ESA PSS-07-101 Issue 1
May 1994

european space agency
agence spatiale européenne

Packet utilisation standard

Prepared by:
ESA Committee for Operations and EGSE Standards (COES)

Approved by:
The Inspector General, ESA

ESA PSS-07-101 Issue 1
May 1994

Packet utilisation standard

Prepared by:
ESA Committee for Operations
and EGSE Standards (COES)

Approved by:
The Inspector General, ESA

european space agency / agence spatiale européenne
8-10, rue Mario-Nikis, 75738 PARIS CEDEX, France

Published by ESA Publications Division,
ESTEC, Noordwijk, The Netherlands.

Printed in the Netherlands.

Price: 80 Dutch Guilders.

ISSN 0379 - 4059

Abstract

The ESA Packet Telemetry and Telecommand Standards (ESA PSS-04-106 and ESA PSS-04-107) and the CCSDS recommendations from which they are derived address the end-to-end transport of telemetry and telecommand data between user applications on the ground and application processes on board a satellite and the intermediate transfer of these data through the different elements of the ground and space segments.

The purpose of this Packet Utilisation Standard (PUS) is to complement the Packet Telemetry and Telecommand 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.

Document Change Record

Issue number and date	Sections affected	Remarks
Issue 1 May 1994	All	New document

Contents

Applicable documents ix
Reference documents x

1. Introduction 1
   1.1 Purpose 1
   1.2 Scope 2
   1.3 Structure of the standard 3
   1.4 Applicability 3

2. Context of the PUS 4

3. Operational requirements 7
   3.1 Introduction 7
   3.2 Telecommanding 9
   3.3 Telecommand transmission and distribution 12
   3.4 Telecommand verification 12
   3.5 Satellite status reporting 16
   3.6 Telemetry timing 21
   3.7 Memory management 22
   3.8 Software management 23
   3.9 On-board scheduling 24
   3.10 On-board monitoring 27
   3.11 Large data transfer 29
   3.12 Telemetry generation and transmission control 30
   3.13 On-board storage and retrieval 31
   3.14 On-board traffic management 33
   3.15 Packet routing 33
   3.16 Testability 34

4. Service specification 35
   4.1 Introduction 35
   4.2 Definitions 36
   4.3 Conventions 37
   4.4 Telecommand packet structure 39
   4.5 Telemetry source packet structure 43
   4.6 Standard services 46

5. Telecommand verification service 47
   5.1 Scope 47
   5.2 Service concept 47
   5.3 Service requests and reports 48
   5.4 Capability sets 50

6. Device command distribution Service 51
   6.1 Scope 51
   6.2 service concept 51
   6.3 Service requests and reports 53
   6.4 Capability sets 55

7. Housekeeping and diagnostic data reporting service 56
7.1 Scope 56
7.2 Service concept 56
7.3 Service requests and reports 58
7.4 Capability sets 65

8. Parameter statistics reporting service 68
8.1 Scope 68
8.2 Service concept 68
8.3 Service requests and reports 69
8.4 Capability sets 73

9. Event reporting service 74
9.1 Scope 74
9.2 service concept 74
9.3 Service requests and reports 74
9.4 Capability sets 75

10. Memory management service 76
10.1 Scope 76
10.2 Service concept 76
10.3 Service requests and reports 77
10.4 Capability sets 83

11. Task management service 84
11.1 Scope 84
11.2 Service concept 84
11.3 Service requests and reports 85
11.4 Capability sets 88

12. Function management service 89
12.1 Scope 89
12.2 Service concept 89
12.3 Service requests and reports 90
12.4 Capability sets 91

13. Time management service 93
13.1 Scope 93
13.2 Service concept 93
13.3 Service requests and reports 94
13.4 Capability sets 94

14. Time reporting service 95
14.1 Scope 95
14.2 Service concept 95
14.3 Service requests and reports 95
14.4 Capability sets 96

15. On-board scheduling service 97
15.1 Scope 97
15.2 Service concept 97
15.3 Service requests and reports 102
15.4 Capability sets 114

16. On-board monitoring service 116
16.1 Scope 116
16.2 Service concept 116
16.3 Service requests and reports 120
16.4 Capability sets 131

17. Large data transfer service 132
17.1 Scope 132
17.2 Service concept 132
17.3 Service requests and reports 136
17.4 Capability sets 141

18. Packet transmission control service 143
18.1 Scope 143
18.2 Service concept 143
18.3 Service requests and reports 144
18.4 Capability sets 149

19. On-board storage and retrieval service 151
19.1 Scope 151
19.2 Service concept 151
19.3 Service requests and reports 154
19.4 Capability sets 161

20. On-board traffic management service 163
20.1 Scope 163

21. Test service 164
21.1 Scope 164
21.2 Service concept 164
21.3 Service requests and reports 164
21.4 Capability sets 165

22. Summary of service requests and reports 166
22.1 Introduction 166

23. Parameter types and structure rules 171
23.1 Introduction 171
23.2 Conventions 171
23.3 Encoding formats of parameter types 172
23.4 Tailoring of packet structures for a mission 172
23.5 Simple parameter types 174
23.6 Structured field types 181

Appendix A EXAMPLES 188

Appendix B MISSION PARAMETERS 202

Appendix C GLOSSARY OF TERMS 205

Appendix D GLOSSARY OF ABBREVIATIONS 209

List of Figures

Figure 3 – 1 Example of a telecommand execution profile 12
Figure 3 – 2 Telecommand system layers 14
Figure 4 – 1 Telecommand packet fields 39
Figure 4 – 2 Telemetry source packet fields 43
Figure 11 – 1 State transition diagram for a software task 85
Figure 15 – 1 The relation between execution result and interlock status 100
Figure 16 – 1 Parameter check definitions and check states 118
Figure 17 – 1 The splitting of a Service Data Unit into parts to be downlinked (uplinked) 133
Figure 19 – 1 An example of a centralised storage and retrieval approach 152
Figure A.2 – 1 Standard packet check sequence generation 192
Figure A.2 – 2 Encoder 194
Figure A.2 – 3 Decoder 195

List of Tables

Table 3-1 Telecommand acceptance and execution telemetry 13
Table 4-1 Standard services specified within this document 46
Table 5-1 Summary of telecommand verification service minimum capabilities . 50
Table 5-2 Summary of telecommand verification service additional capabilities . 50
Table 6-1 Summary of device command distribution service minimum capabilities 55
Table 6-2 Summary of device command distribution service additional capabilities 55
Table 7-1 Summary of housekeeping and diagnostic data reporting minimum capabilities 65
Table 7-2 Summary of report definitions control additional capabilities 66
Table 7-3 Summary of report definitions reporting additional capabilities 66
Table 7-4 summary of sampling time offset reporting additional capabilities 66
Table 7-5 Summary of filtered reports minimum capabilities 67
Table 7-6 Summary of filtered reports additional capabilities 67
Table 8-1 Summary of parameter statistics reporting minimum capabilities 73
Table 8-2 Summary of parameter statistics reporting additional capabilities 73
Table 9-1 Summary of event reporting minimum capabilities 75
Table 10-1 Summary of memory management service additional capabilities 83
Table 11-1 Summary of task management service additional capabilities 88
Table 12-1 Summary of function management service additional capabilities 92
Table 13-1 Summary of time management minimum capabilities 94
Table 14-1 Summary of time reporting minimum capabilities 96
Table 15-1 Decision table for the release status of a telecommand 99
Table 15-2 Summary of on-board scheduling minimum capabilities 114
Table 15-3 Summary of on-board scheduling additional capabilities 115
Table 16-1 Summary of on-board monitoring service minimum capabilities 131
Table 16-2 Summary of on-board monitoring service additional capabilities 131

Table 17 – 1 Summary of sending subservice (downlink) minimum capabilities 141
Table 17 – 2 Summary of receiving subservice (uplink) minimum capabilities 142
Table 17 – 3 Summary of sending subservice (downlink) additional capabilities 142
Table 17 – 4 Summary of receiving subservice (uplink) addtional capabilities 142
Table 18 – 1 Decision table for the transmission status of a packet 144
Table 18 – 2 Summary of packet transmission control minimum capabilities 149
Table 18 – 3 Summary of packet transmission control additional capabilities 150
Table 19 – 1 Summary of packet selection subservice minimum capabilities 161
Table 19 – 2 Summary of storage and retrieval subservice minimum capabilities 161
Table 19 – 3 Summary of packet selection subservice additional capabilities 161
Table 19 – 4 Summary of storage and retrieval subservice additional capabilities 162
Table 21 – 1 Summary of test service minimum capabilities 165
Table 22 – 1 Summary of requests and reports for PUS standard services 166

Applicable Documents

1. **Packet telemetry standard**
   ESA PSS-04-106, Issue 1, January 1988

2. **Telemetry channel coding standard**
   ESA PSS-04-103, Issue 1, September 1989

3. **Packet telecommand standard**
   ESA PSS-04-107, Issue 2, April 1992

4. **Time code formats and procedures standard**
   ESA PSS-04-202, not available before mid 1995; **[AD1]** and **[RD2]** cover the topic until then.

5. **Radio frequency and modulation standard**
   ESA PSS-04-105, Issue 1, December 1989

6. **System safety requirements for ESA space systems and associated equipment**
   ESA PSS-01-40, Issue 2, September 1988

7. **Telecommand decoder specification**
   ESA PSS-04 151, Issue 1, September 1993

Where an Applicable Document is quoted within the body of this document, the abbreviation AD is used, together with the number of the document in the above list (e.g. the 'ESA Packet Telemetry Standard' is referred to as **[AD1]**).

Note: the reader is expected to have a good knowledge of all Applicable Documents, in particular of **[AD1]** and **[AD3]**.

Reference Documents

1. **Telecommand: Part 3, Data Management Service**
CCSDS 203.0 – B – 1, Blue Book, January 1987

2. **Time Code Format**
CCSDS 301.0 – B – 1, Blue Book, January 1987

3. **Mission Data Utilisation Standards**
ESA PSS-07-10, Draft 4, May 1992

4. **Space Segment Operability Requirements**
ESA PSS-07-501, to be issued.

5. **Packet Data Description standard**
ESA PSS-07-201, to be issued.

6. **Safety Policy and Requirements for Payloads using STS**
NASA NSTS 1700.7B, Basic Issue, January 1989

7. **An Arithmetic Checksum for Serial Transmission**
J.G. Fletcher, IEEE Transactions on Communication, **COM-30**(1), January 1982

8. **Information Technology – Protocol for Providing the Connectionless-mode Network Service**
ISO/IEC 8473-1, Second Edition, 1994

Where a Reference document is quoted within the body of this document, the abbreviation RD is used, together with the number of the document in the above list (e.g. the 'ESA Mission Data Utilisation Standards' is referred to as **[RD3]**).

PAGE INTENTIONALLY LEFT BLANK

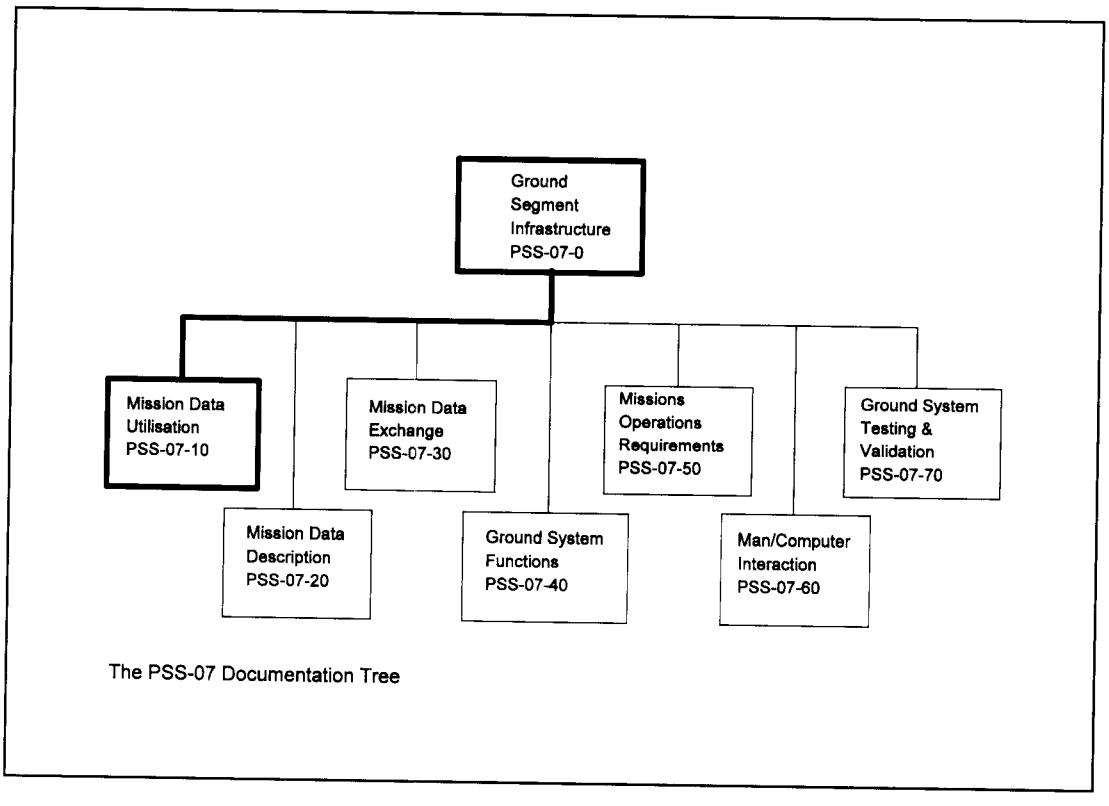
1 Introduction

1.1 Purpose

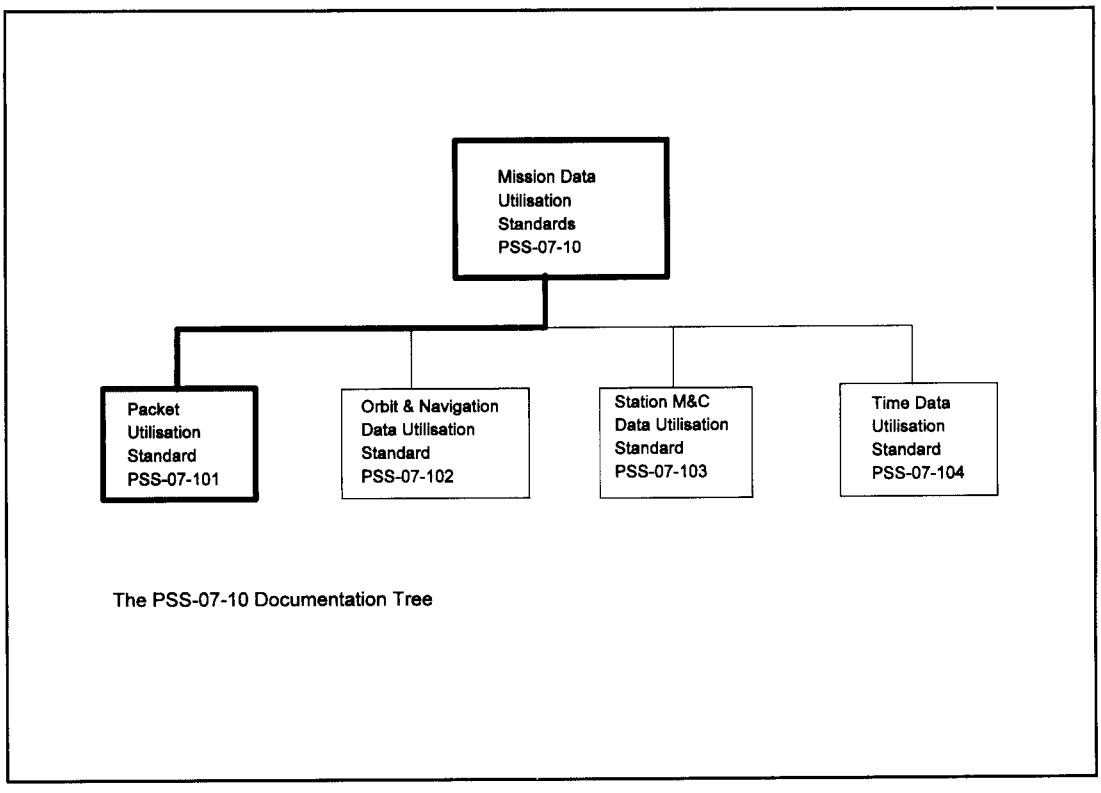
The ESA Packet Telemetry and Telecommand Standards (**[AD1]** and **[AD3]**), and the CCSDS Recommendations (e.g. **[RD1]**) from which they are derived, address the end-to-end transport of telemetry and telecommand data between user applications on the ground and Application Processes onboard the satellite, and the intermediate transfer of these data through the different elements of the ground and space segments.

