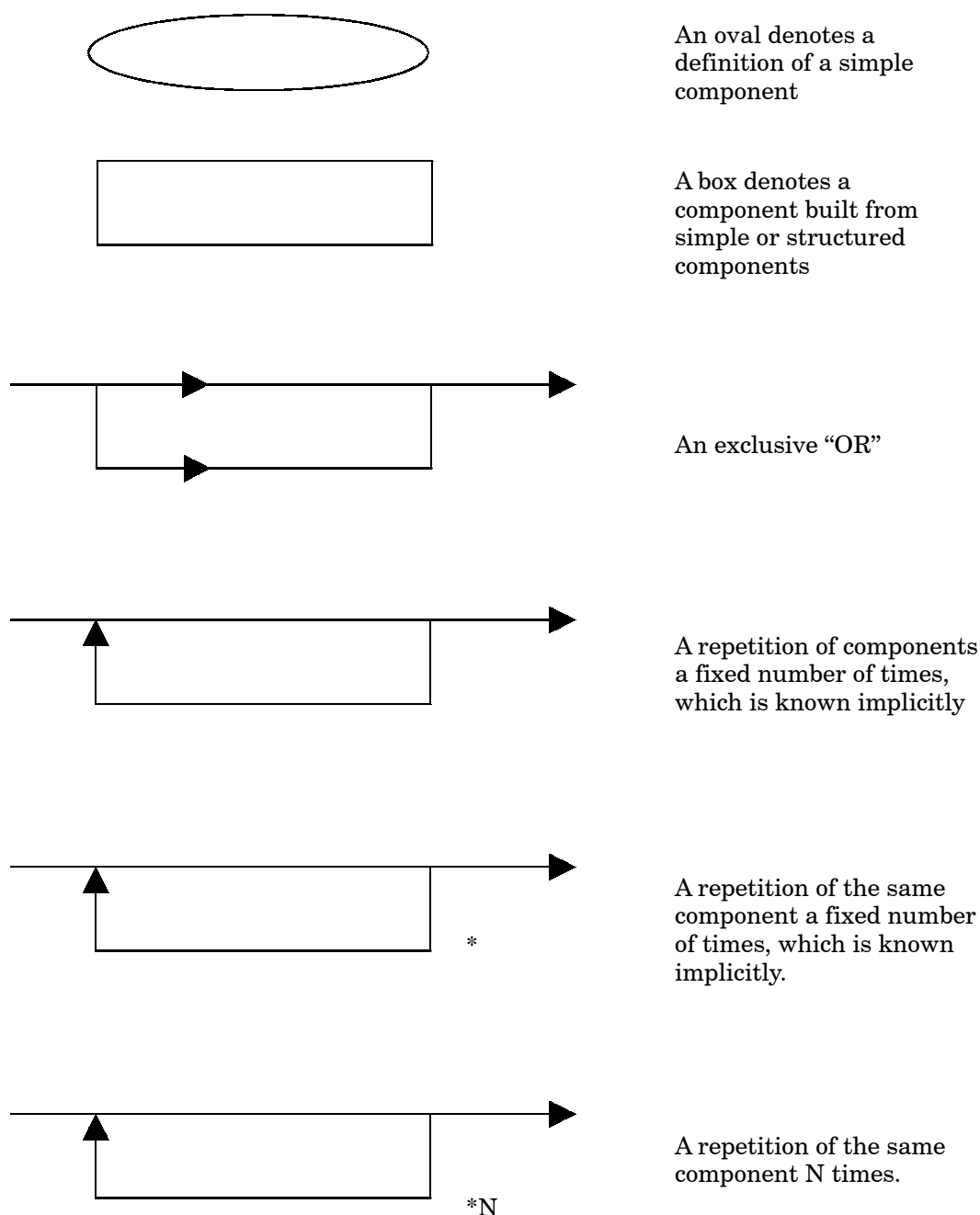
An array can have a fixed number of entries (fixed array) or a variable number of entries (variable array).

- **Deduced structure type:** this defines a field which is composed of one component whose actual type (simple or structured) shall be deduced from the value(s) of other preceding parameter field(s) in the packet or from the value(s) of mission constants or a combination of both.

A deduced parameter field (see subclause 23.5.11) is a particular case of a deduced structure field.

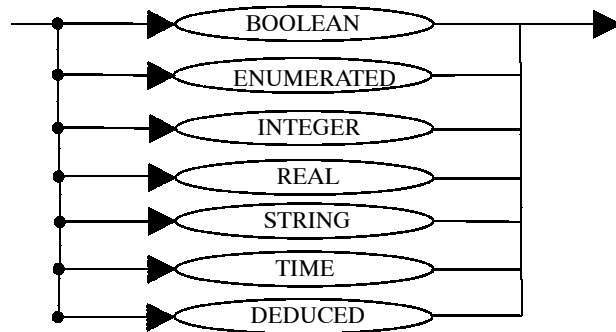
There is no limitation on the level of nesting of components within packets, but this Standard defines specific limitations for some types of packets (other limitations can also be defined by a mission).

The structure rules set #1 utilize the diagram conventions shown in Figure 10.

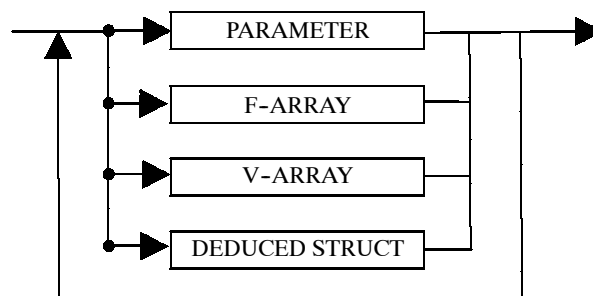


**Figure 10: Diagram conventions used by the structure rules set #1**



**Structure Rules (Set number 1)****PARAMETER****RECORD**

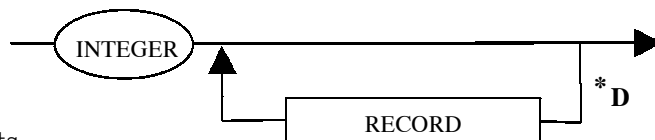
A record is a sequence of distinct parameters, arrays or deduced structures, where the structure of the record is known implicitly

**F-ARRAY**

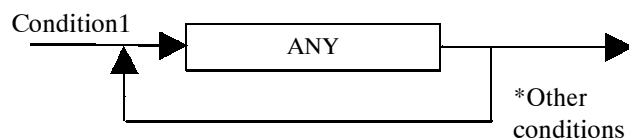
A "Fixed Array" is a repetition of one record n times, where n is known implicitly

**V-ARRAY**

A "Variable Array" is a repetition of one record D times, where D is given explicitly. A Variable Array can only be empty if so stated in its definition (in which case the value D=0 is still present).