The purpose of this Packet Utilisation Standard (PUS) is to complement the Packet Telemetry and Telecommand 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 document falls hierarchically under the higher level document "MIssion Data Utilisation Standards" **[RD3]**; the applicable sections of the PSS-07 standards tree are shown in the next two figures.



The PSS-07 Documentation Tree



The PSS-07-10 Documentation Tree

1.2 Scope

This standard relates to the ***utilisation*** of telecommand packets and telemetry source packets for the purposes of monitoring and remote control of subsystems and payloads. In the case of manned missions, this shall also apply to monitoring and control functions exercised onboard.

The utilisation of telecommand and telemetry source packets is defined in the context of ***PUS Services***, which correspond to particular monitoring and control functions (or sets of closely related functions). This standard covers the known requirements of present and future missions. It will, however, be possible to introduce additional Services and/or sub-Services, as the need arises.

Whilst a given mission may be implemented as a distributed system comprising several cooperating onboard Application Processes and applications on the ground, this standard does not define the protocols or data structures which may be used for communication between onboard Application Processes.

Also, no attempt is made herein to standardise the data content of payload/science packets, since this is generally mission-specific in nature. However, payload/science product packets shall conform with this standard, with the data field header being mandatory.

Audio and video data are not contained within either telecommand or telemetry source packets, but are multiplexed at frame level. These data classes are therefore beyond the scope of this standard.

1.3 Structure of the Standard

This standard is structured as follows. Firstly, operational requirements are identified, covering all the fundamental monitoring and control functions. These requirements themselves constitute part of the standard and are contained in Chapter 3 of this document. A set of Services is then defined which satisfies these operational requirements. The specification of these Services, including the structure of the associated Service Requests (telecommand packets) and Service Reports (telemetry source packets) is contained in Chapters 4 to 21 inclusive.

It should be noted, however, that many of the operational requirements do not give rise directly to any aspect of an onboard Service, but can only be fulfilled by an appropriate design of the onboard hardware or of the ground system(s).

In summary, therefore, this standard covers:

i) requirements relating to satellite monitoring and control functions and to testability;

ii) derived Service specifications including the definition of telecommand packet and telemetry source packet data content and data structure.

1.4 Applicability

The objective of this Packet Utilisation Standard is that it shall be applicable to the full spectrum of missions which may be operated in the future by ESA.

However, it is not the intention that all the operational requirements detailed herein and the PUS Services specified to meet these requirements should be applicable to every mission. Rather, requirements have been defined which cover all foreseeable operational scenarios and, it is accepted that, for a given mission, only a sub-set of these requirements (and hence PUS Services) may be applicable.

The PUS should therefore be viewed as a "Shopping List" from which the applicable Requirements and corresponding Services should be selected by a given mission.

It is foreseen that the results of this selection process will be documented in a mission-specific ground/space Interface Control Document (ICD).

The following strong recommendations are made concerning this selection process:

i) operational requirements and Services should be de-selected from the PUS (rather than positively selected). This will ensure that all requirements are properly considered and only dropped if a sound technical and/or operational justification can be cited;

ii) if a given function (Service) is required for a given mission, the PUS Standard Service should be used;

iii) it is foreseen that different missions may require varying levels of complexity/functionality from a given Service. A Service may therefore 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, and at least this set must be included in every implementation of a given Service.

2 Context of the PUS

Telemetry and telecommands constitute the data of the closed-loop system with which the ground achieves control of a satellite in order to fulfil mission objectives and agreed mission plans. They are also used (along with stimulation and measurement data, not available in-orbit) during electrical integration and testing of the satellite.

The complexity of future ESA missions may vary considerably. At one extreme, an unmanned geostationary satellite which is operated purely under ground control (because of its continuous ground coverage) may have a relatively simple onboard design. At the other extreme, a deep-space mission which must accomplish adaptive operations and "in-situ" scientific measurements essentially autonomously, because of the extremely large signal propagation delay, will inevitably be complex. Manned missions bring new dimensions in terms of the capabilities for onboard control and the requirements for safety and reliability.

Whilst it is beyond the scope of this standard to perform a detailed analysis of the differing requirements which emerge from widely different missions, it can nevertheless be concluded that the three principal driving factors which place the strongest demands on telemetry and telecommanding are the degree of onboard complexity, the degree of oonboardautonomy (these two factors are, of course, strongly inter-related) and the data transmission capacity.

2.1. Onboard Complexity.

The level of onboard complexity strongly affects the types of telemetry and telecommands required to control the satellite. The simple satellites of the past consisted essentially of hardware and hard-wired logic for which telecommands of the conventional on/off and memory-load type sufficed for control purposes. Monitoring of these satellites was achieved by sampling status information, oonboardregisters and analogue sensor readings.

In the late 1970s/early 1980s, the first onboard software elements began to appear in the data-handling and AOCS subsystems. The ground system then had to provide support for program control and for loading and dumping onboard memory (in the case of RAM-based software). Recent advances in micro-processor technology have heralded a corresponding proliferation in the amount of onboard software, such that today most platform subsystems and payloads have some element of onboard software control. This has implications on the design of the telemetry and telecommand interfaces, since the interaction between the ground and the onboard systems has now become more in the nature of process control than direct control.

2.2. Onboard Autonomy.

The rationale for introducing more autonomy into the onboard design comes from several sources:

i) to reduce the dependency on the ground by performing routine monitoring tasks onboard. The logical extension is that the onboard system should then be capable of reconfiguring autonomously into a safe mode when anomalies or failures are detected (e.g. a survival mode where the payload may be switched off);

ii) to permit the satellite to respond adaptively to conditions determined onboard for example in order to maximise the quality and quantity of mission products;

iii) to allow the implementation of a mission plan under autonomous control where ground control would not be feasible owing to discontinuous ground station coverage or a long signal propagation delay (or both). This might also cover the situation where it is preferable to undertake operations autonomously onboard rather than to utilise a costly ground segment purely to perform the same operations under ground control;

iv) to be able to reconfigure to a safe and ***productive*** mode in the presence of one or more onboard failures.

It is believed that all classes of mission would benefit to some degree from aspects i) and ii) above. However, many missions may only be realisable if they possess the capability for the onboard autonomous execution of a "pre-loaded" mission plan and/or the ability to reconfigure to a productive mode.

To date, onboard autonomous functions have been implemented on an ad-hoc basis. There is, however, a clear desire (and a trend) towards establishing a standardised approach to the implementation of onboard autonomous functions. As regards the autonomous execution of mission plans, the concept foresees the ground uplinking "high-level directives" to a centralised onboard control authority (for example "Configure the experiments to observing mode X whenever the satellite is over the sea"). This authority then derives subordinate directives which are passed to lower-level authorities residing in the satellite subsystems and payloads.

Unfortunately, a standardised approach to the implementation of onboard autonomous functions has not yet been defined in detail, but will inevitably lead to further changes in the nature of ground/satellite interactions (telecommand and telemetry). The present standard is limited, therefore, to rather basic autonomous functions (for example, Onboard command scheduling, Onboard parameter monitoring) but will be extended in this area in the future.

2.3. Data Transmission Capacity

Some satellite missions are very demanding in terms of the telemetry downlink bit-rate and require that some optimisation processes have to be performed onboard to reduce the data transmitted to the ground to match the available downlink capacity. Typical processes might include:

- onboard processing of payload data to extract valid events, data "fusion", the formation of histograms or time profiles etc. prior to the generation of science packets;

- data generation only as the result of prescribed onboard events such as the detection of an onboard anomaly or failure, the action of an onboard process such as the initiation of an autonomous reconfiguration, the acceptance and/or execution of a telecommand etc.;

- data generation initiated on ground request (such as dumping of an onboard memory);

- onboard adaptive control to optimise the generation of mission products.

It should be noted that data types and generation methods are usually inter-related, some generation methods being more appropriate to particular data types (e.g. event-generation being used for report data, ground request being used for dumping memory data).

Some satellite missions will also have constraints on the uplink data throughput capability. This may be due, for instance, to limitations in ground station coverage (LEO missions) or extremely narrow bandwidth (e.g. deep-space missions). The use of a high-level telecommand interface with the ground, minimising the quantity of uplink data, will clearly facilitate the operations of such missions.

These factors, amongst others, characterise the trends in the design of unmanned satellite missions. They are included therefore, together with the more traditional requirements (derived from experience) for the monitoring and control of satellite subsystems and payloads, in elaborating operational requirements for the future for telecommands and telemetry data.

3 Operational Requirements

3.1 Introduction

3.1.1 Purpose

The purpose of this chapter is to elaborate operational requirements for satellite monitoring and control which relate in any way to telemetry and telecommand packets. These requirements may determine, for example, the need for a particular onboard Service or for the content of, generation criteria for, or routing of, associated packets.

3.1.2 Requirements Classification

The operational requirements are classified (grouped) according to the different monitoring and commanding functions which may be needed to conduct both nominal and contingency operations for the full spectrum of future mission types.

As already discussed, the operations concept for future ESA missions may vary quite considerably from one particular mission to the next. At one extreme, all operations may be executed in real-time under direct ground control, whilst at the other extreme, all operations may be executed "autonomously" onboard in accordance with instructions (e.g. time-tagged commands) loaded from ground at some earlier time.

Nevertheless, the following basic generic capabilities will be required in order to conduct mission operations:

Monitoring

- routine health monitoring of the subsystems and the payload;
- anomaly identification and recovery;
- verification of telecommand execution at each distinct stage;
- performance evaluation on the ground for long-term trend analysis and feedback into the mission planning cycle;
- inspection of the content of onboard memories.

Commanding

- control of subsystem and payload hardware;
- control of, and communication with, onboard software e.g. functions of an Application Process, task-level control etc.;
- loading, dumping and checking of onboard memories. In principle, however, the loading and dumping of memories should be necessary only for onboard software maintenance or contingency scenarios;

- control and interaction with other onboard functions, which may be multi-mission in nature (e.g. a Command Schedule, an Onboard Storage function or an OnboardParameter Monitor) or mission-specific;

- control of packet generation and transmission to the ground.

In the event of onboard problems, all the functionalities above will be required for investigating anomalies and performing contingency operations. However, the ground may also need to set up special conditions onboard in order to assist the troubleshooting process, for instance:

- performing high time-resolution sampling of particular onboard parameters in order to investigate short-timescale phenomena or transient effects. This mode is referred to hereafter as "Diagnostic mode" which is synonymous with the terms "Dwell mode" and "Oversampling mode", which have been used in the past;

- operating an onboard unit or process 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.

3.1.3 Conventions

i) A mnemonic has been devised for each distinct group (i.e. subject) and requirements have been assigned a serial number within that group for identification purposes.

ii) In order to keep the formulation of the requirements themselves concise, definition statements (contained within a box) have been inserted or an introductory preamble is included which serves to identify the overall context of the requirements.

iii) Where amplification of an individual requirement or the provision of a specific example is deemed necessary, this is provided in Appendix A to this document; requirements with such examples are identified with an asterisk (*).

iv) Some operational requirements introduce quantities for which values cannot be defined "across the board", but which will have to be defined on a mission-by-mission basis. Examples of such quantities are time intervals, response times, the maximum number of parameters monitored onboard etc. These are termed "Mission Parameters" and are identified within this document in right-angled parentheses, for example <TIME_CORREL_ACCUR>, and where appropriate, ***typical*** values are indicated. These Mission Parameters are also summarised in Appendix B.

3.2 Telecommanding

The design of any satellite will include a number of elementary (indivisible) control actions, implemented either in hardware or software. Examples of such control actions are: switching a relay, starting or stopping a software task etc.

Elementary control actions may be grouped or sequenced in such a way that they constitute a higher-level control action. For example, a particular sequence of "relay on" commands and software configuration commands may switch on an experiment in a given mode.

A **telecommand function** is an operationally self-contained control action. A telecommand function may comprise, or invoke, one or more lower-level control actions.

TC-1* A telecommand packet shall contain one and only one telecommand function.

TC-2 There shall be no requirement for the ground to send telecommands, as the result of anomalies detected from the telemetry, with a response time of less than <ANOM_RESP_TIME>, typically 1 minute.

TC-2.1 Even then, this is only acceptable during short, well-defined time intervals and for a small number of pre-agreed anomaly conditions which can be recognised easily and unambiguously from the telemetry.

TC-2.2 If more complex telemetry processing is required to detect an anomalous situation (e.g. assessment of several parameters or trends of parameters) then a correspondingly longer reaction time shall also be required.

TC-2.3 All such situations shall be agreed with the Mission Control Authority on a case-by-case basis.

A **device telecommand** is a telecommand which is routed to and executed by onboard hardware e.g. a relay switching telecommand, a telecommand to load an onboard register etc.

TC-3 It shall be possible to command all onboard devices individually from the ground, i.e. if a device is normally commanded using a higher-level telecommand function, it shall nevertheless be possible for the ground to issue a device telecommand destined solely for that device.

TC-4 If a device telecommand executes more than one control action, these actions must be operationally strictly related, such that they constitute a single logical telecommand function. For example, a device telecommand shall not put a battery in trickle charge and at the same time switch on a heater, unless these operations are always strictly related.

TC-5 Where more than one device telecommand is required to invoke a given function, it shall be possible to "pack" such device telecommands within a single telecommand packet.

TC-5.1 This packing shall not, however, be mandatory i.e. the option shall always be available to transmit the device telecommands in separate telecommand packets.

The following **categories of telecommand criticality** are identified:

Category 1: Potentially hazardous telecommands. In accordance with **[AD6]** and **[RD6]** these can be further subdivided into:

Category 1a: Potentially **catastrophic** hazardous telecommands which, if executed at the wrong time or in the wrong configuration, could cause loss of the space segment or disabling injury or loss of life in the case of a manned mission.

Category 1b: Potentially **critical** hazardous telecommands which, if executed at the wrong time or in the wrong configuration, could cause major damage to the space segment leading to significant degradation to the mission or non-disabling personnel injury in the case of a manned mission.

Category 1c: Potentially **marginal** hazardous telecommands which, if executed at the wrong time or in the wrong configuration, could cause minor damage to the space segment or minor non-disabling injury in the case of a manned mission.

For example, pyro commands and ABM fire commands would typically fall into one or other hazardous category.

Category 2: Vital telecommands. These are telecommands which are not hazardous, but which, if not executed at the foreseen time, could cause a significant degradation to the mission.

Category 3: Non-critical telecommands. These are telecommands not belonging to either of the above categories.

TC-6* Telecommands of category 1 shall be implemented onboard by means of two consecutive and independent commands.

TC-7 For a manned mission, telecommands of category 1a shall be implemented oonboard by means of three consecutive and independent commands.

TC-8 Redundant telecommands shall be provided for all telecommands of categories 1 and 2, by means of a maximum diversity onboard routing, i.e. via onboard routes which share no common nodes nor paths. This implies the use of different command chains, MAPs and Application Process IDs.

TC-9 It shall be possible to bypass the Packet Assembly Controller (PAC) and to uplink specified category 2 commands at telecommand transfer frame level. This capability would be used in non-nominal situations where the normal route(s), if any, for activating a given vital device driver by means of a telecommand packet is (are) not available (e.g. to reset a PAC or a microprocessor or to bypass an Application Process which is not responding etc.).

TC-10* In order to circumvent potential lock-out problems affecting onboard telecommand routing and delivery, as well as to be able to resolve these problems, it shall be possible to issue device telecommands from the CPDU within the decoder.

TC-10.1 It shall be possible to issue telecommands of category 1 and 2 in this manner.

The transmission of a given telecommand may be defined as **configuration-dependent** i.e. the telecommand shall only be sent if a particular configuration of a subsystem or payload prevails, or alternatively, shall ***not*** be sent under certain defined conditions.

TC-11* For in-orbit operation, the conditions under which a configuration-dependent telecommand may be sent (or may **not** be sent) shall be determinable unambiguously from the telemetry reporting the status of the satellite.

TC-12* On ground (e.g. during testing), protection against the erroneous transmission of a telecommand may have to be ensured by appropriate design of the ground system.

Protection systems are those onboard functions (implemented either in hardware or software) which are provided to monitor sensor or logic readings and, on the basis of their outputs, either direct or processed, to reconfigure the satellite system or subsystem into a "safe" configuration. Subsequent analysis and recovery action will normally be performed by the ground.

TC-13 It shall be possible to enable and to disable any onboard protection system (or safety mode logic) by means of a single telecommand packet.

TC-14 Where applicable, parameters of protection system detection criteria (e.g. thresholds, number of failure repetitions etc.) shall be modifiable by telecommand.

TC-15* Where the protection system mode has several inputs which are logically "OR-ed" to detect failure (e.g. sensor readings, unit status etc.), it shall be possible to enable and disable each independent input by means of a single telecommand packet.

TC-15.1 The protection system shall assume an appropriate value for a disabled input (i.e. one which corresponds to "no failure").

TC-16 Where a protection system selects redundant units for automatic reconfiguration, it shall be possible to specify, by telecommand, which units are to be monitored (for failure) and which combination of units is to be selected for the reconfiguration (from the allowable combinations).

3.3 Telecommand Transmission and Distribution

TCTRANS-1 It shall be possible to "interrupt" the transmission of a long telecommand packet by the transmission of another telecommand packet of higher priority.

TCTRANS-1.1 This shall be possible both when the packets are destined for different Application Process IDs and when they are destined for the same Application Process ID.

TCTRANS-1.2 This capability shall be achieved by the appropriate allocation and use of MAPs i.e. the commands shall be multiplexed on different MAPs with the higher priority command being routed to the priority MAP.

TCTRANS-2* The onboard reception, processing and distribution of telecommands shall ensure that no restrictions arise when the ground transmits telecommands, of any type, at the highest possible rate i.e. making full use of the available uplink bandwidth.

3.4 Telecommand Verification

Introduction

The basic telecommand verification concept is that the onboard design shall permit feedback on the execution of a given telecommand at whichever stages are considered meaningful for that telecommand. A schematic of a possible telecommand execution profile is shown in Figure 3-1 below. The delay between transmission of the telecommand from the ground and acceptance by the "destination" Application Process may correspond to propagation delay (deep-space missions) or the onboard storage of the telecommand prior to release to the Application Process.

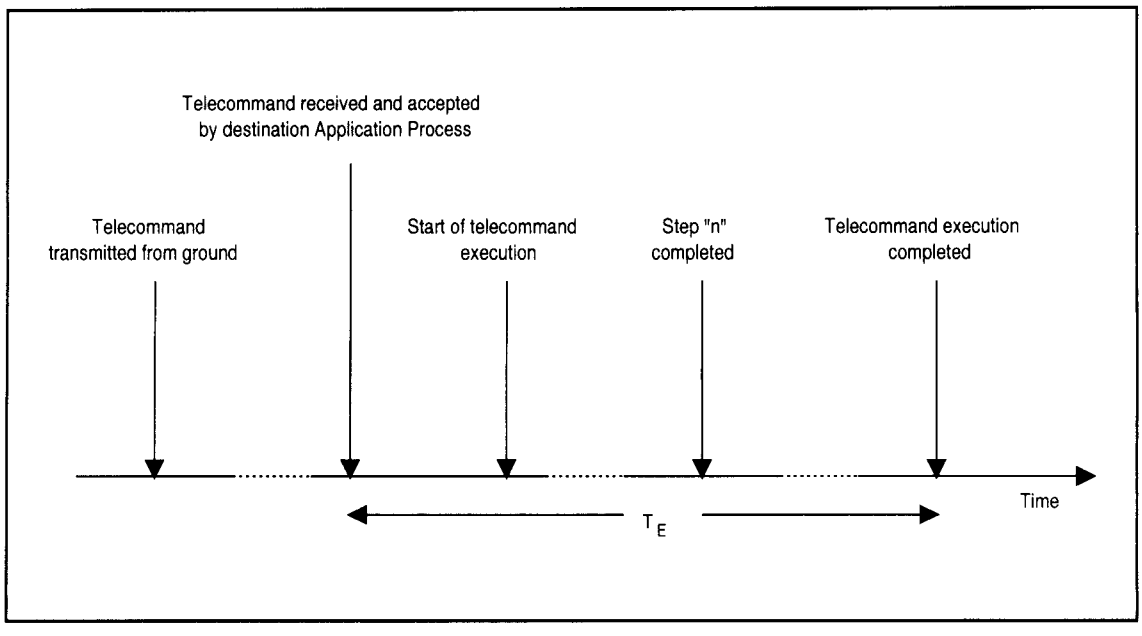


Figure 3-1 Example of a Telecommand Execution Profile

For an immediate telecommand, notification of telecommand acceptance may not be necessary and, for a short execution duration telecommand, only one execution verification report packet (combining start and completion status) may be sufficient. Typical requirements for the different stages of telecommand verification are summarised in Table 3-1 below for telecommands with different characteristics:

	Immediate Command	**Delayed Execution Command**
Short Execution Duration $T_E < 10$ seconds	(A) + E_t	A + E_t
Long Execution Duration $T_E > 10$ seconds	(A) + E_s + E_p + E_t	A + E_s + E_p + E_t

Table 3-1 Telecommand Acceptance and Execution Telemetry

Where:

A = Notification of telecommand acceptance

E = Execution report:

E_s = Execution start
E_p = Execution progress
E_t = Execution complete

Failure of telecommand execution at any of the identified stages will always be reported. However, which reports of successful execution will be generated will be selected for each instance of the telecommand depending on the prevailing operational scenario (e.g. whether or not there is ground coverage).

It should be emphasised that the existence of the capability for different stages of telecommand verification does not imply that it shall be possible to verify every telecommand in this manner. Which stages can be verified will be specified for each telecommand of a mission and the ground system will ensure that only verification of these stages can be requested.

Figure 3-2 shows the different layers of the telecommand system; the layers below the dotted line are the subject of the ESA Packet Telecommand Standard **[AD3]**, whilst the present standard addresses the Application Process layer, i.e. everything above the dotted line.

The packets notifying telecommand acceptance[1] and telecommand execution verification[1] 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.

[1]Where requested

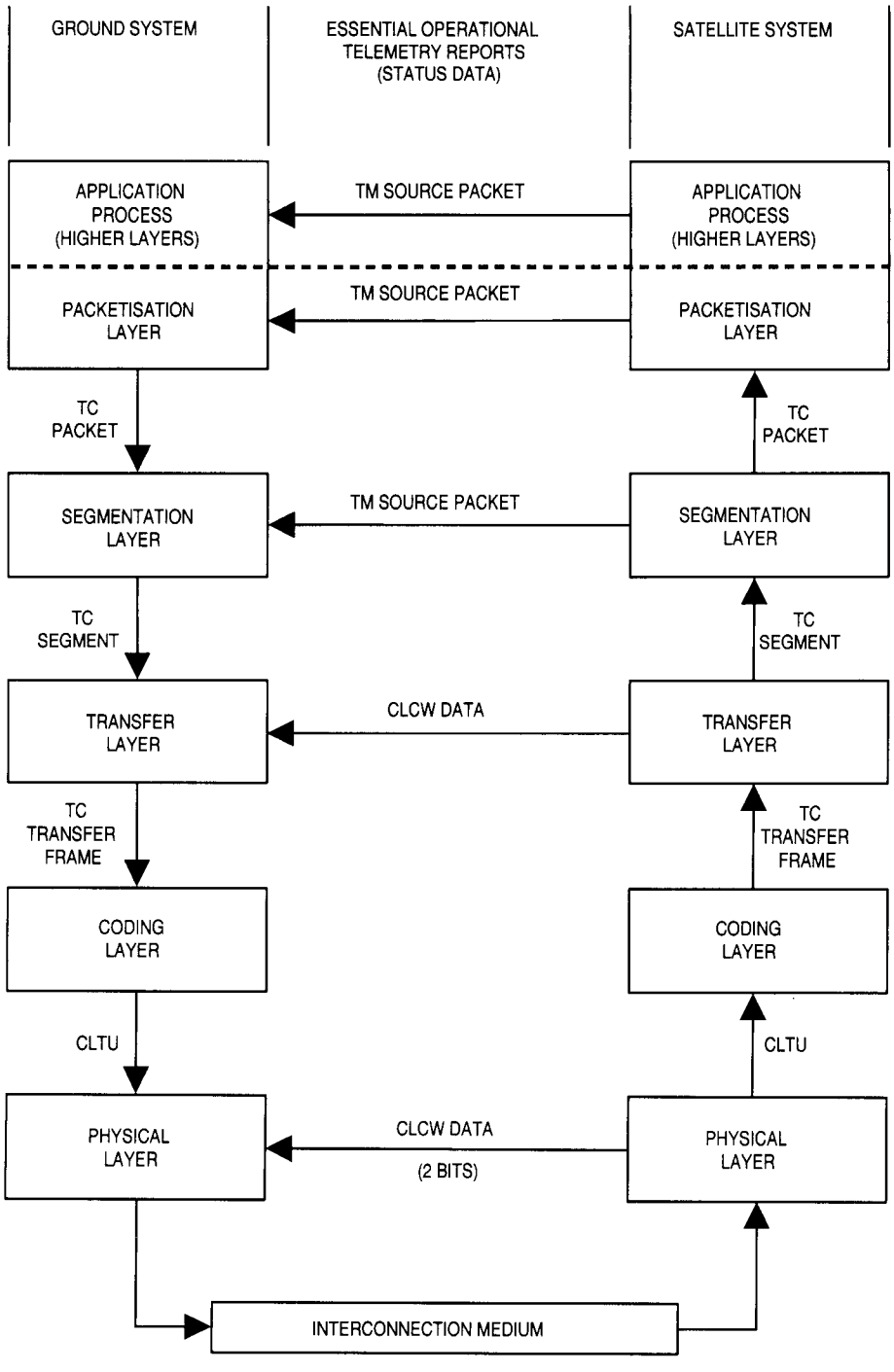


Figure 3-2 Telecommand System Layers

TCVERIF-1 Verification telemetry shall be provided for all telecommands that have been properly executed, whether these are sent directly from the ground, are stored onboard for release at a later time, or are generated autonomously onboard.

TCVERIF-1.1 This verification telemetry shall be provided with a delay of less than <TC_VERIF_DELAY> with respect to the time of completion of the telecommand execution (typically, less than 10 seconds).

TCVERIF-1.2 A telecommand shall be verified by telemetry measured at the same level as that at which the telecommand is executed, e.g. a device telecommand shall be verified by a hardware measurement which is directly telemetred, without intermediate processing.

TCVERIF-1.3 If a telecommand results directly in one or more changes in the satellite configuration, these changes shall be reflected in the telemetry reporting the satellite status.

TCVERIF-2 For each telecommand, verification packets shall be provided that allow complete and unambiguous verification of well-defined stages of telecommand execution.

TCVERIF-2.1 Failure of telecommand execution at any of the identified stages shall always be reported.

TCVERIF-2.2 Which reports of successful execution will be generated shall be indicated within the telecommand packet itself and the selection possibilities shall range from "no verification packets" to "verification at each distinct step of acceptance and execution".

TCVERIF-2.3 The event of telecommand ***acceptance*** shall be notified by a telecommand verification packet indicating the proper acquisition and the validity of the telecommand for execution (e.g. valid type/sub-type).

TCVERIF-2.4 The telecommand acceptance verification packet shall include the following information:

TCVERIF-2.4.1 telecommand identification and the originating source;

TCVERIF-2.4.2 telecommand acceptance status indicating whether the telecommand was successfully received and was considered to be valid;

TCVERIF-2.4.3 ancillary error information sufficient to determine unambiguously the cause of telecommand rejection.

TCVERIF-2.5 Each telecommand packet shall include a Packet Error Control (PEC) field, in the form of either a checksum or a Cyclic Redundancy Code (CRC).

TCVERIF-2.5.1 The PEC shall be used by the destination Application Process to confirm the correct content of the packet as part of the telecommand acceptance stage.

TCVERIF-2.6 Following valid acceptance, the verification of ***execution*** of a telecommand shall be provided through the generation of one or more telecommand verification packets, reporting "on-line" on the different stages of the execution of the telecommand.

TCVERIF-2.7 These telecommand execution verification packets shall include the following information:

TCVERIF-2.7.1 telecommand identification and the originating source;

TCVERIF-2.7.2 telecommand execution status (i.e. start, progress or completion of execution);

TCVERIF-2.7.3 in the case of an abnormality, all the ancillary information required to determine the cause.

TCVERIF-2.8 The telecommand execution time shall be given by the packet time of the corresponding execution verification packet.

TCVERIF-3 If a telecommand packet is received onboard which does not conform with the ESA Packet Telecommand Standard **[AD3]**, it shall be rejected.

TCVERIF-3.1 An appropriate telemetry report shall also be generated.

3.5 Satellite Status Reporting

SSREP-1 "Standard" telemetry source packets shall be provided which allow access to all the data required for the execution of all nominal operations and foreseen contingency operations for the satellite platform subsystems and the payload.

SSREP-1.1 The data shall be provided in such a manner that complete and unambiguous evaluation of satellite status and performance is possible.

SSREP-1.2 These data shall include sensor readings, register readouts, equipment status, software status, reports of onboard events and autonomous actions etc.

SSREP-2 For a given mission, there will be a sub-set of data that is sufficient to characterise the current state of the satellite and indicate whether there is an anomalous condition that requires intervention from the ground.