**DEDUCED STRUCTURE**

An instance of a "Deduced Structure" can be of any type (simple or structured)



**Figure 10 (continued): Diagram conventions used by the structure rules set #1**

A “Deduced Structure” cannot be fully represented with the same diagram conventions since the actual type for an instance of the deduced structure in a packet shall be deduced from the value(s) of one or several preceding parameter field(s) in the packet or from the value(s) of one or several mission constants (called “determining parameters”) or a combination of both. However, within a given instance of a packet, the deduced structure shall be either a parameter, a record, a variable array, a fixed array or even nothing.

EXAMPLE A determining parameter can be an explicit sampling time or indicate an event to which the values in the deduced structure relate.

There shall be a direct correspondence between the value(s) of the set of determining parameter(s) and the deduced structure instance (e.g. a look-up table can be used).

EXAMPLE A deduced structure dependent on a preceding parameter field P can be a variable array when P = 0, a record A when P = 1 and another record B when P = 2 (see the corresponding illustration at the end of subclause 23.6.4).

The type of a determining parameter can only be Boolean, enumerated, signed or unsigned integer or a character string. It can also be deduced, in which case its actual type shall depend on the value(s) of some of the other determining parameters.

NOTE The determining parameter(s) do not need to be placed consecutively in the packet or placed just before the deduced structure (i.e. there can be other fields separating them).

If an array entry contains a deduced structure, then the entry number can also be used to determine the type of the deduced structure instance.

### 23.6.3 Nesting and identification rules (NIR)

- NIR1:

By definition, in a packet all components which are not components of any array are at level 1. Therefore an array is itself at level 1 if it is an array not nested in another array.

- NIR2:

In a packet, all fields which are components of an array are said to be at the level of the array plus one. By convention, this shall apply to the array entry number which is considered to be an implicit component of the array if it is used for determining the type of a deduced structure instance.

The implications of the two foregoing rules are indicated below each example presented in subclause 23.6.5.

- NIR3:

A parameter field in a packet can contain:

1. The value of a concrete parameter, such as the sampled value of an on-board measurement or the value of a threshold to be set. A concrete parameter shall have a uniquely identifying name and its meaning and interpretation shall be uniquely defined for a mission.

2. The value of a formal parameter. A formal parameter field shall have a meaning and interpretation which is defined with respect to one or more component(s).

EXAMPLE A parameter field T providing the time of sampling of a concrete parameter P is formal (its meaning is “sampling time of P”). In the example presented in subclause 23.5.11, both parameter fields are formal: the meaning of Parameter# is “identification of a concrete parameter” and the meaning of Parameter Value is “value of concrete parameter identified by Parameter#”.

- NIR4:

A concrete parameter can only be explicitly named once in the definition of the contents of a packet and at level 1 or 2 (therefore all parameters at level 3 and above are formal).

This implies that the values of a concrete parameter sampled several times per packet cannot be scattered “randomly” in a packet but shall be components of one, and only one, array. Also, a concrete parameter may only occur in certain instances of a packet if it appears in the definition of a single deduced structure at level 1 or 2.

- NIR5:

A concrete parameter can be explicitly named more than once in the definition of a packet if it has a specialized meaning in each case.

EXAMPLE If it appears in a deduced structure providing information on an event, the specialized meaning can be “value of concrete parameter when event occurred”.

This implies that the packet contains formal parameter fields which hold the “special purposes” values of the concrete parameter (these values can be scattered “randomly” in these packet).

- NIR6:

The last (or only) determining parameter of a deduced structure shall be at the same level or at one level less than the level of its deduced structure. The first determining parameter of a deduced structure shall be at level 1 or 2, and there shall be at least one determining parameter at each level between the level of the first determining parameter and the level of the last determining parameter.

This rule denotes the fact that an array hierarchy provides information which relates to a functional or physical hierarchy for which the determining parameter(s) define an access and interpretation path.

## 23.6.4 Encoding of the structured fields

The rules for the construction of structured types which are described in the previous subclause do not specify how the resulting construct shall be physically encoded in the packets.

This subclause specifies the encoding rules for the structured type fields. These rules apply recursively.

### Record Type

The fixed number (N) of ordered components of a record shall be placed consecutively in the packet, by means of their own physical encoding format:

Component 1	...	Component N
Component 1 Type		Component N Type

A parameter field shall be bit-wise aligned with the last bit of the parameter field which precedes it and shall occupy exactly the number of bits defined for its physical encoding (as defined by its PFC). A specific number of padding bits (see subclause 23.4) can be introduced before the first component and between two successive components.

The value of a variable-length parameter type shall occupy exactly the actual number of bits applicable for the physical encoding of its length field and of its value field (as defined by their PFCs).

### Array Type

The ordered set of entries of an array shall be placed consecutively in the packet and shall be bit-wise aligned. Padding bits can be introduced before the array and between the successive entries of the array.

The set of entries of a variable array shall be preceded by an unsigned integer parameter field whose value (D) is the number of entries in the variable array:

D	Entry 1	...	Entry D
Unsigned Integer	Array Item Type		Array Item Type

### Deduced Structure Type

The deduced structure shall be placed in the packet by means of the encoding format of its actual structure instance. It shall be bit-wise aligned with the preceding field in the packet. Padding bits can be introduced before the deduced structure (this applies to all structure instances).

EXAMPLE If a deduced structure S depends on the value of a preceding determining parameter P, the generic structure shall be:


...	P	...	S
	Enumerated		Deduced Structure

The actual variants of this deduced structure might be


- a variable array when P = 0,
- a record A when P = 1 and
- a record B when P = 2.

The three resulting packet structures are:

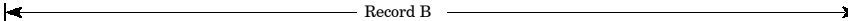
...	P = 0	...	D	Entry 1	Other Entries	Entry D
	Enumerated		Unsigned Integer	Array Item Type	...	Array Item Type



...	P = 1	...	Component A1	Other Components	Component An
	Enumerated		Component A1 Type	...	Component An Type



...	P = 2	...	Component B <sub>1</sub>	Other Components	Component B <sub>m</sub>
	Enumerated		Component B <sub>1</sub> Type	...	Component B <sub>m</sub> Type



### 23.6.5 Examples of structure rules set #1

A simple example of the use of the structure rules set #1 is a classical structure of the parameters field of a housekeeping packet that contains only single samples of parameters, viz.:

R <sub>a</sub>															
P <sub>a</sub>	P <sub>b</sub>	P <sub>c</sub>	P <sub>d</sub>	P <sub>e</sub>	P <sub>f</sub>	P <sub>g</sub>	P <sub>h</sub>	P <sub>i</sub>	P <sub>j</sub>	P <sub>k</sub>	P <sub>l</sub>	P <sub>m</sub>	P <sub>n</sub>	P <sub>o</sub>	P <sub>p</sub>

In the above example, R<sub>a</sub> and its components (P<sub>a</sub> to P<sub>p</sub>) are at level 1.