SSREP-2.1 An appropriate reserved downlink bandwidth shall be provided for these data (even though it may not always be used).

SSREP-3 Event-based reporting shall be achieved by means of dedicated report packets (progress/anomaly reports) which may be derived onboard from the processing of low-level measurements, e.g. a failure detection and recovery function.

SSREP-3.1 All low-level inputs to such onboard processing functions shall also be ***accessible*** to the ground (though not necessarily all at the same time) via satellite status report packets.

SSREP-3.2 All onboard "events" of operational significance shall be reported in a complete and unambiguous manner.

SSREP-3.2.1 This includes notification of all onboard autonomous actions.

SSREP-3.2.2 The level of information shall allow the ground to predict the onboard state accurately (i.e. in order to be able to compare the state observed in the telemetry with that "expected" on ground).

SSREP-3.3 Anomaly reports shall contain a unique identification of the anomaly, its time of occurrence and a record of the input data to the anomaly detection function.

SSREP-3.3.1 These input data shall be recorded in such a way that they can be reported for an appropriate interval of time <ANOM_REPORT_INTERV> centred on the time of occurrence of the anomaly.

SSREP-3.4 The design of reporting mechanisms and of the report packets themselves shall be such that the utilisation of the downlink bandwidth shall not be excessive.

SSREP-3.4.1 To this effect, related events shall be combined into the same report where appropriate (e.g. the detection of a failure and the action taken to correct it).

SSREP-3.4.2 Failure detection algorithms shall not repeat the same exception packet if the same failure is detected at each successive failure detection cycle, although a separate report should be generated if the exception condition disappears.

SSREP-3.5 Information to identify the nature (in particular, the severity) of a report packet shall be included in the packet data field header.

SSREP-4 The low-level satellite status telemetry shall fulfil the following requirements:

SSREP-4.1* the information shall be provided from direct measurements rather than from secondary effects;

SSREP-4.2 the complete status of the satellite shall be derivable from the telemetry without the need for reference to the telecommand history;

SSREP-4.2.1 Software status shall be deemed to include all commandable software parameters such as monitoring and control thresholds, software tables, software flags etc.

SSREP-4.2.2 This implies, in particular, that the effect of telecommands shall always be observable in the low-level telemetry.

SSREP-4.3 the values of telemetred parameters shall be self-contained. This precludes, for instance, the telemetring of delta-changes (the actual value must be given) or status changes (the actual value or status must be given);

SSREP-4.4 in some cases, parameters may change value for very short time periods. If it is necessary, for operational reasons, to detect such occurrences (e.g. as an indication of an onboard failure) and no adequate sampling of the parameter can be provided, the event shall be memorised and the memorised value shall be telemetred;

SSREP-4.5 the design of "analogue" telemetry shall be such that the "full-scale" range covers the full performance range of the parameter being measured, whilst ensuring that the resolution is also adequate for monitoring and evaluation purposes.

SSREP-4.5.1 This shall take into account the performance range of the parameter during all mission phases as well as excursions resulting from predictable life-cycle degradation.

SSREP-4.5.2 It shall be possible to modify the resolution in orbit, when necessary (for instance, by range switching).

SSREP-5 The design of the overall Mission Operations System (i.e. constituting both the ground and space segments) shall ensure that control loops which have short response times are implemented onboard.

SSREP-6 There may be ground control loops with identified response times <GRND_RESP_TIME> (there may be several such parameters for a given mission). Satellite status telemetry shall, therefore, comply with the following requirements for generation and transmission:

SSREP-6.1 under all circumstances, the elapsed time for an Application Process to build and release telemetry source packets shall be such that the overall delay between the generation of a packet and its reception at the Mission Control Centre is compatible with any response times which have been identified for ground control loops;

SSREP-6.2* parameters within telemetry source packets which are generated periodically shall be sampled at a frequency which ensures that no information of operational significance is lost;

SSREP-6.2.1 This shall be true for all nominal and foreseen contingency operations.

SSREP-6.3* the frequency of packet generation shall be compatible with the response times which have been identified for any control loops implemented on ground.

In some instances, data and the ancillary information needed for the ground processing of such data may be transmitted in different packets, which will be released by the Application Process at different times.

SSREP-7 The time differences between such packets shall be such that the effective operational information (i.e. after ground processing) is available within delays compatible with the requirements for ground-control loop-response times.

SSREP-8 In exceptional cases, where the chronological order of transmission of such packets to the ground cannot be guaranteed, the time difference between consecutive (time inverted) packets shall not exceed <TIME_INVERS_MAX>, typically 1 second. This requirement applies only to real-time telemetry (not to telemetry recorded onboard).

It may only be meaningful to interpret a given telemetry parameter if certain well-defined conditions prevail. For example, the angular output of a gyro may only have a valid engineering meaning if the power to the gyro is "on"; at other times the output may be random (or at best should not be relied upon). Such a parameter is deemed **Conditionally Valid**, with its **Validity** determined from the power status.

SSREP-9 Where the validity of a parameter depends on the values of other onboard parameters, combined in such a way as to yield a "True or False" result, then these parameters shall all be telemetred in the same packet.

SSREP-10 A telemetry parameter shall always have the same structure and interpretation even if it appears in several telemetry source packets.

SSREP-11 If the same telemetry parameter appears more than once in the same housekeeping telemetry source packet (i.e. it is supercommutated), then:

SSREP-11.1 the parameter shall be sampled regularly in time, at an interval which is the same for all occurrences of the packet (this also implies that the parameter sampling interval must be a sub-multiple of the packet generation interval);

SSREP-11.2 there shall be no missing parameter samples from one packet to the next;

SSREP-11.3 the time offset of the first sample of this parameter within a packet shall be known (see also timing requirements below).

SSREP-12 The design of housekeeping telemetry source packets shall not permit the use of subcommutation.

SSREP-13 It shall be possible for the ground to define (and re-define) the packet contents for housekeeping telemetry source packets.

SSREP-14 It shall be possible to request that housekeeping telemetry source packets are only sent to ground if there has been change in value of specified contained parameters by more than a specified threshold.

SSREP-14.1 This "filtered" mode and periodic generation mode shall be mutually exclusive.

SSREP-14.2 In filtered mode, parameters shall be sampled in the same manner as for periodic mode and the value of a parameter shall be compared with the previous sample to determine whether a change has occurred.

SSREP-14.3 In the case of supercommutated parameters, changes of value shall be detected between any two consecutive parameter samples.

SSREP-14.4 If no packet is generated within a specified "timeout" interval, then a one-shot packet shall be generated.

SSREP-15 It shall be possible for a one-shot housekeeping telemetry source packet to be generated as the result of a mission-specific event detected by any onboard Application Process (e.g. an onboard anomaly).

For the purposes of anomaly investigation (troubleshooting), it may be desirable to set up a telemetry sampling scenario which is quite different from that used in normal operations. Typically, a selected set of parameters pertaining to the problem in hand will be sampled onboard at a higher rate than normal and then telemetred to the ground. This scenario is termed **Diagnostic Mode**. When operating in Diagnostic Mode, the ground will need to carefully select the extent to which these diagnostic packets will occupy the downlink bandwidth taking due account of the other prevailing operational requirements.

SSREP-16 It shall be possible for the ground to define (and re-define) the packet contents for diagnostic mode telemetry source packets.

SSREP-17 Diagnostic packets shall have all the same characteristics as housekeeping packets (and shall therefore comply with all the relevant foregoing requirements); they shall however be separately identifiable for routing (and possible storage) and processing purposes.

SSREP-18 All onboard telemetry parameters shall be accessible via diagnostic packets.

SSREP-19* In order to ensure that high-rate sampling can be achieved, a minimum sampling interval of <DIAG_MIN_INTERV>, typically 10 to 30 msec, shall be possible for all onboard parameters. In practice, the sampling interval selected for different sub-sets of parameters might vary quite considerably. The minimum sampling interval may 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 (say, 1 sec or more) may be used.

Statistical Data Reporting

SSREP-20 It shall be possible to report statistics relating to a specified set of parameters over an interval of time.

SSREP-20.1 It shall be possible to report maximum, minimum and mean values.

SSREP-20.2 It shall be possible to report the standard deviation.

SSREP-21 It shall be possible to add and delete parameters from the set being evaluated.

SSREP-22 It shall be possible to clear the set of parameters being evaluated.

SSREP-23 It shall be possible to request a report of the current set of parameters being evaluated.

SSREP-24 It shall be possible to reset the start time of the interval over which the parameter statistics are evaluated.

SSREP-25 Alternatively, it shall be possible to specify that the parameter statistics are evaluated and reported periodically.

3.6 Telemetry Timing

TIMING-1 Timing information shall be provided in the telemetry which allows the correlation of onboard time with ground time with an accuracy of <TIME_CORREL_ACCUR>.

TIMING-2 All timing information within the telemetry shall be synchronised with a single onboard Master Clock.

TIMING-3 For different operational purposes (e.g. detailed performance evaluation, data or event correlation at the time of onboard anomalies etc.) it shall be possible to establish the original onboard sampling time of satellite status telemetry parameters appearing in telemetry source packets.

TIMING-3.1 Particularly for the case of anomaly troubleshooting, it shall be possible to establish this information retroactively in time.

TIMING-3.2* It shall be possible to determine the absolute (onboard) sampling time of parameters to an accuracy of <PARAM_ABS_SAMPL_TIME>.

TIMING-3.3* It shall be possible to determine the relative sampling time of any two parameters to an accuracy of <PARAM_REL_SAMPL_TIME>. This shall be possible even if the parameters are telemetred in different packets.

TIMING-3.4 Normally, the requirements for timing shall be achieved from a knowledge of the onboard sampling sequence of these parameters.

TIMING-3.4.1 This may be either from implicit knowledge or from explicit (one-shot) measurements made onboard and telemetred to the ground.

TIMING-3.4.2 The time reference shall be any well-defined time-stamp contained within the same telemetry source packet.

TIMING-3.5 Only in exceptional cases, where the corresponding timing requirements are stringent, shall resort be taken to the explicit datation of parameters to achieve the requirements for timing (e.g. by sampling a time register at the same time as sampling the parameter).

3.7 Memory Management

3.7.1 Memory Loading

MLOAD-1 It shall be possible for the ground to load any changeable memory area.

MLOAD-2 It shall be possible to load, with a single telecommand packet, either a contiguous memory area (e.g. by indicating the start address for loading and the length of the load) or to perform scatterloads (e.g. by specifying pairs of memory address and data to be loaded).

MLOAD-3 The start address of a memory load shall be specified either as an absolute address within the given memory or as a "base + offset" couplet.

MLOAD-4 As part of the onboard acceptance of a memory load, the destination Application Process shall be able to detect data corruptions.

MLOAD-5 The onboard 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 load data).

3.7.2 Memory Dumping

MDUMP-1 It shall be possible for the ground to dump any memory area, on request.

MDUMP-2 The memory dump request shall specify the name of the memory to be dumped and indicate either a contiguous memory area (e.g. by indicating the start address and the length of the dump) or a scatter-dump (e.g. by specifying a series of pairs of start address and length).

MDUMP-3 The start address for a memory dump shall be specified either as an absolute address within the given memory or as a "base + offset" couplet.

MDUMP-3.1 For a given memory, the same convention shall be used for dumping as for loading.

MDUMP-4 Only a single telecommand packet shall be required for a memory dump request, even if several telemetry source packets are required to convey the dumped data to the ground.

3.7.3 Memory Checking

MCHECK-1 It shall be possible for the ground to request a check of a specified area of an onboard memory. The following options shall be provided:

MCHECK-1.1 check a range of addresses;

MCHECK-1.2 check several ranges.

MCHECK-2 In response to a request to check memory, the onboard action shall be to perform a checksumming over the requested addresses and report the result to the ground.

3.8 Software Management

An Application Process may perform one or more software **functions** which can be invoked from the ground. This is the "normal" way in which the activities of (i.e. the Services provided by) an Application Process will be invoked. This document defines a number of standard Services. For a given mission, some of these Services may be implemented within all Application Processes; some within only one Application Process; others may not be implemented at all. Mission-specific Services may also be implemented, for example to control the operation of a payload or an AOCS.

Functions are ultimately achieved by the operation of one or more software **tasks** (usually referred to as processes) running under the control of an Application Process. Under some circumstances, it may be required for the ground to directly control the software tasks.

SWMAN-1 It shall be possible to exercise control over an Application Process in the following manner:

SWMAN-1.1 suspend (deactivate) an Application Process;

SWMAN-1.2* reset an Application Process.

SWMAN-2* Any communication between the ground and an onboard software function or software task shall be effected by means of telecommand and telemetry source packets specifically designed for the purpose.

SWMAN-3 It shall be possible to control and interact with functions of an Application Process in the following manner:

SWMAN-3.1 activate a function;

SWMAN-3.2 deactivate a function;

SWMAN-3.3 perform an activity of a function;

SWMAN-4* It shall be possible to control tasks managed by an onboard Application Process in the following manner:

SWMAN-4.1 start a task;

SWMAN-4.2 stop a task;

SWMAN-4.3 suspend a task;

SWMAN-4.4 resume a task;

SWMAN-4.5 abort a task.

SWMAN-5 In normal operations (as opposed to non-nominal situations), it shall be possible to communicate with a task (i.e. pass it parameters) without the need for the ground to first stop, or suspend, the task.

3.9 Onboard Scheduling

An Onboard Schedule is a facility which allows the control and execution of commands which have been loaded in advance from the ground. In its simplest form, the Onboard Schedule stores time-tagged commands loaded from ground and releases them to the destination Application Process when their onboard time is reached, but with no feedback being generated by the destination Application Process. This simple form of onboard scheduling will be retained. However, within this standard, additional functionalities are also foreseen:

i) The Schedule shall be able to "interlock" commands. That is to say, the release of commands can be pre-conditioned on the successful (or unsuccessful) execution of commands which were released at an earlier time from the Schedule. This capability allows simple interlocking of commands from the standpoint of safety. However, it also allows ***alternative sequences of commands*** to be sent from the Schedule depending on the success or otherwise of commands released at an earlier time (IF, THEN...ELSE capability).

ii) Distinct "sub-schedules" are supported. Basic operations on the Schedule, such as start, stop, insert and delete commands, can be performed at the level of a sub-schedule. The interlocking function described above is also implemented at sub-schedule level, i.e. commands which are interlocked must reside in the same sub-schedule.

In principle, several onboard Application Processes may support such an Onboard Scheduling function. The derived requirements for the Onboard Scheduling function are as follows:

Interlocking

OBSCH-1 Up to <NUM_SUB_SCHEDS> distinct sub-schedules shall be supported, typically five to ten. The sub-schedules are logical subsets of the commands within the schedule, which are independently controllable.

OBSCH-2* It shall be possible for a command to set an interlock on (i.e. condition the release of) subsequent commands within the same sub-schedule.

OBSCH-2.1 This interlock may be dependent on the command ***either*** being executed successfully ***or*** failing execution.

OBSCH-2.2 For such commands, the destination Application Process shall provide a feedback to the Onboard Schedule, indicating success or failure of execution.

OBSCH-3 If the feedback from the destination Application Process does not arrive before the time of release of an interlocked command or before a specified time-out has expired, then the command setting the interlock shall be deemed to have failed execution.

OBSCH-4 A command shall be deemed to have failed execution for any subsequent commands which are interlocked to it if it is not released from the Onboard Schedule for one of the following reasons:

OBSCH-4.1 it was interlocked to an earlier command which prevented its release;

OBSCH-4.2 the Onboard Schedule (or the sub-schedule) was started after the due time of release of the command.

OBSCH-5 Commands may only be interlocked by a single command released earlier from the same sub-schedule.

Operations on the Schedule

OBSCH-6 It shall be possible to start the Onboard Schedule with the following options:

OBSCH-6.1 the complete Onboard Schedule (i.e. commands for all Application Process IDs in all sub-schedules);

OBSCH-6.2 specified sub-schedules, for all Application Process IDs;

OBSCH-6.3 specified sub-schedules, but for specified Application Process IDs only.

OBSCH-7 It shall be possible to stop the Onboard Schedule either immediately or at a specified time in the future. The options for stop shall be:

OBSCH-7.1 the complete Onboard Schedule (i.e. commands for all Application Process IDs in all sub-schedules);

OBSCH-7.2 specified sub-schedules only (but for all Application Process IDs);

OBSCH-7.3 specified sub-schedules, but for specified Application Process IDs only.

OBSCH-8 It shall be possible to insert and append commands to the Onboard Schedule, without the necessity of first stopping it.

OBSCH-8.1 A command added to the Onboard Schedule shall inherit the status (started/stopped) of the Application Process ID and sub-schedule to which it belongs.

OBSCH-9 It shall be possible to delete commands from the Onboard Schedule, without the necessity of first stopping it. The delete options shall include:

OBSCH-9.1 all commands (i.e. to reset the schedule contents);

OBSCH-9.2 commands from a specified time onwards[1];

OBSCH-9.3 commands between specified times[1];

OBSCH-9.4 commands in a specified sub-schedule, from a specified time onwards[1];

OBSCH-9.5 commands in a specified sub-schedule, between specified times[1];

OBSCH-9.6 individual commands.

OBSCH-9.7 [1]These options shall be possible for either "all Application Process IDs" or "specified Application Process IDs only".

OBSCH-10 It shall be possible to time-shift (advance or retard) commands which have already been loaded in the Onboard Schedule. This function is required to support re-planning of operations and to avoid having to delete and re-load the affected commands. The options for time-shifting shall include:

OBSCH-10.1 commands from a specified time onwards[2];

OBSCH-10.2 commands between specified times[2];

OBSCH-10.3 commands in a specified sub-schedule, from a specified time onwards[2];

OBSCH-10.4 commands in a specified sub-schedule, between specified times[2];

OBSCH-10.5 individual commands.

OBSCH-10.6 [2]These options shall be possible for either "all Application Process IDs" or "specified Application Process IDs only".

OBSCH-11 It shall be possible to request a report of the contents of the Onboard Schedule. The options for reporting shall include:

OBSCH-11.1 full report or summary only;

OBSCH-11.1.1 The summary option shall indicate the telecommand packet header but not its content.

OBSCH-11.2 the complete Onboard Schedule;

OBSCH-11.3 commands from a specified time onwards[3];

OBSCH-11.4 commands between specified times[3];

OBSCH-11.5 commands in a specified sub-schedule, from a specified time onwards[3];

OBSCH-11.6 commands in a specified sub-schedule, between specified times[3];

OBSCH-11.7 [3]These options shall be possible for either "all Application Process IDs" or "specified Application Process IDs only".

OBSCH-12 The Onboard Schedule shall be able to accept all telecommands ***including*** those which operate on the Onboard Schedule itself.

OBSCH-12.1 The only exception shall be the telecommand to start **this** OonboardSchedule.

Relative Time

OBSCH-13 As an alternative to an absolute time-tag, it shall be possible to attach a relative time-tag to commands within the Onboard Schedule.

OBSCH-14 The reference for relative time may be one of the following events:

OBSCH-14.1 the starting of the Onboard Schedule;

OBSCH-14.2 the starting of the sub-schedule in which the command is loaded;

OBSCH-14.3 the execution time (as opposed to the release time) of the command to which this command is interlocked.

3.10 Onboard Monitoring

An onboard parameter monitoring function is envisaged which is functionally similar to the traditional monitoring which is done on ground. The primary purpose of this function is to reduce the necessity for continuously downlinking all the low-level housekeeping data and/or for the ground to be continuously available to monitor this data. In the future it is envisaged that the scope of the function may be widened to include the triggering of oonboardactions in response to detected out-of-limit conditions.

The onboard monitoring function may be supported by more than one Application Process and the maximum number of parameters which can be simultaneously monitored per Application Process is <MONLIST_MAX_PARAMS>, typically 200 to 300. Parameters can be either "limit-checked" (i.e. checked to lie within a specified range) or "fixed status checked" (i.e. checked against a specified value) and/or "delta-checked (i.e. its change in value is checked against a pair of thresholds). A parameter may be conditionally valid, in which case its validity will 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 fixed status and/or delta checks can be specified per parameter, to a maximum of <MONLIST_MAX_CHECKS>, each with an associated "check selection parameter". The parameter is ***independently*** checked against ***each*** check whose check selection parameter is currently TRUE. The derived requirements for such a monitoring function are as follows:

OBMON-1 It shall be possible to enable and disable the onboard parameter monitoring in the following manner:

OBMON-1.1 at the level of the Application Process;

OBMON-1.2 for selected parameters.

OBMON-2 If the onboard monitoring is disabled at the level of the Application Process, the monitoring status (enabled/disabled) at parameter level shall remain unchanged. That is to say, if the monitoring is subsequently enabled for the Application Process, only those parameters which were previously enabled are now monitored.

OBMON-3 It shall be possible to add parameters to the monitoring list by specifying:

OBMON-3.1 the parameters to be monitored;

OBMON-3.2 for each parameter which is conditionally valid, an associated validity parameter;

OBMON-3.2.1 Such a parameter is only monitored if its associated validity parameter is "TRUE".

OBMON-3.2.2 A parameter with no associated validity parameter is unconditionally valid and is always monitored.

OBMON-3.3 either one or more sets of upper and lower limits to a limit of <MONLIST_MAX_CHECKS>, typically five;

OBMON-3.4 or one or more fixed status checks, to a limit of <MONLIST_MAX_CHECKS>, typically five;

OBMON-3.5 and/or one or more delta checks, to a limit of <MONLIST_MAX_CHECKS>, typically five;

OBMON-3.6 for each check, a Boolean parameter which determines whether the check is applied (check selection parameter);

OBMON-3.7 the sampling interval for the onboard monitoring (i.e. the monitoring interval);

OBMON-3.7.1 The sampling interval shall be expressed as a multiple of the minimum onboard sampling interval defined for diagnostic mode, <DIAG_MIN_INTERV>.

OBMON-3.7.2 The sampling interval may be specified on a parameter basis.

OBMON-3.7.3 The sampling interval may be specified globally for all parameters in the set to be added.

OBMON-3.8 a repetition filter, specifying the number of times that a parameter shall be registered as failing a check before being reported as such (see below).

(The different parameter types are defined within Chapter 23 of this document).

OBMON-4 It shall be possible to modify any sub-set of the monitoring information for a parameter (i.e. without having to first delete the parameter and then add it again to the monitoring list).

OBMON-5 It shall be possible to clear (i.e. to reset) the monitoring list.

OBMON-6 It shall be possible to delete any sub-set of parameters from the monitoring list.

OBMON-7 Telemetry report packets shall be generated in the following instances:

OBMON-7.1 on the first occasion that a parameter is detected out-of-limits, i.e. has a value greater than the high limit or less than the low limit;

OBMON-7.2 on the first occasion that a parameter is detected back in limits, having previously been out-of-limits, i.e. it now has a value less than or equal to the high limit but greater than or equal to the low limit;

OBMON-7.2.1 If a parameter passes from being out-of-limits "high" to out-of-limits "low" between successive checks, it shall be considered as being newly out-of-limits, since a new limit has been transgressed.

OBMON-7.3 on the first occasion that a parameter fails its fixed status check;

OBMON-7.4 on the first occasion that a parameter passes its fixed status check, having previously failed it;

OBMON-7.5 on the first occasion that a parameter fails its delta check;

OBMON-7.6 on the first occasion that a parameter passes its delta check, having previously failed it;

OBMON-7.7 whenever the value of a validity parameter transits between "TRUE" and "FALSE";

OBMON-7.8 whenever the value of a check selection parameter transits between "TRUE" and "FALSE".

OBMON-8 The telemetry report packet shall contain details of only those parameters whose check status has changed since the last report and shall include the following information:

OBMON-8.1 for parameters which are limit checked, the parameter value and the value of the limit being crossed;

OBMON-8.2 for parameters which are fixed status checked, the parameter value and the fixed status being violated;

OBMON-8.3 for parameters which are delta checked, the parameter value and the value of the delta being violated.

OBMON-9 It shall be possible to request a report of the contents of the monitoring list.

OBMON-9.1 The report shall give the list of monitored parameters together with their associated validity parameters, check definitions and check selection parameters.

OBMON-10 It shall be possible to request a report of the full set of parameters currently "check failed" for a given Application Process ID.

OBMON-10.1 The report shall indicate, for each parameter, the nature and time of check failure, the identification of the check being violated and the value of the parameter at that time.

3.11 Large Data Transfer

LDATA-1 Taking into account such factors as uplink bandwidth, ground contact periods, time to recover from telecommand failure etc., it shall be possible for a given mission to define a maximum telecommand packet length which is less than the maximum allowed by the ESA Packet Telecommand Standard **[AD3]**.

LDATA-1.1 This maximum telecommand packet length shall be a Mission Parameter <TCPKT_MAX_LENGTH>.

LDATA-2 Taking into account such factors as downlink bandwidth, ground contact periods etc., it shall be possible for a given mission to define a maximum telemetry source packet length which is less than the maximum allowed by the ESA Packet Telemetry Standard **[AD1]**.

LDATA-2.1 This maximum telemetry source packet length shall be a Mission Parameter <TMPKT_MAX_LENGTH>.

LDATA-3 Where it is necessary to transfer a set of data which would exceed the maximum packet size specified for the mission, a capability shall be provided to transfer this data in more than one packet.

LDATA-3.1 This "large data transfer" capability shall be provided both for telecommand and telemetry source packets.

LDATA-3.2 It shall be possible to send large data transfer packets with or without acknowledgement of receipt (either intermediate or final).

LDATA-3.3 In the event of onboard rejection of a telecommand packet by its onboard destination, it shall nevertheless be possible to uplink succeeding packets.

LDATA-3.3.1 At the end of the operation it shall then be necessary only to re-uplink the rejected packet(s) but not any succeeding packet that has been successfully accepted.

3.12 Telemetry Generation and Transmission Control

One or more Users may require the reception of particular telemetry source packets. The ground will be the primary User for many packets, but another onboard Service may also require packet delivery, for example an Onboard Storage Service. This Service may either be external, or internal, to the generating Application Process.

The source Application Process will therefore generate the required packets and transmit them to the relevant User(s). This process may utilise various resources, such as oonboardprocessing, onboard communications networks, the space-ground link etc. The generation and transmission of telemetry source packets must therefore be so designed that these resources are only used when required, i.e. adequate controllability of the generation and transmission functions must be provided. (This may imply a point-to-point communication between generating and receiving entities.)

TMGEN-1 It shall be possible to selectively enable and disable the transmission of telemetry source packets to the ground. The selection shall be at the level of Application Process ID and shall include the following possibilities:

TMGEN-1.1 all packets;

TMGEN-1.2 specified type(s)/sub-type(s);

TMGEN-1.3 specified housekeeping packets;

TMGEN-1.4 specified diagnostic packets;

TMGEN-1.5 specified report packets.

TMGEN-2 It shall be possible to selectively enable and disable the request for delivery of telemetry source packets by an onboard Service which requires those packets, thereby interrupting the transmission by the generating Application Process of the selected packets. The selection shall be at the level of Application Process ID and shall include the following possibilities:

TMGEN-2.1 all packets;

TMGEN-2.2 specified type(s)/sub-type(s);

TMGEN-2.3 specified housekeeping packets;

TMGEN-2.4 specified diagnostic packets;

TMGEN-2.5 specified report packets.

TMGEN-3 The process of telemetry source packet generation shall be automatically terminated by the generating Application Process in the case that no User/Service requires packet delivery.

TMGEN-4 It shall be possible to enable and disable telemetry source packet generation at the level of the originating Service.

TMGEN-5 Each Application Process shall provide status information indicating those packets whose generation is currently enabled and (where appropriate) the generation frequency and those packets which are selected for transmission to the ground and/or other onboard Services.

3.13 Onboard Storage and Retrieval

Missions with Intermittent Ground Coverage

STORE-1* For missions with intermittent ground coverage, the onboard storage capability shall be sufficient to store all packets generated onboard, which may be required for satellite monitoring and control purposes, for a duration at least equal to the longest non-coverage period plus a mission-dependent margin <PKT_STORAGE_TIME>, typically one orbit for low-Earth-orbiting satellites.

STORE-2 Onboard storage shall be such that the ground can retrieve the stored data within specified delays <PKT_RETR_DELAY>. There may be several such parameters for a given mission. For example, packets of high operational significance, such as anomaly report packets, will normally be required on the ground with shorter delays than routine status reporting packets.

Lost Packet Recovery

STORE-3 In the case of the loss of a telemetry source packet during transmission, the ground must be capable of identifying the lost packet unambiguously.

STORE-3.1 This shall be achieved by the source Application Process ensuring that packets issued in real time have a contiguous Source Sequence Count.

STORE-3.2 Ideally, the Source Sequence Count should never re-initialise; however, under no circumstances shall it "short-cycle" (i.e. have a discontinuity other than to a value of zero).

STORE-4 For missions with continuous ground coverage, loss of one-shot packets (i.e. event-driven or request-driven packets) shall be remedied by the short-term onboard storage of the last <PKTS_NUM_STORED> packets, typically the last one hundred. Note that this requirement includes telecommand verification packets (they are request driven).

General

STORE-5 Housekeeping information shall be provided on the state of the onboard storage and retrieval function.

STORE-5.1 A count of the number of packets stored (since the last reset) shall be maintained.

STORE-5.2 It shall be possible for the ground to reset this count by telecommand.

STORE-6 For each independent onboard store, it shall be possible for the ground to enable and disable the storage function.

STORE-7 For each independent onboard store, it shall be possible to specify which packets shall be stored.

STORE-8 The possibilities for storage shall be either:

STORE-8.1 cyclically, where the oldest data are overwritten when the store is full (this is the only option for the lost packet recovery store);

STORE-8.2 linearly, where storage terminates when the onboard store is full.

STORE-9 It shall be possible for the ground (and only the ground) to clear the contents of an onboard store.

STORE-9.1 It shall be possible to clear the store completely.

STORE-9.2 It shall be possible to clear the store on the basis of time (e.g. all packets older than a specified time).

STORE-10 The storage of packets shall not be interrupted if the ground requests a retrieval from, or reset of, the onboard storage.

STORE-11 It shall be possible for the ground to request the retrieval of packets from oonboardstores according to one of the following criteria:

STORE-11.1 all packets;

STORE-11.2 packets within a specified packet range;

STORE-11.3 packets within a specified time window.

3.14 Onboard Traffic Management

TRAFF-1 The Packet Assembly Controller (PAC) shall report the status of the process of re-assembly of telecommand packets from telecommand segments (see also **[AD3]**, section 7.3).

TRAFF-2 The Packet Assembly Controller (PAC) shall report on any problems relating to the re-assembly of telecommand packets from telecommand segments.