On the other hand, the following structure utilizes the array constructs defined in structure rules set #1:

R <sub>a</sub>															
R <sub>b</sub>			R <sub>c</sub>	FA <sub>a</sub>						VA <sub>a</sub>			R <sub>d</sub>		
P <sub>a</sub>	P <sub>b</sub>	P <sub>c</sub>	P <sub>d</sub>	R <sub>e</sub>		R <sub>e</sub>		R <sub>e</sub>		D=3	R <sub>f</sub>	R <sub>f</sub>	R <sub>f</sub>	P <sub>h</sub>	P <sub>i</sub>
				P <sub>e</sub>	P <sub>f</sub>	P <sub>e</sub>	P <sub>f</sub>	P <sub>e</sub>	P <sub>f</sub>		P <sub>g</sub>	P <sub>g</sub>	P <sub>g</sub>		

P = Parameter; R = Record; FA = Fixed array; VA = Variable array.

In the second example, R<sub>b</sub> and R<sub>c</sub> can be combined into one record without changing the lower level structure. However, it can often be useful to define separate records corresponding to different sets of information which can appear elsewhere in different structures.

In the second example, R<sub>a</sub>, R<sub>b</sub>, R<sub>c</sub>, R<sub>d</sub> (and their components), FA<sub>a</sub>, VA<sub>a</sub> and D are at level 1 while R<sub>e</sub> and R<sub>f</sub> (and their components) are at level 2.

*(This page is intentionally left blank)*

## Annex A (normative)

### Examples

#### A.1 Specification of the cyclic redundancy code (CRC)

##### A.1.1 General specification

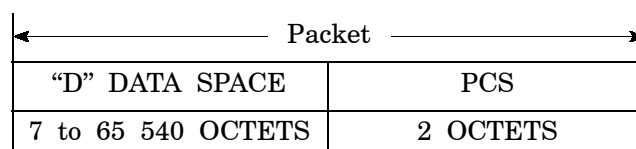
The packet error control field provides the capability for detecting errors which have been introduced into the telemetry source packet (or telecommand packet) by the lower layers during the transmission process or during other processing or storage activities.

The standard error detection encoding/decoding procedure, which is described in detail in the following subclauses, produces a 16-bit Packet Check Sequence (PCS) which is placed in the packet error control field. The characteristics of the PCS are those of a cyclic redundancy code, and can be expressed as follows:

- a. The generator polynomial is:  

$$g(x) = x^{16} + x^{12} + x^5 + 1$$
- b. Both encoder and decoder are initialized to the “all-ones” state for each packet.
- c. PCS generation is performed over the data space “D” as shown in Figure A-1 where “D” covers the entire packet including the packet header but excluding the final packet error control field.
- d. The error detection properties of the CRC can be expressed as follows:
  1. The proportion of all errors in the data that are not detected is approximately  $1,53 \times 10^{-5}$ .
  2. An error in the data affecting an odd number of bits shall always be detected.
  3. An error in the data affecting exactly two bits, no more than 65 535 bits apart, shall always be detected.
  4. If an error in the data affects an even number of bits (greater than or equal to 4), the probability that the error shall not be detected is approximately  $3 \times 10^{-5}$  for a data length of 4 096 octets. The probability increases slightly for larger data lengths and decreases slightly for smaller data lengths.
  5. A single error burst spanning 16 bits or less of the data shall always be detected. Not all intermediate bits in the error burst span need be affected.

This code is intended only for error detection purposes and no attempt should be made to utilize it for correction.



**Figure A-1: Standard packet check sequence generation**

### A.1.2 Encoding procedure

The encoding procedure accepts an (n-16)-bit message and generates a systematic binary (n, n-16) block code by appending a 16-bit Packet Check Sequence (PCS) as the final 16 bits of the block. This PCS is inserted into the packet error control field. The equation for PCS is:

$$\text{PCS} = [x^{16} \times M(x) + x^{(n-16)} \times L(x)] \text{ MODULO } G(x)$$

where:

$M(x)$  is the (n-16)-bit message to be encoded expressed as a polynomial with binary coefficients, n being the number of bits in the encoded message (i.e. the number of bits in the complete packet).

$L(x)$  is the presetting polynomial given by:

$$L(x) = \sum_{i=0}^{15} x^i \text{ (all "1" polynomial of order 15)}$$

$G(x)$  is the CCITT recommendation V.41 (Reference [10]) generating polynomial given by:

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

+

is the modulo 2 addition operator (Exclusive OR).

**NOTE** The encoding procedure differs from that of a conventional cyclic block encoding operation in that the  $x^{(n-16)} \times L(x)$  term has the effect of presetting the shift register to an all ones state (rather than a conventional all zeros state) prior to encoding.

### A.1.3 Decoding procedure

The error detection syndrome,  $S(x)$  is given by:

$$S(x) = [x^{16} \times C^*(x) + x^n \times L(x)] \text{ MODULO } G(x)$$

where:

$C^*(x)$  is the received block in polynomial form.

$S(x)$  is the syndrome polynomial which is zero if no error has been detected.

### A.1.4 Realization of a CRC encoder - decoder

#### A.1.4.1 General

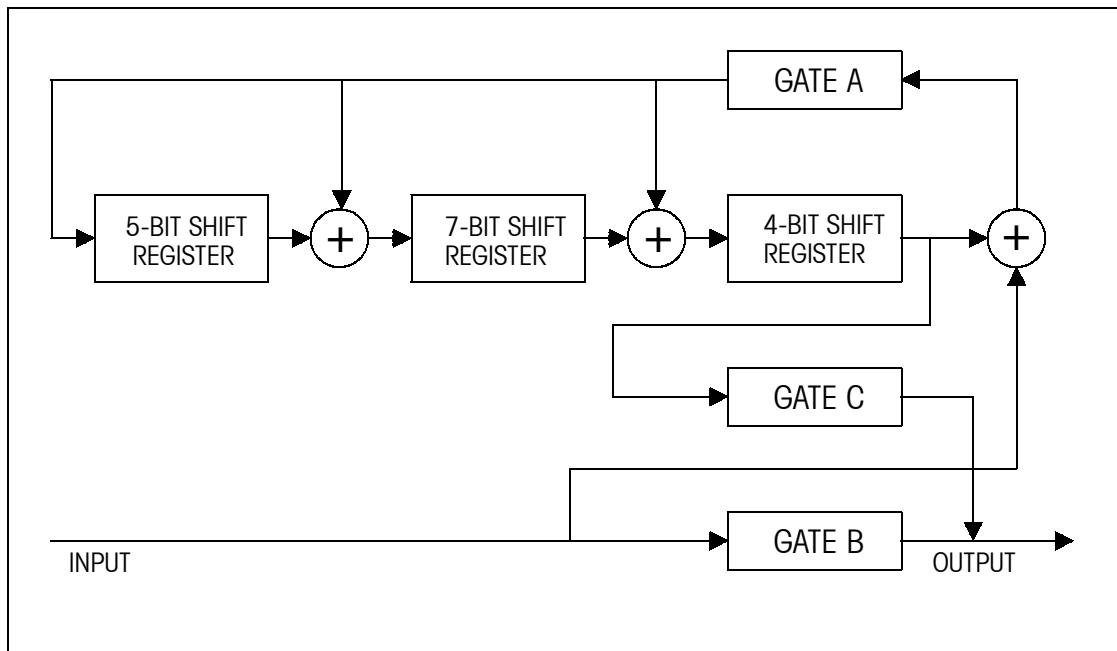
This subclause describes two arrangements, based on a shift register, for encoding and decoding a telemetry source packet (or telecommand packet) according to the packet check sequence procedures defined above.