TRAFF-3 The onboard packet distribution system shall generate a report whenever a problem arises with the onboard traffic (e.g. a bottleneck in the distribution of telecommand packets or of telemetry source packets on the packet bus).

TRAFF-3.1 Such reports shall identify unambiguously the nature of the problem.

TRAFF-4 Adequate control capabilities shall be provided to permit the ground to resolve all pre-identified onboard problems relating to telecommand packet re-assembly (see also **[AD3]**, section 7.3), telemetry source packet segmentation or onboard traffic.

TRAFF-5 Packet bus management and resource parameters, such as average and peak bus loading, numbers of packet retransmissions etc., shall be routinely reported to the ground.

3.15 Packet Routing

Within the ESA packet telemetry and telecommand concept, the Application Process ID (APID) is used purely as an onboard address, without any functional meaning. This implies that, in principle, any onboard Application Process could host any type of Service (function) without restriction (the exceptions being APID=0, which is reserved by **[AD1]** for the Time Packet and APIDs which are allocated to CPDUs for telecommand distribution).

ROUTE-1* The Application Process ID shall be used purely as an onboard address indicating the source (telemetry source packets) or destination (telecommand packets) of packets.

ROUTE-2* To the maximum extent possible, different platform subsystems and payload(s) shall be assigned different Application Process IDs.

ROUTE-2.1 The assignment of Application Process IDs shall remain unchanged during the mission.

ROUTE-3 The same Application Process ID shall be used for related telemetry source and telecommand packets (since the Service/function to which these packets belong is necessarily below the level of the host APID).

ROUTE-4* Telecommand routing shall ensure that an onboard "blockage" resulting from payload commanding shall have no impact on platform commanding.

ROUTE-5* Virtual channels shall be used to allocate different downlink bandwidths (or, rather, different proportions of the currently available bandwidth) to different classes of data.

ROUTE-6 The assignment of virtual channels to different data classes shall not be changed during the mission.

ROUTE-7 The simultaneous transmission of real-time telemetry data and deferred telemetry data shall be possible.

ROUTE-8 For some missions, it may be possible to modify the allocation of onboard routes (e.g. a ground-modifiable table mapping APIDs to MAPs). In these cases, it shall also be possible to request a report of the onboard routing information.

ROUTE-9 In order to be able to unambiguously identify the source of a telecommand (e.g. different control centres on the ground), the source shall be explicitly indicated within the telecommand packet itself.

ROUTE-10 Request driven telemetry source packets (e.g. telecommand verification packets) shall only be routed to the originating source of the telecommand.

ROUTE-10.1 In this respect, all ground control centres shall constitute a single source from the viewpoint of the satellite ("the ground").

3.16 Testability

TEST-1 Each Application Process shall provide the capability to perform a set of end-to-end test functions which can be exercised under ground control.

TEST-1.1 It shall be possible to activate all such test functions by means of standard telecommand packets and to receive all the "test data" so generated by means of standard telemetry source packets.

TEST-1.2 An "are you alive" function shall be provided for testing the end-to-end connection between ground and an Application Process.

TEST-2 In order to be able to test onboard hardware redundancy, it shall be possible to operate redundant onboard functions involving hardware in an "off-line" manner, i.e. in parallel with, but without any disturbance to, the prime function. Note that, whilst this requirement relates essentially to the testing of hardware redundancy, depending on the onboard design, this may entail replication of any software element which is an integral part of the function.

TEST-2.1 Provision shall be made for the function to receive inputs and to produce telemetry outputs, whilst not providing any onboard Service or mission products.

4 Service Specification

4.1 Introduction

Chapter 3 identifies many conceptually related groups of operational requirements which imply distinct sets of capabilities to be implemented onboard the satellite along with corresponding monitoring and control facilities on the ground.

These sets of capabilities are considered to constitute "Services", the monitoring and control of which is achieved via a well-defined set of interactions between the provider of the Service (i.e. an onboard Application Process) and the user(s) of the Service (i.e. a ground system).

This document contains the specification of those standard Services which may be provided by onboard Application Processes in fulfilment of the operational requirements for satellite monitoring and control.

A Service specification will define:

- what can be requested of the Service;
- what can be issued by the Service to notify the user of the success or failure of his requests and to report events detected during the execution of the Service activities;
- what internal activities the Service should 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 must be functionally equivalent).

The following criteria have been used for identifying and defining these standard Services:

- ***Commonality:*** A Service should correspond to a group of functionalities commonly required by many missions.
- ***Coherence:*** The capabilities to be provided by a Service should be strongly and naturally related and their scope should be unambiguously specified. This generally means that a Service should encompass all the activities for managing inter-related state information and all activities which use that state information. It also means that 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 onboard architecture (hardware and/or software).

The remainder of this chapter contains:

- definitions and conventions which are used within the later Service specification chapters;
- items which are common between Services. This includes the overall structure of telecommand and telemetry source packets, which is defined elsewhere (**[AD1]**, **[AD3]**) and those standard elements of packets which are defined within this PUS;
- a summary of the standard Services defined within this Standard.

4.2 Definitions

The following definitions are used in the context of Service specification:

1. A **Service** denotes a set of onboard functions which provide the conceptually related Service capabilities which can be controlled and monitored by a ground system through a well-defined set of Service Requests and Reports.

   A Service may interact with other onboard functions for the execution of its activities; however, the specification of these internal interactions is beyond the scope of this Standard.

   The Minimum Capability Set denotes the set of Service capabilities and Service Requests and Reports which must be supported by all implementations of a Service.

   An Additional Capability Set denotes a set of Service capabilities and/or Service Requests and Reports which may optionally be supported by a given implementation of a Service.

2. The **Service Provider** denotes the onboard Application Process which executes the service activities and which is the destination of the Service Requests and the source of the Service Reports.

   If permitted, distinct instances of a Service may be provided by several onboard Application Processes.

   A given Service cannot be split across more than one Application Process; however, a given Application Process may provide several Services.

3. The **Service User**(s) denote the ground system(s) which initiates the Service Requests and receives the Service Reports.

   The specification of the activities performed by a Service User (e.g. to process a Service Report) is beyond the scope of this Standard.

4. An **Activity** denotes a precise set of actions performed by a Service Provider whose execution may either be requested by a Service User (**user-initiated activity**) or constitute part of the continuous activities required to accomplish the Service (**continuous activity**).

5. A **Service Request** denotes a data exchange between a Service User (the request initiator) and a Service Provider to request the execution of a particular activity.

   The invocation of an activity by a Service User results in the transmission of one or more telecommand packets or telecommand segments to the Service Provider, the reception of which initiates the corresponding activity.

   If several users can interact with a Service, then additional information must be 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.

   All user-initiated activities defined in this Standard are **optionally confirmed**. This means that the user invoking an activity indicates which level of acknowledgment (e.g. for telecommand verification) is required for each instance of activity invocation.

6. A **Service Report** denotes a 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**.

   The initiation of a Service Report by an onboard Service Provider results in the sending of one or more telemetry source packets to a Service User.

   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".

   For example, a Service Response is used to downlink the memory data corresponding to a memory dump Service Request.

   In all other cases, the Service Report is called a "Service Indication". A Service Indication is thus 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.

7. A **Supporting Service** is a Service which can only be used 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.

4.3 Conventions

In the following chapters, the specification of a Service consists of:

- An introduction containing the rationale for, and the scope of, the Service.
- 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 operational requirements.

The activity and information concept 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 need only implement those aspects required by the minimum capabilities set and the subset of additional capabilities which it supports.

- The definition of the structure and content of the Service Requests and Reports of the user initiated and continuous activities.

  An activity specification contains the following information:

  - 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 (or directly in the telecommand segment frame, in one instance) used to transport the request;

  - the actions to be executed by the Service Provider to perform the activity;

    indicate (a non-exhaustive set of) standard errors, identified in the specification of an activity. This does not preclude the identification of implementation-specific errors when a Service is designed and implemented for a particular mission.

  - for each Report (Response or Indication) which may 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.

- The definition of the minimum and additional capability sets of the Service, if any.

The Service data units generated by a Service Request or Report are preceded by their **Name** and their Service Type (x) and Service Sub-type (y) in the format **(x,y)**. The structure of the Service data unit is presented in a graphical tabular format. The first row of the table gives the name of the parameters of the Service data unit. The second row of the table indicates the type of the parameters. The allowable parameter types are defined in Chapter 23.

Where the presence of a field, or group of fields is optional, the corresponding fields are indicated below the table by: "Optional" or "Mission Optional". A field, or group of fields is Mission Optional if its presence is determined at the level of the mission, Application Process or Service instance. A field, or group of fields is Optional if its presence is determined by the value(s) of one or more preceding fields in the Service data unit.

The omission of an Optional or Mission Optional field implies that the value to be used is known by both the Service Provider and the Service User. For example, the Service may use a fixed value or a "current value" which may be set by the Service User through other means. The Service 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 array or a variable 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 array, N is given explicitly at the start of the array; in the case of a fixed array, N is known implicitly for the mission.

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"
BitString Parameter Fixed length bit string Variable length bit string	 "Fixed BitString" "Variable BitString"
OctetString Parameter Fixed length octet string Variable length octet string	 "Fixed OctetString" "Variable OctetString"
CharString Parameter Fixed length character string Variable length character string	 "Fixed CharString" "Variable CharString"
Time Parameter Absolute Time Parameter Relative Time Parameter	 "Time" "Relative Time"
Parameter with a deduced type	"Deduced"
A Data Structure following the Structure Rule Set 1 defined in chapter 23.	"Any"

4.4 Telecommand Packet Structure

All telecommand packets must conform to the structure defined in **[AD3]** and shown in Figure 4-1 below.

PACKET HEADER (48 bits)							**PACKET DATA FIELD (VARIABLE)**		
PACKET ID				PACKET SEQUENCE CONTROL		PACKET LENGTH	DATA FIELD HEADER (Optional)[1]	APPLIC-ATION DATA	PACKET ERROR CONTROL (Optional)[2]
Version Number (=0)	Type	Data Field Header Flag	Application Process ID	Sequence Flags	Sequence Count				
3	1	1	11	2	14				
16				16		16	Variable	Variable	16

Figure 4-1 Telecommand Packet Fields

Note 1: Although this field is indicated as optional in **[AD3]**, this Standard specifies it to be mandatory for all telecommand packets except for the CPDU telecommands described in **[AD7]** and Chapter 6 of this standard, which have no data field header.

Note 2: Although this field is indicated as optional in **[AD3]**, this Standard specifies it to be mandatory for all telecommand packets.

4.4.1 Packet Header

Packet ID (16 bits)

Version Number:

By changing the Version Number, future variations of the telecommand packet structure can be introduced. AT THE PRESENT, HOWEVER, ONLY ONE VERSION NUMBER IS PERMITTED: VERSION 0 (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, this standard requires that all telecommand packets must have a Data Field Header, for which this bit must be set to 1.

Application Process ID (APID):

The Application Process ID corresponds uniquely to an onboard Application Process which is the destination for this packet. The choice of Application Process ID values is mission-specific.

Packet Sequence Control (16 bits)

Sequence Flags:

The sequence flags are intended for future use if a series of packets needs to be sent in a particular sequence. This standard requires that, for all ESA missions, these 2 bits are set to "11", which means "stand-alone" packet.

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.

The Sequence Count is divided into 2 sub-fields, as follows:

Source ID:

The <NUM_SOURCE_BITS> most significant bits indicate the source of the telecommand (e.g. different control centres on the ground). For a given mission, this field may even be absent (i.e. <NUM_SOURCE_BITS> = 0), if the mission has only a single telecommand source. Note that a telecommand loaded from ground for later release from an onboard schedule nevertheless has the source ID of its origin on the ground.

Source Sequence Count:

The remaining bits of the Sequence Count field constitute a separate sequence count which is maintained by each telecommand source for each Application Process ID. The source sequence count will be used to unambiguously identify the corresponding telecommand.

When an acknowledgement of a packet is generated (see "Ack" field in the data field header below), it is mandatory that the packet sequence control field 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 is an unsigned integer "C" where:

C = (Number of octets in Packet Data Field) - 1

It should be noted that the actual length of the entire telecommand packet, including the Packet Header, is 6 octets longer. The largest possible telecommand packet length is 65542 octets.

4.4.2 Packet Data Field

Data Field Header (24 bits)

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 document, however all data field headers have the same basic structure, as follows:

TC Packet PUS Version Number	Checksum Type	Ack	Service Type	Service Sub-type
Enumerated (3 bits)	Enumerated (1 bit)	Enumerated (4 bits)	Enumerated (8 bits)	Enumerated (8 bits)

TC Packet PUS Version Number:

By changing the PUS Version Number, future variations of the telecommand packet can be introduced. AT THE PRESENT, HOWEVER, ONLY ONE PUS VERSION NUMBER IS PERMITTED: VERSION 0 (VALUE = 0).

Checksum Type:

This indicates which type of check is used for the Packet Error Control Field at the end of the Packet Data Field.

value = 0 means ISO standard 16-bit checksum (see Appendix A.3);
value = 1 means Cyclic Redundancy Code, CRC (see Appendix A.2).

Ack:

This field is used to indicate which acknowledgements, in the form of telecommand verification packets, are required to notify acceptance and to verify execution of this telecommand packet. This relates only to acknowledgement of successful acceptance and execution, since failure reports are generated by default; all zeroes means no success acknowledgements are required.

In addition to informing the onboard Application Process of the various telecommand verification packets required, the ground system will also know what stages of verification to expect for this telecommand packet and can report (as and when required) on "packets uplinked but not yet verified".

The bit settings corresponding to these stages are as follows:

---1 acknowledge acceptance of the packet by the Application Process
--1- acknowledge start of execution
-1-- acknowledge progress of execution
1--- acknowledge completion of execution (whether successful or failed)

A value of '1111' in the ack field would indicate that acknowledgement of ***at least*** 4 different steps in the telecommand execution are expected in the telemetry.

Note that acknowledgement packets are routed only to the source of the original telecommand. This means, for instance, that the ground only receives acknowledgements for those telecommands which originate from the ground and not for any telecommands which may be generated autonomously onboard.

Service Type:
This indicates the Service to which this packet relates.

Service Sub-type:
Together with the Service Type, the Sub-type uniquely identifies the nature of the Service Request constituted by this telecommand packet.

The definition of Service type and sub-type is unique across all Application Processes.

Service types 0 to 127 are reserved for this standard, whilst types 128 to 255 are mission-specific.

Service sub-types 0 to 127 are reserved for this standard, whilst sub-types 128 to 255 are mission-specific.

Application Data (Variable length)

The telecommand application data constitute the data element of the Service Requests from the Service user (the ground). The Application Data Field is a multiple of octets.

Packet Error Control (16 bits)

The purpose of the Packet Error Control field is to transport an error detection code (either a checksum or a CRC) so that the receiving Application Process can verify the integrity of the complete telecommand packet.

4.5 Telemetry Source Packet Structure

All telemetry source packets must conform to the structure defined within **[AD1]** and shown in Figure 4-2 below.

<table>
<tr><th colspan="6">PACKET HEADER (48 bits)</th><th colspan="3">PACKET DATA FIELD (VARIABLE)</th></tr>
<tr><td colspan="4">PACKET ID</td><td colspan="2">PACKET SEQUENCE CONTROL</td><td>PACKET LENGTH</td><td>DATA FIELD HEADER (Optional)[1]</td><td>SOURCE DATA</td><td>PACKET ERROR CONTROL (Optional)</td></tr>
<tr><td>Version Number (=4)</td><td>Type</td><td>Data Field Header Flag</td><td>Application Process ID</td><td>Segment-ation Flags</td><td>Source Sequence Count</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>11</td><td>2</td><td>14</td></tr>
<tr><td colspan="4">16</td><td colspan="2">16</td><td>16</td><td>Variable</td><td>Variable</td><td>Variable</td></tr>
</table>

Figure 4-2 Telemetry Source Packet Fields

Note 1: Although this field is indicated as optional in **[AD1]**, this Standard specifies it to be mandatory for all telemetry source packets except for the "Spacecraft Time Source Packet" described in **[AD1]** and Chapter 14 of this standard, which has no data field header.

4.5.1 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 THE PRESENT, HOWEVER, ONLY ONE VERSION NUMBER IS PERMITTED: VERSION 2 (VALUE = 4).

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, this Standard requires that all telemetry source packets must have a Data Field Header, for which this bit must be set to 1.

Application Process ID (APID):

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

Application Process ID = 0 is reserved by **[AD1]** for the Time Packet.
Application Process ID = "11111111111" (11 ones) is reserved for Idle packets.

Packet Sequence Control (16 bits)

Segmentation Flags:

The segmentation flags must be set to "11" to indicate that the telemetry source packet is "unsegmented".

Source Sequence Count:

A separate source sequence count is maintained for each Application Process ID and will be incremented by 1 whenever the source (APID) releases a packet. Therefore the counter corresponds to the order of release of packets by the source and enables the ground to detect missing packets. Ideally, this counter should never re-initialise; however, under no circumstances will 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 is an unsigned integer "C" where:

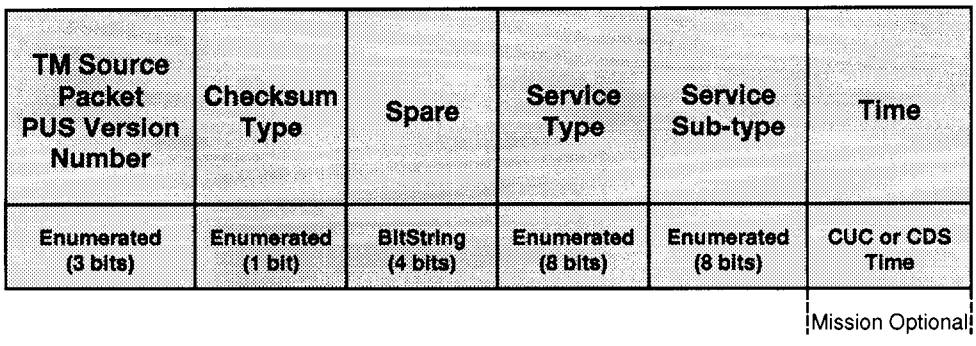
C = (Number of octets in Packet Data Field) - 1

It should be noted that the actual length of the entire telemetry source packet, including the Source Packet Header, is 6 octets longer.

4.5.2 Packet Data Field

Data Field Header (Variable)

The data field header is the subject of definition within this standard; its content depends on the nature of the Service Reports defined in the remainder of this document, however all data field headers have the same basic structure, as follows:



TM Source Packet PUS Version Number:

By changing the PUS Version Number, future variations of the telemetry source packet can be introduced. For example, it is foreseen to introduce a new version in the future for telemetry source packets which contain more than one Service data unit. AT THE PRESENT, HOWEVER, ONLY ONE PUS VERSION NUMBER IS PERMITTED: VERSION 0 (VALUE = 0).

Checksum Type:

This indicates if there is a Packet Error Control Field at the end of the Packet Data Field and, if so, which type of checksum is used.

value = 0 means ISO standard 16-bit checksum or no check included (the ground will know which);
value = 1 means Cyclic Redundancy Code (CRC).

Spare:

Reserved for future use.

Service Type:

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

Service Sub-type:

Together with the Service Type, the Sub-type uniquely identifies the nature of the Service Report constituted by this telemetry source packet.

The definition of Service type and sub-type is unique across all Application Processes and the combination of these fields may be used on the ground to determine the processing priority level.

Service types 0 to 127 are reserved for this standard, whilst types 128 to 255 are mission-specific.

Service sub-types 0 to 127 are reserved for this standard, whilst sub-types 128 to 255 are mission-specific.

Time:

This is the onboard reference time of the packet, expressed in either the CUC or CDS format (as defined in **[AD4]** and **[RD2]**). 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 and/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 parameters <APPL_TIME_CODE> of which there are as many as there are onboard Application Processes. The allowable representations of the time field are those defined in Chapter 23 of this standard.

If the CDS format is used, the standard CCSDS epoch of 1958 January 1 is applicable, see **[RD2]**.

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

Source Data (Variable)

The telemetry source data constitutes the data element of the Service Reports to the ground. Note that the Source Data field is a multiple of octets.

Packet Error Control (Variable)

The purpose of the Packet Error Control field is to transport an error detection code (either a checksum or a CRC) so that the ground can verify the integrity of the complete telemetry source packet.

4.6 Standard Services

The Standard Services listed in Table 4-1 below are specified within the following chapters. The Services may have a minimum capability set and one or more additional capability set(s); this is indicated in the corresponding Service Chapter.

For each Service, the minimum capability set corresponds to the minimum possible implementation of the Service which also remains sensible and coherent. In this respect, it is the "opposite" of the corresponding Service operational requirements (Chapter 3), which correspond to the maximum set of Service capabilities.

Service Type	Service Name
1	Telecommand Verification Service
2	Device-Level Commanding Service
3	Housekeeping & Diagnostic Data Reporting Service
4	Parameter Statistics Reporting Service
5	Event Reporting Service
6	Memory Management Service
7	Task Management Service
8	Function Management Service
9	Time Management Service
10	Time Reporting Service
11	Onboard Scheduling Service
12	Onboard Monitoring Service
13	Large Data Transfer Service
14	Packet Transmission Control Service
15	Onboard Storage and Retrieval Service
16	Onboard Traffic Management Service
17	Test Service

Table 4-1 Standard Services specified within this document

5 Telecommand Verification Service

5.1 Scope

The Telecommand Verification Service provides the capability for explicit verification of each distinct stage of execution of a telecommand packet, from onboard acceptance through to completion of execution. In this sense, it should be seen as 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, it should be understood that there is no implication that all telecommands for a given mission will necessarily be verifiable at each distinct stage. Indeed, for many telecommands, the Application Process may have little or no knowledge of the command execution profile or effects. The extent to which the telecommand verification process is supported onboard for a given telecommand will be specified on a mission-specific basis.

Telemetry source packets and telecommand packets belonging to the Telecommand Verification Service are denoted by **Service Type = 1**.

5.2 Service Concept

The following stages of telecommand processing are identified:

- **Acceptance** of the telecommand by the destination Application Process. At this stage of telecommand processing, it is assumed that all checks which can sensibly be applied to the telecommand packet prior to the start of execution are performed. Typically, this will include verification that the telecommand has not been corrupted (by checksumming the telecommand, where applicable) and that the Application Process supports the Service, and/or sub-service, requested etc. This element of the telecommand verification Service should therefore be supported for all telecommands.

- **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 will include feasibility and/or validity checks, such as confirmation that the telecommand parameters are correctly encoded, are within their allowed range of values and are valid for the current state of the Service.

- **Progress of Execution**. The various steps which reflect the progress of execution of the telecommand will be telecommand-specific in nature.

- **Completion of Execution** of the telecommand.

The telecommand verification Service will generate a report if a telecommand should fail at any one of its identified stages of execution. It will 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 will provide sufficient auxiliary data for the ground to fully understand the report (e.g. to identify the nature and cause of a telecommand failure).

For the purposes of selective routing of reports (both onboard and on ground), different sub-types are allocated to success and failure of telecommand verification.

5.3 Service Requests and Reports

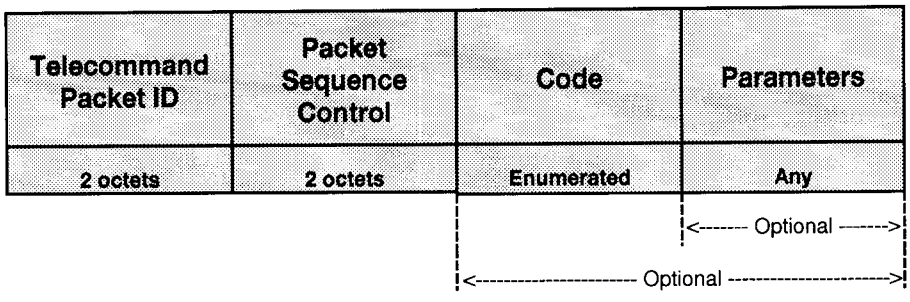
Currently no User-initiated Service Requests are defined.

5.3.1 Telecommand Acceptance

The reports of acceptance of a telecommand packet are as follows:

Telecommand Acceptance Report - Success, (1,1)
Telemetry Source Packet, Source Data:

Telecommand Acceptance Report - Failure, (1,2)
Telemetry Source Packet, Source Data:



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.

Code:
The code is an identifier, which may be command specific (e.g. dependent on the type and/or sub-type and/or address), for the auxiliary information provided with this report. In some instances, the code may be assigned "across the board" for a mission, e.g to denote a standard reason for failure of acceptance of the telecommand.

Parameters:
The parameters field provides complementary information relating to the particular value of the code field. Note that for full interpretation of failure of a command, knowledge may also be needed of the nature of the command. The parameters field complies with the Structure Rules set #1 and each parameter must be one of the data types defined in this Standard (see Chapter 23).

5.3.2 Telecommand Execution Started

The reports of start of execution of a telecommand packet are as follows:

Telecommand Execution Started Report - Success, (1,3)
Telemetry Source Packet, Source Data:

Telecommand Execution Started Report - Failure, (1,4)
Telemetry Source Packet, Source Data:

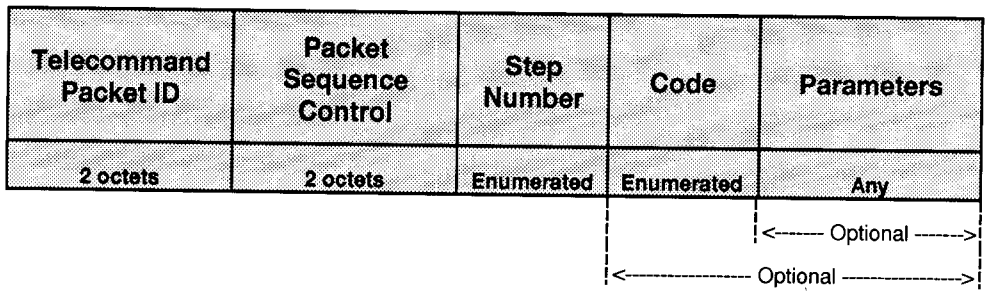
Same as for Sub-types 1 and 2

5.3.3 Telecommand Progress

The reports of progress of execution of a telecommand packet are as follows:

Telecommand Progress Report - Success, (1,5)
Telemetry Source Packet, Source Data:

Telecommand Progress Report - Failure, (1,6)
Telemetry Source Packet, Source Data:



5.3.4 Telecommand Execution Complete

The reports of completion of execution of a telecommand packet are as follows:

Telecommand Execution Started Report - Success, (1,7)
Telemetry Source Packet, Source Data:

Telecommand Execution Started Report - Failure, (1,8)
Telemetry Source Packet, Source Data:

Same as for Sub-types 1 and 2

5.4 Capability Sets

5.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Reports:

Sub-type	Service Request or Report
1	Telecommand Acceptance Report - Success
2	Telecommand Acceptance Report - Failure

Table 5-1 Summary of Telecommand Verification Service Minimum Capabilities

5.4.2 Additional Capability Sets

The Telecommand Verification Service may implement the following additional capabilities; note that these options may be effected at the level of individual telecommands.

Sub-type	Service Request, Report or Capability	Implication(s)
3	Telecommand Execution Started Report - Success	→sub-type 4
4	Telecommand Execution Started Report - Failure	→sub-type 3
5	Telecommand Execution Progress Report - Success	→sub-type 6
6	Telecommand Execution Progress Report - Failure	→sub-type 5
7	Telecommand Execution Complete Report - Success	→sub-type 8
8	Telecommand Execution Complete Report - Failure	→sub-type 7

Table 5-2 Summary of Telecommand Verification Service Additional Capabilities

6 Device Command Distribution Service

6.1 Scope

The Device Command Distribution Service provides the capability for device level commanding both at telecommand packet level and at telecommand segment level (i.e. before the packetisation layer).

Commanding at telecommand packet level comprises the distribution of On/Off and Register Load commands,

Commanding at telecommand segment level is considered to constitute part of this oonboardService, although, by definition, command processing (if any) prior to distribution is not performed by an Application Process.

The Service also provides the capability to re-configure vital spacecraft functions using Command Pulse Distribution Unit (CPDU) commands contained in a telecommand packet.

Telemetry source packets and telecommand packets belonging to the Device Command Distribution Service are denoted by **Service Type = 2**.

6.2 Service Concept

6.2.1 Device Commanding at Packet Level

Following reception of a Service Request containing On/Off commands or Register Load commands:

i) the commands are 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;

ii) the command(s) is (are) reformatted, if necessary, to the structure required by an internal command bus and/or the destination device driver;

iii) the command(s) is (are) distributed. If the packet contains more than one command, these are distributed in the same sequence in which they occur within the packet.

The Service Request is uplinked as one telecommand packet transferred using the AD Service of the Sequence Controlled Service defined in **[AD3]**.

An example of how a typical OBDH 24-bit command could be accommodated within this service data unit structure is given in Appendix A.4.

6.2.2 Device Commanding using a CPDU

As specified in Chapter 9 of **[AD7]**, 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 may have up to 256 output lines.

Each CPDU has a dedicated Application Process ID and can received "CPDU telecommand packets" containing Command Pulse instructions. A CPDU telecommand packet has no Data Field Header but has a Packet Error Control field and is required to be carried inside a single telecommand segment. The telecommand segment may be uplinked using either the AD or BD Service defined in **[AD3]**.

Following reception of a CPDU telecommand packet, it is checked as defined in **[AD7]** and if no error is found, the CPDU processes the Command Pulse instructions in the same sequence as that in which they occur within the packet.

For a given implementation, there will be a limit on the number of Command Pulse instructions in a CPDU telecommand packet (at least 12 and at most 120).

If the processing of a CPDU telecommand packet is not completed, then any new CPDU telecommand packet received by the CPDU is ignored.

6.2.3 Device Commanding at Telecommand Segment Level

Device level commanding at telecommand segment level may be required in non-nominal circumstances such as:

- when a Multiplexed Access Point (MAP) and/or a Packet Assembly Controller (PAC) on the satellite are not usable (e.g. in the event of a software anomaly);
- when an Application Process responsible for device level commanding at telecommand packet level is not available (e.g. a software anomaly or if the Application Process or its processor are not running).

Device commanding at telecommand segment level is performed by embedding one device command within a single telecommand segment. The telecommand segment may be uplinked using either the AD or BD Service defined in **[AD3]**.

The telecommand segment header identifies the MAP which is to be used for the uplink (note that it must be a dedicated MAP through which the uplink of telecommand packets is not permitted).

The telecommand segment data field contains the device command structured as required by an internal command bus and/or the destination device driver as described above.