#### A.1.4.2 Encoder

Figure A-2 shows an arrangement for encoding with the aid of a shift register. To encode, the storage stages are set to “one”, gates A and B are enabled, gate C is inhibited, and (n-16) message bits are clocked into the input. They appear simultaneously at the output.



After the bits have been entered, the output of gate A is clamped to “zero”, gate B is inhibited, gate C is enabled, and the register is clocked a further 16 counts. During these counts, the applicable check bits appear in succession at the output.

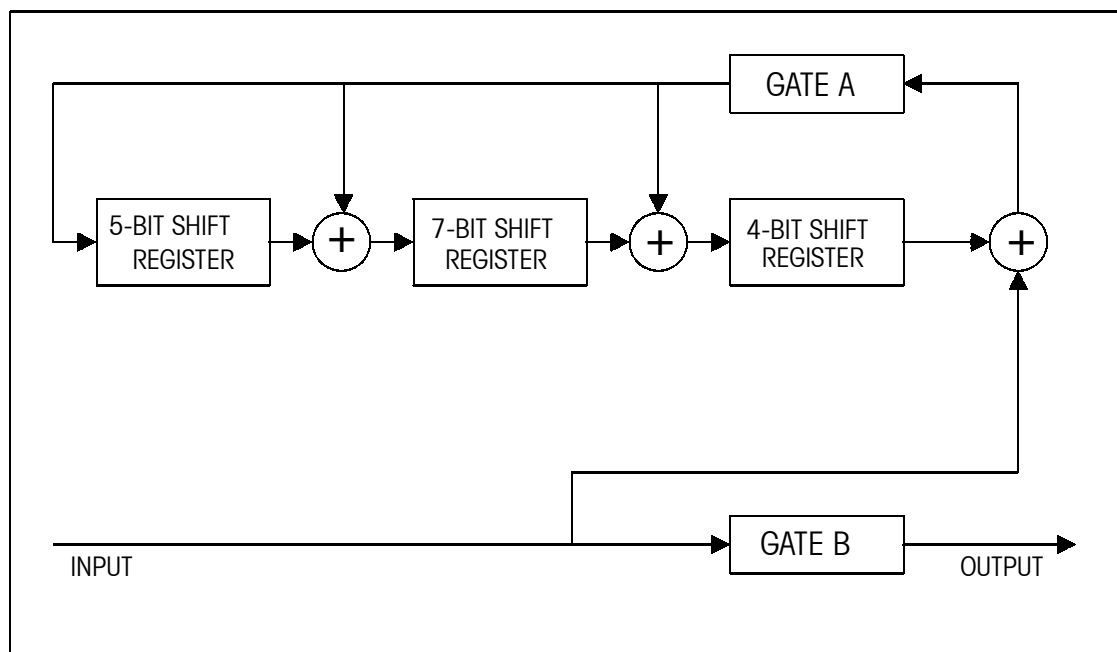


**Figure A-2: Encoder**

#### A.1.4.3 Decoder

Figure A-3 shows an arrangement for decoding with the aid of a shift register. To decode, the storage stages are set to “one” and gate B is enabled.

The received  $n$  bits (i.e. the  $(n-16)$  message bits plus the 16 bits of PCS) are then clocked into the input and after  $(n-16)$  counts gate B is inhibited. The 16 check bits are then clocked into the input and the contents of the storage stages are then examined. For an error-free packet, the contents shall be “zero”. Non-zero contents indicates an erroneous packet.



**Figure A-3: Decoder**

### A.1.5 Verification of compliance

The binary sequences defined in this subclause are provided to the designers of packet systems as samples for early testing, so that they may verify the correctness of their CRC error-detection implementation.

All data are given in hexadecimal notation. For a given field (data or CRC) the leftmost hexadecimal character contains the most significant bit.

Data	CRC
00 00	1D 0F
00 00 00	CC 9C
AB CD EF 01	04 A2
14 56 F8 9A 00 01	7F D5

### A.1.6 Software implementation

#### A.1.6.1 Introduction

In addition to their interesting performance, CRC codes are particularly efficient when it comes to hardware implementation. Software implementation, on the other hand, is more complex.

The following C-language code describes the software routines to implement the CRC encoder. To implement the CRC decoder, the same routines can be used: data and the syndrome are encoded and the resulting syndrome should be equal to zero if no error is present.

#### A.1.6.2 Functions applicable to generate the CRC placed at the end of a packet.

##### a. Crc FUNCTION

The Crc function calculates the CRC for one byte in serial fashion and returns the value of the calculated CRC checksum.

##### b. Crc\_opt FUNCTION

This function can be used instead of the Crc function given above.

The Crc\_opt function generates the CRC for one byte and returns the value of the new syndrome. This function is approximately 10 times faster than the non-optimized Crc function.

##### c. InitLtbl FUNCTION

The InitLtbl function initiates the look-up table used by Crc\_opt.

```

unsigned int Crc(Data, Syndrome)
    unsigned char Data; /* Byte to be encoded */
    unsigned Syndrome; /* Original CRC syndrome */
{
    int i;
    for (i=0; i<8; i++) {
        if ((Data & 0x80) ^ ((Syndrome & 0x8000) >> 8)) {
            Syndrome = ((Syndrome << 1) ^ 0x1021) & 0xFFFF;
        } else {
            Syndrome = (Syndrome << 1) & 0xFFFF;
        }
        Data = Data << 1;
    }
}

```

```

    }
    return (Syndrome);
}

unsigned int Crc_opt (D, Chk, table)
    unsigned char D;          /* Byte to be encoded */
    unsigned int Chk;         /* Syndrome */
    unsigned int table [ ];   /* Look-up table */
{
    return (((Chk << 8) & 0xFF00)^table [(((Chk >> 8)^D) & 0x00FF)]);
}

void InitLtbl (table)
    unsigned int table [ ];
{
    unsigned int i, tmp;
    for (i=0; i<256; i++) {
        tmp=0;
        if ((i & 1) != 0) tmp=tmp ^ 0x1021;
        if ((i & 2) != 0) tmp=tmp ^ 0x2042;
        if ((i & 4) != 0) tmp=tmp ^ 0x4084;
        if ((i & 8) != 0) tmp=tmp ^ 0x8108;
        if ((i & 16) != 0) tmp=tmp ^ 0x1231;
        if ((i & 32) != 0) tmp=tmp ^ 0x2462;
        if ((i & 64) != 0) tmp=tmp ^ 0x48C4;
        if ((i & 128) != 0) tmp=tmp ^ 0x9188;
        table [i] = tmp;
    }
}

/* Simple program to test both CRC generating functions */
void main ( )
{
    unsigned int Chk; /* CRC syndrome */
    unsigned int LTbl[256]; /* Look-up table */
    unsigned char indata[32]; /* Data to be encoded*/
    int j;
    indata[0] = 0x31; indata[1] = 0x23; indata[2] = 0x48; indata[3] = 0x07;
    indata[4] = 0x00; indata[5] = 0xEC; indata[6] = 0xD0; indata[7] = 0x37;
    Chk = 0xFFFF; /* Reset syndrome to all ones */
    for (j=0; j<8; j++) {
        Chk = Crc(indata[j], Chk); /* Unoptimized CRC */
    }
}

```

```

printf(" CRC = %x (should be 0)\n",Chk);
InitLtbl(LTbl); /* Initiate look-up table */
Chk = 0xFFFF; /* Reset syndrome to all ones */
for(j=0;j<8;j++) {
Chk = Crc_opt(indata[j],Chk,LTbl); /* Optimized CRC */
}
printf(" CRC = %x (should be 0)\n", Chk);
}

```

## A.2 Specification of the ISO checksum

### A.2.1 General specification

The standard error detection encoding or decoding procedure, which is described in detail in the following subclauses, produces a 16-bit checksum (the "ISO Checksum") using integer arithmetic (see J.G. Fletcher (Reference [3]) and ISO 8473-1:1998 (Reference [4])).

The ISO checksum procedure can be easily implemented in software on processors using a compact and efficient algorithm. In contrast to the CRC algorithms (see subclause A.1), it does not use a look-up table and does not perform bit-wise operations on the data to be checked. Indeed, it only performs integer arithmetic on the octets of the data to be checked.

This Standard specifies that the ISO checksum procedure shall be used to check the contents of an on-board memory area using the services of the memory management service (see clause 11). All octets of the on-board memory area shall be processed in turn and the calculated ISO checksum value shall be placed in the checksum field of the Memory Check Report.

This Standard also specifies that the ISO checksum procedure can be used to detect errors which have been introduced into a telemetry source packet (or a telecommand packet) during packet processing, transmission or storage activity. All octets of the entire packet including the packet header but excluding the final packet error control field shall be processed in turn and the calculated ISO checksum value shall be placed in the packet error control field.

The error detection properties of the ISO checksum procedure are almost the equal of those of the CRC. The error detection properties can be expressed as follows:

- a. The proportion of all errors in the data that are not detected is approximately  $1,54 \times 10^{-5}$ , i.e. the checksum detects virtually the same proportion of all errors as does the CRC.
- b. A single bit in error shall always be detected.
- c. In contrast to the CRC, an error in the data which affects an odd number of bits is not always detected. However, since the checksum has essentially the same overall detection capability as the CRC, this is compensated by more detections of an error in the data which affects an even number of bits.
- d. An error in the data affecting exactly two bits, no more than 2 040 bits apart, shall always be detected.
- e. The probability that a single error burst spanning 16 bits or less of the data shall not be detected is approximately  $1,9 \times 10^{-7}$ . Not all intermediate bits in the error burst span need be affected.

This probability is non-zero because the algorithm does not detect an error burst which causes 8 consecutive bits to change from all zeros to all ones or vice-versa.

This code is intended only for error detection purposes and no attempt should be made to utilize it for correction.

### A.2.2 Symbols and conventions

$C_0, C_1$  are variables used in the encoding and decoding procedures;  
 $B_i$  is the integer value of the  $i^{\text{th}}$  octet to be checked;  
 $N$  is the number of octets of data to be checked;  
 $CK_1$  is the value of the left most octet of the calculated checksum;  
 $CK_2$  is the value of the right most octet of the calculated checksum.  
 Addition is performed modulo 255 (1's complement arithmetic).

### A.2.3 Encoding procedure

The encoding procedure accepts  $N$  octets of data to be checked and generates a systematic 16-bit checksum value. This checksum value is placed in the destination field (e.g. the packet error control field).

The algorithm is:

- a. Initialize  $C_0$  and  $C_1$  to zero.
- b. Process each octet of the data to be checked sequentially from  $i = 1$  to  $N$  as follows:

$$C_0 = C_0 + B_i$$

$$C_1 = C_1 + C_0$$

On completion, we have:

$$C_0 = \sum_{i=1}^N B_i$$

$$C_1 = \sum_{i=1}^N (N - i + 1) * B_i$$

- c. Calculate an intermediate ISO checksum value as:

$$CK_1 = - (C_0 + C_1)$$

$$CK_2 = C_1$$

- d. If  $CK_1 = 0$ , then  $CK_1 = 255$ .
- e. If  $CK_2 = 0$ , then  $CK_2 = 255$ .
- f. Place the resulting values of  $CK_1$  and  $CK_2$  in their destination fields.

### A.2.4 Decoding procedure

The decoding procedure accepts  $N+2$  octets of data to be checked and reports if an error is detected or not.

The  $N+2$  octets consist of:

- $N$  octets of data to be checked;
- the 2 checksum octets (e.g. the packet error control field) which are appended to the  $N$  octets of data.

The algorithm is:

- a. If either, but not both, checksum octets contains the value zero, then report Error-Detected.
- b. Initialize  $C_0$  and  $C_1$  to zero.
- c. Process each octet of the data to be checked sequentially from  $i = 1$  to  $N+2$  by
 
$$C_0 = C_0 + B_i$$

$$C_1 = C_1 + C_0$$
- d. When all of the octets have been processed, if the values  $C_0$  and  $C_1$  are both zero, then report No-Error-Detected; otherwise report Error-Detected.

### A.2.5 Verification of compliance

The binary sequences defined in this subclause are provided to the designers as samples for early testing, so that they can verify the correctness of their ISO Checksum error-detection implementation.

All data are given in hexadecimal notation. For a given field (data or ISO Checksum) the leftmost hexadecimal character contains the most significant bit.

Data	ISO Checksum
00 00	FF FF
00 00 00	FF FF
AB CD EF 01	9C F8
14 56 F8 9A 00 01	24 DC

## A.3 Use of service type 2, device command distribution

The following examples show how OLYMPUS Mode 5 commands (distributed via the OBDH Bus) can be accommodated within PUS service type 2 (Device Command Distribution):

- a. High Level On-Off command, originally implemented as:

RCC Address	0 0 0 0 0 0 0 0 0 1 1	Channel Address	0
5 bits	11 bits	7 bits	1 bit

Implemented as a service type 2, subtype 1 request:

N=1	RCC Address	0 0 0 0 0 0 0 0 0 1 1	Channel Address	0
Unsigned Integer	5 bits	11 bits	7 bits	1 bit
	3 octets unsigned integer (Address)			

- b. Register load command (previously called memory load), originally implemented as:

RCC Address	ML Address	ML Data 0 - 7	ML Data 8 - 15
5 bits	3 bits	8 bits	8 bits

Implemented as a service type 2, subtype 2 request:

N=1	RCC Address	ML Address	ML Data 0 - 7	ML Data 8 - 15
Unsigned Integer	5 bits	3 bits	8 bits	8 bits
	1 octet unsigned integer (Address)		2 octets	

## Annex B (normative)

---

### Mission constants

The following set of telemetry and telecommand related constants shall be assigned values for each mission:

**<APPL\_TIME\_CODE>**

This mission constant identifies the presence or absence of the telemetry source packet time field as well as the time format (CUC or CDS) and the encoding of the time field. There is one such mission constant for each on-board application process.

**<CPDU\_DURATION\_UNIT>**

The pulse duration unit defined for the CPDU, which can be any value between 10 ms and 15 ms. The actual pulse duration for a given CPDU instruction is expressed as a multiple of this value.

**<CPDU\_MAX\_INSTR>**

The maximum number of command pulse instructions that can be contained within a CPDU telecommand packet (at least 12 and at most 120).

**<DIAG\_MIN\_INTERV>**

The minimum sampling interval for the on-board sampling of parameters in diagnostic mode.

**<MISSION\_TIME\_CODE>**

This mission constant defines the value of the P-field for the time report packet, where this is not contained explicitly within the "Satellite Time" field of that packet.

**<MONLIST\_MAX\_CHECKS>**

The maximum number of limit pairs or expected values which can be specified for the on-board monitoring of a parameter. This mission constant shall be specified separately for each application process which supports on-board parameter monitoring.

**<MONLIST\_MAX\_PARAMS>**

The maximum number of parameters which can be monitored at any given time by an on-board parameter monitoring function. Also therefore, the maximum number of parameters which can be added to, or deleted from, the monitoring list within a single telecommand packet. This mission constant shall be specified separately for each application process which supports on-board parameter monitoring.

## &lt;PARAM\_ABS\_SAMPL\_TIME&gt;

The accuracy to which the absolute (on-board) sampling time of a telemetry parameter can be determined.

## &lt;PARAM\_REL\_SAMPL\_TIME&gt;

The accuracy to which the relative sampling time of any two parameters, which can be telemetered in different packets can be determined.

## &lt;PKT\_STORAGE\_TIME&gt;

The time for which telemetry source packets shall be stored on-board, for later dumping to ground, over and above the longest time interval without ground coverage. Only applicable for missions with discontinuous ground coverage.

## &lt;PKTS\_NUM\_STORED&gt;

The number of event-generated packets stored on-board for subsequent retrieval from ground, for missions with continuous ground coverage.

## &lt;PSLIST\_MAX\_PARAMS&gt;

The maximum number of parameters whose statistical values can be evaluated at any given time by an on-board application process. Also therefore, the maximum number of parameters which can be added to, or deleted from, the parameter statistics list within a single telecommand packet. This mission constant shall be specified separately for each application process which supports the parameter statistics reporting service.

## &lt;SMALLEST\_ADDRESSABLE\_UNIT&gt;

The smallest unit that a given on-board processor can address, hence there are as many of these mission constants as there are on-board processors. This unit can be an 8- (octet) 16-, 24- 32- or 64-bit word. The base for the memory management service is referred to this "Smallest Addressable Unit (SAU)".

## &lt;TCPKT\_MAX\_LENGTH&gt;

The maximum length of a telecommand packet, which can be less than the maximum length defined by the CCSDS Recommendations.

## &lt;TC\_CHECKSUM\_TYPE&gt;

The type of checksum used for checking the integrity of telecommand packets for the given application process. This shall be either an ISO standard checksum or a Cyclic Redundancy Check (CRC).

## &lt;TMPKT\_MAX\_LENGTH&gt;

The maximum length of a telemetry source packet, which can be less than the maximum length defined by the CCSDS Recommendations.

## &lt;TM\_CHECKSUM\_TYPE&gt;

This indicates the presence and (if present) the type of checksum used for checking the integrity of telemetry source packets for the given application process. The type shall be either an ISO standard checksum or a Cyclic Redundancy Check (CRC).



## Annex C (normative)

# Spacecraft time protocols

### C.1 Introduction

The text in this annex has been transcribed from ESA PSS-04-106 Packet Telemetry Standard, Issue 1, January 1998.

All spacecraft shall maintain a spacecraft reference time, called “Spacecraft Elapsed Time”, for the purpose of time-tagging telemetered events. The spacecraft clock is a free-running counter from an arbitrary starting point (epoch). The corresponding absolute time in the UTC (Universal Time Coordinated) reference frame is derived by ground correlation of the spacecraft clock. This correlation information is normally distributed, on the ground, together with the source packets. It can also be re-transmitted to the spacecraft, if UTC is used by the spaceborne data systems as an additional service.

The three subsections which follow cover the following fields:

- Presentation of spacecraft elapsed time.
- Format of standard spacecraft time source packet.
- Spacecraft time-correlation procedures.

### C.2 Presentation of spacecraft elapsed time

The spacecraft elapsed time is maintained by the central clock on board the spacecraft and made available to all users. The time code format used is the CCSDS Unsegmented Time Code (CUC) format, which is represented as a combination of a preamble field (P-field) and a time specification field (T-field). The P-field may be either explicitly or implicitly conveyed with the spacecraft time code. If it is explicitly conveyed, it shall immediately precede the T-field.

The general format of both fields is as follows:

- P-field (1 octet): spacecraft time format specification.
- T-field (1 to 7 octets): spacecraft time specification.

The spacecraft elapsed time shall be represented as an unsegmented binary count of seconds and binary powers of subseconds, counting from an arbitrary epoch. The

resolution can be  $2^0$ ,  $2^{-8}$ ,  $2^{-16}$  or  $2^{-24}$  seconds. The time shall be formatted as follows:

- a. P-Field:
  - bits 0 – 3 = 0010
  - bits 4 – 5 = no. of octets of coarse time, minus one, i.e. 1 to 4 octets.
  - bits 6 – 7 = no. of octets of fine time, i.e. 0 to 3 octets.
- b. Complete spacecraft time:

P-Field	T-Field							
00100000								
00100001								
00100010								
00100011								
00100100								
00100101								
00100111								
00100111								
00101000								
00101001								
00101010								
00101011								
00101100								
00101101								
00101110								
00101111								
	$2^{32}$	$2^{24}$	$2^{16}$	$2^8$	$2^0$	$2^{-8}$	$2^{-16}$	$2^{-24}$

- c. Unit of time: TAI (Temps Atomique International) second.
- d. Epoch: arbitrary, i.e. the epoch is of no importance, since elapsed time samples are UTC correlated on ground (see following sections).

If spacecraft time information is used as ancillary data for the interpretation of the source packet data, then time shall be inserted in the data field header of the source packet data field.

The presence of time in a data field header shall be application dependent. The time code format used can be either that of the spacecraft elapsed time (CUC) or that of the spacecraft UTC (CCSDS Day Segmented time code (CDS)) when this additional service is provided on board the spacecraft.

The P-field shall be included if application-specific interpretation of the T-field is mandated, otherwise it is optional.

The length of the T-field is application dependent, as defined by the user. It can be smaller than the distributed central spacecraft time. If it is longer, the additional clock octets shall be generated by the on-board user. In all cases, the length of the T-field shall be sufficient to avoid ambiguity during the mission, and the resolution shall be sufficient to meet the requirements of interpreting the data without additional processing of the time information.

### C.3 Standard spacecraft time source packet

The standard source packet used to transport the regular spacecraft elapsed time samples to ground for time correlation with UTC is formatted as follows:

- Application process identifier: “all zeros” (reserved APID).
- Data field header flag: set to “0”.
- Packet data field: the data field of the standard source packet only contains spacecraft time system information formatted as follows:
  1. The first octet in the data field shall specify the sampling rate of the spacecraft clock contents as explained further in subclause C.4. The format and specification of this octet is as follows:
 

Bits 0, 1, 2, 3: Not specified in this Standard (to be obtained from the supporting infrastructure or to be set to ‘all zeros’ if not used).

Bits 4, 5, 6, 7: Specify the sampling rate of the spacecraft clock contents as described in subclause C.4. The code allocation is as follows:

Bit 4	Bit 5	Bit 6	Bit 7	Rate (in Frames)
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256

2. The next octets contain spacecraft time information in the form of a P-field and a T-field, with length and contents as described in subclause C.2.
3. The last octets contain any relevant spacecraft time-system parameters. This includes the provision of system configuration flags for the identification of the various fixed, on-board delays. The format for this part of the packet data field shall be obtained from the supporting infrastructure.

### C.4 Spacecraft time correlation procedures

On board the spacecraft, the contents of the spacecraft elapsed time clock are sampled at the instant of occurrence of the leading edge of the first bit of the attached synchronisation marker of that telemetry transfer frame of virtual channel “0” with a virtual channel frame count “0”. This time sample shall then be placed into the standard spacecraft time source packet and telemetered to ground before the frame counter of virtual channel “0” has counted 255 more frames, so as to avoid ambiguity.

Should this sampling rate (intervals of 256 frames of virtual channel “0”) prove too low for the mission requirements, it is permissible to sample the spacecraft clock contents at intervals of 128, 64, 32, 16, 8, 4, 2 or 1 frame(s) of virtual channel “0”, by choice. Consequently, the time sample shall be telemetered to ground before the selected number of frames of virtual channel “0” have elapsed (128, 64, 32, 16, 8, 4, 2 or 1, all counts starting from virtual channel frame count “0”).

The ground data capture system shall:

- a. accurately time-tag the instant of reception of the same first bit of the attached synchronisation marker of the virtual channel “0” transfer frame

with a virtual channel frame count “0”. The time standard used is CDS-coded UTC;

- b. extract the standard spacecraft time source packet and collect the CUC-coded time sample.

Thus, a correlation between the spacecraft elapsed time and the UTC reference on ground shall be established, which can be used:

- on ground, to transform the spacecraft elapsed time information contained in the source packets into UTC information;
- on board the spacecraft, to achieve directly the same service as on the ground.

## Annex D (normative)

# Command pulse distribution unit

### D.1 General requirements

The text in this annex has been transcribed from ESA PSS-04-151 Telecommand decoder specification, Issue 1, September 1993.

The requirements that lead to the definition of the CPDU are determined by the operational environment of the free-flying spacecraft: it is an automated vehicle that cannot be reached by human crews and repaired by them in case of failure or severe anomaly.

In most instances, the human operators on ground may rely on the nominal telemetry and telecommand functions of the spacecraft Data Management System (DMS): telemetry supervisory data are available, the telecommand Sequence-Controlled Service (i.e. the AD Service) can be used, and the data distribution services of the DMS are functional.

However, if and when the DMS fails, these operators can be confronted with a complete lack of nominal telemetry and telecommand functions: there is no telemetry, no data distribution services can be expected on board the spacecraft, and only the telecommand Expedited Service (i.e. the BD Service) can be used. Nevertheless, the spacecraft operators shall be provided with the means to reconfigure the DMS, using whatever redundancy is available on board the spacecraft. This is where the CPDU comes into play.

The CPDU is a simple unit that is solely accessible from ground. It can have the capability to operate command pulse lines that are also operated through the DMS (for instance: because of a requirement for the delayed (time-tagged) execution of some critical command): there is no contradiction in this, even if it leads to the apparent duplication of command pulse lines driving certain “actuators” (e.g. relays). The CPDU can also be used to “authorize” the DMS to access potentially dangerous “actuators” at certain given times during a mission (e.g. for the delayed execution of the critical command already mentioned above).

The CPDU is identified by the application process identifier placed in the telecommand packet header. Application process identifiers are mission-specified, and shall be specific to each application process on board a given spacecraft. Therefore, the application identifier of each CPDU shall be programmable.

## D.2 Specification

### D.2.1 Checking the telecommand segment

Notwithstanding cross-coupling of redundant CPDUs, the CPDU receives telecommand segments (each segment containing a complete telecommand packet) from one MAP interface that responds to only one MAP identifier. In theory, the MAP identifier can be any of the possible 64. In practice, the MAP identifier assigned to that CPDU is MAP 0.

When CPDUs are cross-coupled, the identifier of the second MAP interface may be any value other than MAP 0. In practice, MAP 1 is a likely choice.

When it receives the telecommand segment, the CPDU stores it for further processing of the telecommand packet it contains, unless it has already received one such telecommand packet and not completed the execution of the command instructions it contained down to the last command pulse: in which case, it shall ignore any incoming telecommand segment, whether it was transferred in an AD or a BD transfer frame. This is important: there is no packetization layer “abort” command for the CPDU. Once it has accepted a telecommand packet, the CPDU cannot release it until all command instructions specified in that packet have been executed.

#### IMPORTANT NOTE:

The above is not the same as the “Aborted Data Transfer” (ADT) signal provided by the MAP interface. The ADT signal is used to indicate that a BD frame has arrived, and that this has resulted in telecommand segment data being erased (telecommand segment data that were resident in the storage device of the segmentation layer: the “back-end” buffer). In such a case, any incompletely transferred telecommand segment data shall be erased. This also applies to the CPDU.

The CPDU is a simple unit: no Packet Assembly Controller (PAC) is needed. All telecommand packets shall be carried inside a single telecommand segment and, therefore, a single telecommand transfer frame (because of the BD service requirement).

The 6-bit MAP identifier of the telecommand segment header is ignored (this function is already provided by the MAP interface allocated to the CPDU).

### D.2.2 Checking the CPDU-specific telecommand packet

The telecommand packet specified to operate a CPDU shall have a minimum (total) length of 10 octets, and a maximum length of 248 octets. It shall always be made of an even number of octets.

The last two octets of the packet shall be used for error detection: they shall contain a 16-bit CRC identical to that used in the telecommand transfer frame. As for the transfer frame, it is used to detect errors over the complete packet structure.

When the telecommand segment segmentation flags have been checked to be 11, the actual number of octets making up the telecommand packet shall be verified to be (a) consistent with the packet length field and (b) an even number between 10 octets and the maximum value supported by the particular implementation (see design requirements, in subclause D.3).

After this check has been passed, the position of the CRC being known, an error check over the full packet takes place. If the packet is found to be error-free (“CLEAN”), the process continues. Otherwise, the complete telecommand packet is erased.

After the telecommand packet has been found “CLEAN”, the following fields are checked:

- Version number: shall be 000
- Type bit: shall be 1
- Data field header flag (CCSDS term: Secondary header flag): shall be 0
- Application process identifier: the 11 bits shall be as programmed for the particular CPDU
- Packet sequence flags: shall be 11
- Packet name or sequence count: not verified, only telemetered back to ground
- Packet length: already checked by the “CLEAN” verification process

When all the above checks have been passed successfully, the telecommand packet is declared “LEGAL” and its application data (command instructions) read out and executed as described in the next subclause.

If any of the checks fails, the packet is erased.

### D.2.3 Processing the application data

The application data of the CPDU consist of at least one command instruction in the form of one double octet, or several of such double-octet command instructions, up to the maximum supported by the particular implementation.

Each double octet is formatted as follows:

- First octet: specifies one of 256 command pulse outputs.
- Second octet: specifies the duration of the command pulse to be issued on the specified output, as follows:
  - Bits 0 to 4 (5 MSBs) are reserved for future use. They are normally set to all zeros by the sending end. They are IGNORED by the CPDU.
  - Bits 5 to 7 (3 LSBs) specify the duration of the pulse as follows:
 

000 =	$1 \times D$ , where D (for “Duration”) is a fixed value to be selected by the implementer, and which can be any value between 10 and 15 ms.
001 =	$2 \times D$
010 =	$4 \times D$
011 =	$8 \times D$
100 =	$16 \times D$
101 =	$32 \times D$
110 =	$64 \times D$
111 =	$128 \times D$ (i.e. a value between 1,28 and 1,92 seconds)

Where there is more than one command instruction in the packet, each instruction shall be executed one after the other, in the original sequence.

A telecommand packet may happen to contain command instructions that correspond to outputs that have not been physically implemented (a possible instance during testing). In such a case, the CPDU is not expected to detect this anomaly, and behaves as if the output existed, i.e. there is no report of an anomaly by the CPDU (the anomaly is expected to be detected at a higher level, where the lack of response or effect is reported).

## D.3 Design requirements

### D.3.1 Application process identifier

The application process identifier belongs to the category of “Mission-Programmable Data”. For each CPDU, it shall be possible to easily program (or re-program) the 11-bit application process identifier.

### D.3.2 Maximum capability of the CPDU

The command pulse output circuitry of the “basic” CPDU shall support the number of discrete pulse outputs for a specific mission, up to a maximum of 256.

As regards the minimum and maximum lengths of a CPDU-specific telecommand packet, they are specified in subclause D.2.2 to be 10 and 248 octets respectively. The designer shall select the maximum capacity of his CPDU: this is the size of the largest telecommand packet his CPDU can accept. This shall be selected between the following two capacity values:

- Smallest capacity: packet length of 32 octets (i.e. a capability of 12 command instructions)
- Largest capacity: packet length of 248 octets (i.e. a capability of 120 command instructions)



---

## Bibliography

- [1] CCSDS 100.0-G-1, *Telemetry: Summary of Concept and Rationale, Green Book, December 1987*
- [2] CCSDS 200.0-G-6, *Telecommand: Summary of Concept and Rationale, Green Book, January 1987*
- [3] J.G. Fletcher, *An Arithmetic Checksum for Serial Transmission, IEEE Transaction on Communication, Vol. COM-30, No.1, January 1982*
- [4] ISO 8473-1:1998, *Information Technology - Protocol for Providing the Connectionless-Mode Network Service: Protocol specification second edition*
- [5] ISO 8824, *Information Technology - Abstract Syntax Notation One (ASN.1)*
- [6] ISO 8825, *Information Technology - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*
- [7] IEEE 754:1985, *Standard for Binary Floating-Point Arithmetic, IEEE Computer Society Document R(1991)*
- [8] MIL-STD-1750a, *Military Standard Sixteen-Bit Computer Instruction Set Architecture, 2<sup>nd</sup> July 1980*
- [9] ANSI X3.4, *Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-bit ASCII), American National Standards Institute, 1986*
- [10] ITU-T V.41, *Code-Independent Error Control System - Data Communication over the Telephone Network, 1989*  
(before renaming known as *Information Technology - CCITT V.41*)
- [11] ECSS-M-00-02, *Space project management - Tailoring of space standards*

*(This page is intentionally left blank)*

<b>ECSS Document Improvement Proposal</b>		
<b>1. Document I.D.</b> ECSS-E-70-41A	<b>2. Document date</b> 30 January 2003	<b>3. Document title</b> Ground systems and operations — Telemetry and telecommand packet utilization
<b>4. Recommended improvement</b> (identify clauses, subclauses and include modified text or graphic, attach pages as necessary)		
<b>5. Reason for recommendation</b>		
<b>6. Originator of recommendation</b>		
Name:	Organization:	
Address:	Phone: Fax: e-mail:	<b>7. Date of submission:</b>
<b>8. Send to ECSS Secretariat</b>		
Name: W. Kriedte ESA-TOS/QR	Address: ESTEC, P.O. Box 299 2200 AG Noordwijk The Netherlands	Phone: +31-71-565-3952 Fax: +31-71-565-6839 e-mail: Werner.Kriedte@esa.int

**Note:** The originator of the submission should complete items 4, 5, 6 and 7.

An electronic version of this form is available in the ECSS website at: <http://www.ecss.nl/>  
At the website, select "Standards" - "ECSS forms" - "ECSS Document Improvement Proposal"

*(This page is intentionally left blank)*