On reception of the telecommand segment onboard, the device command is extracted and distributed (for example, using a poin-to-point hard-wired connection to the destination device driver). No destination Application Process is involved in the processing of the telecommand segment.

If a reliable uplink of the telecommand segment was required then the onboard system reports on the telecommand segment reception via the telemetry link. Otherwise the uplink of the transfer frame containing the telecommand segment is performed "in the blind".

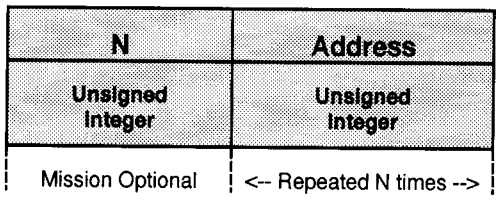
6.3 Service Requests and Reports

Currently no Service Reports are defined.

6.3.1 Distributing On/Off Commands

The request for the distribution of on/off command(s) by means of a telecommand packet is:

Distribute On/Off Commands, (2,1)
Telecommand Packet, Application Data:



N: The number of On/Off commands which follow (N > 0).

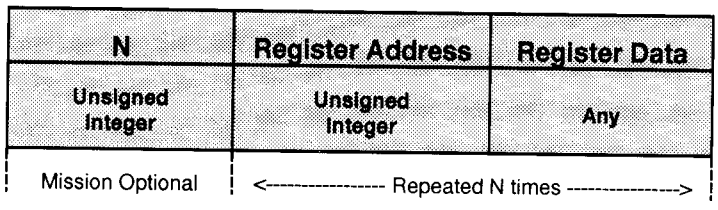
This field is systematically omitted if the Service only supports the distribution of one On/Off command at a time (i.e. N=1).

Address:
This gives the hardware address to which the On/Off command is to be routed.

6.3.2 Distributing Register Load Commands

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

Distribute Register Load Commands, (2,2)
Telecommand Packet, Application Data:



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

This field is 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.

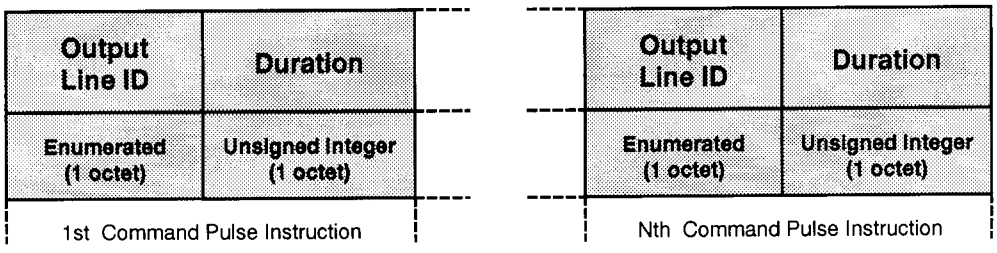
Register Data:

The register data consist of a set of parameters whose structure is known implicitly from the foregoing register address and which conform with the Structure Rule Set #1 (see Chapter 23).

6.3.3 Distributing CPDU Commands

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

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 can take are mission specific.

Duration:

A value between 0 and 8 which determines the duration of the Command Pulse as follows:

$$\text{Command Pulse duration} = \text{Duration\_Unit} * 2^{\text{Duration}}$$

where Duration_Unit is defined for the CPDU (between 10 and 15 ms).

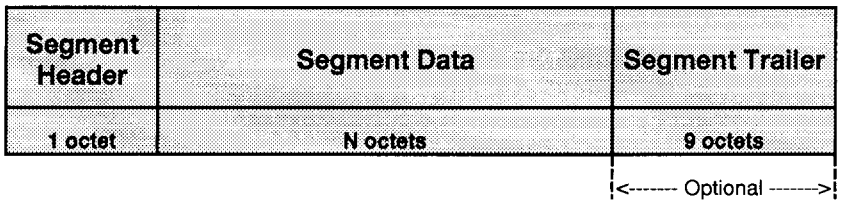
The number of Command Pulse Instructions in the CPDU telecommand packet is variable.

On reception of this request, the CPDU performs the processing outlined in Section 6.2.2 and specified in **[AD7]**.

6.3.4 Device Commanding at Telecommand Segment Level

The request for the uplink of a device command by means of a telecommand transfer frame containing a telecommand segment whose structure is fully defined in **[AD3]** is as follows:

Distribute Command in Telecommand Segment, (2,4):



Segment Header:

This contains a "Sequence Flags" field (2 bits) with the binary value "11", indicating a standalone segment, followed by a 6-bit MAP Identifier (see **[AD3]**).

Segment Data:

This field contains the hardware address of the command and the command data (if any). The field is formatted as required by an internal command bus and/or the destination device driver. The structure of the field is known implicitly from the hardware address.

Segment Trailer:

If present, this contains an Authentication Tail (see **[AD3]**).

6.4 Capability Sets

6.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Request:

Sub-type	Service Request or Report
1	Distribute On/Off Commands
2	Distribute Register Load Commands
3	Distribute CPDU Commands

Table 6-1 Summary of Device Command Distribution Service Minimum Capabilities

6.4.2 Additional Capability Sets

The Device Command Distribution Service may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
4	Distribute Command in Telecommand Segment	

Table 6-2 Summary of Device Command Distribution Service Additional Capabilities

7 Housekeeping and Diagnostic Data Reporting Service

7.1 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 which is not explicitly provided within the reports of another Service.

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

i) housekeeping data reporting (both periodic and filtered in nature);

ii) diagnostic data reporting.

Each onboard Application Process provides a single instance of this Service.

Telemetry source packets and telecommand packets relating to the Housekeeping and Diagnostic Data Reporting Service are denoted by **Service Type = 3**.

7.2 Service Concept

7.2.1 Housekeeping Service Concept

The housekeeping data reporting sub-service samples sets of housekeeping parameters in accordance with a set of reporting definitions stored onboard. There will be a pre-defined set of such definitions onboard (designed by the satellite manufacturer) as deemed appropriate 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 at any time.

A Structure Identification (SID) is associated with each distinct reporting definition and associated housekeeping-parameter report. The SID will be used on the ground, together with the Application Process ID, Service Type and Sub-type to identify the housekeeping parameters report and to interpret its content.

7.2.1.1 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 may be either simply commutated (sampled once per collection interval) or super-commutated (sampled more than once). Supercommutated parameters are sampled regularly in time at a rate which is determined onboard from the combination of the data collection interval and the parameter commutation index.

7.2.1.2 Parameter-Report Generation

Two modes of generating housekeeping parameter reports exist:

i) **periodic mode**, where a housekeeping-parameter report is generated once each collection interval;

ii) **"filtered" mode**, where a housekeeping-parameter report is only generated at the end of a collection interval when the value of one or more specified (contained) parameters has changed by more than a specified threshold since the previous collection interval. In the case of simply commutated parameters, the value from the current collection interval is compared with that from the previous collection interval. In the case of supercommutated parameters, the change of value of any sample 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 oonboardprocessing load).

The requests for selecting the mode of housekeeping-parameter report generation are part of this service, but the requests for disabling and enabling the transmission of housekeeping-parameter reports to the ground are part of the generic Packet Transmission Control Service (see Chapter 18).

Filtered mode is foreseen to be useful primarily for status-type parameters, where only a significant change of status is to be reported to the ground. However, in principle, any parameter within the reporting definition can be used to trigger a filtered parameter report. Those parameters that are not to be used for this purpose are "masked out" from the comparison process whilst all unmasked parameters are used. Also, to limit the circumstances under which the parameter report is generated, a threshold for the change is defined, on a global basis, and expressed as a proportion of "full-scale" value.

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

7.2.1.3 Parameter Sampling Times

It is required that the absolute (as well as relative) sampling times of parameters in housekeeping-parameter reports can be determined to a given accuracy <PARAM_ABS_SAMPL_TIME> (and <PARAM_REL_SAMPL_TIME>) on the ground. Three alternative methods are foreseen to satisfy these requirements:

i) The parameter sampling times (e.g. time offsets with respect to packet time) may be deducible from a knowledge of the onboard parameter sampling mechanism;

ii) Telemetry parameters may be **explicitly** datated in the telemetry. For example, a datation counter may be provided, which is sampled and telemetered along with each sample of the parameter. Alternatively, the Application Process responsible for sampling may measure the sampling time and insert this time in the telemetry. Both are expensive solutions, however, in terms of packet overhead;

iii) The sampling time offsets (with respect to packet time) are measured (or otherwise evaluated) onboard, over a given collection interval and reported to the ground. The sampling "pattern" and hence the time offsets are then assumed to hold true for all housekeeping-parameter reports.

For case iii), explicit reporting (on request) of the measured time offsets is supported.

7.2.2 Diagnostic Service Concept

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

Because of the nature of diagnostic mode, it is anticipated that the corresponding parameter reports will contain a predominance of fixed arrays corresponding to parameters sampled at very high rates many times per report.

For this reason, it may be important that the ground has the possibility to disable the generation of certain diagnostic parameters reports (whose definitions may remain onboard for intermittent use, for example when a particular anomaly occurs).

7.3 Service Requests and Reports

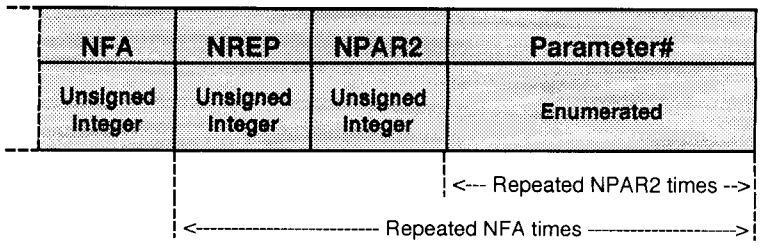
7.3.1 Defining New Housekeeping or Diagnostic Parameter Reports

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

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
	Mission Optional		<--- Repeated NPAR1 times -->

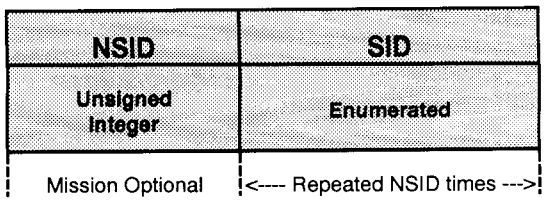


7.3.2 Clearing Housekeeping- or Diagnostic-Parameter Report Definitions

The requests to clear one or more housekeeping-parameter report definitions (or diagnostic-parameter report definitions) are:

Clear Housekeeping-Parameter Report Definitions, (3,3)
Telecommand Packet, Application Data:

Clear Diagnostic-Parameter Report Definitions, (3,4)
Telecommand Packet, Application Data:



NSID
This field is 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 are cleared (released).

7.3.3 Controlling the Generation of Housekeeping- or (Diagnostic-) Parameter Reports

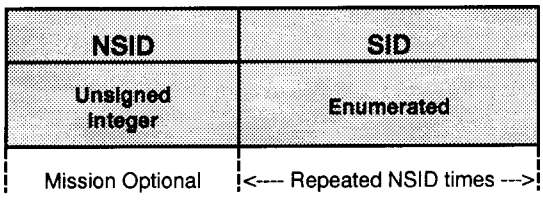
The requests to enable and disable the generation of one or more housekeeping- (diagnostic-) parameters report definitions are:

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
This field is systematically omitted if the Service can only enable/disable the generation of one parameter report definition at a time (i.e. NSID=1).

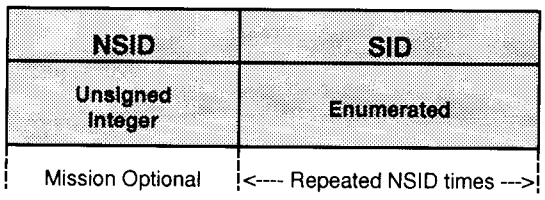
When this request is received, the activities for the generation of the indicated housekeeping- (diagnostic-) parameter reports are started (stopped).

7.3.4 Reporting Housekeeping- or Diagnostic-Parameter Report Definitions

The requests for a report of one or more housekeeping-parameter report definitions (or diagnostic-parameter report definitions) are:

Report Housekeeping-Parameter Report Definitions, (3,9)
Telecommand Packet, Application Data:

Report Diagnostic-Parameter Report Definitions, (3,11)
Telecommand Packet, Application Data:



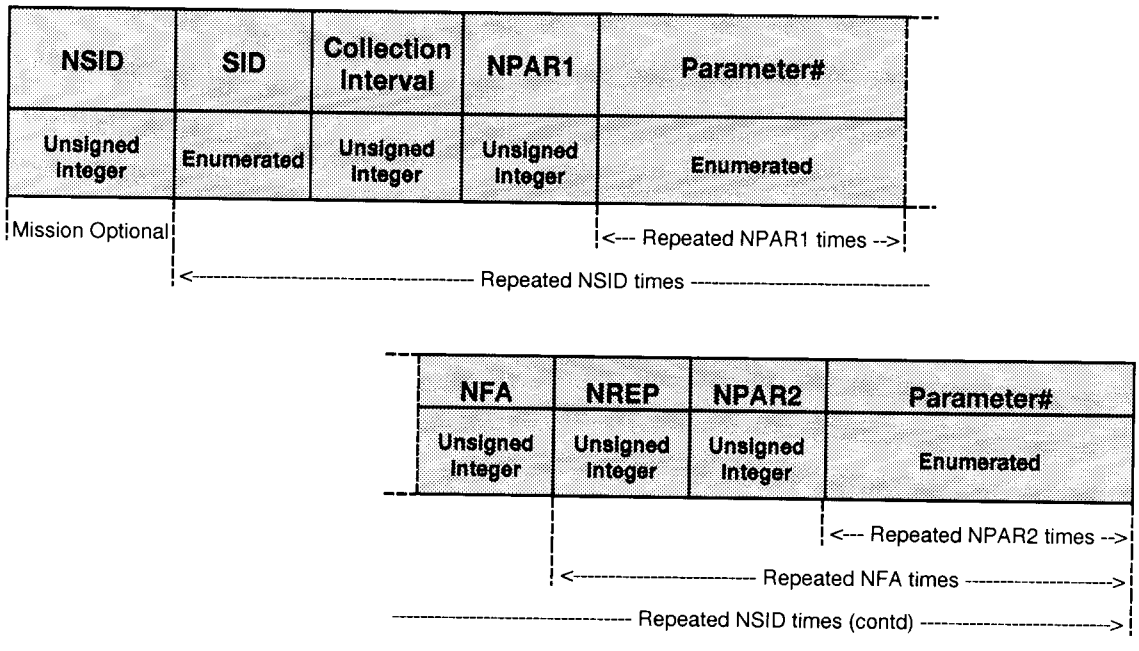
NSID

This field is systematically omitted if the Service can only report one parameters report definition at a time (i.e. NSID=1).

When this request is received, the following response is generated containing the requested housekeeping-parameter report definitions (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 is systematically omitted if the Service can only report one parameter-report definition at a time (i.e. NSID=1).

The definitions (SIDs) are reported in the same sequence as in the corresponding request (Sub-type 9 or 10 above).

7.3.5 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 are:

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 offsets for the next housekeeping (or diagnostic) parameters report of this reporting definition (Structure ID) are measured (or evaluated) and reported as follows. This report is also 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 array, only the time offset of the first occurrence of a parameter within the array is reported (the time offsets of subsequent samples are derivable from a knowledge of the parameter sampling interval).

7.3.6 Selecting Housekeeping- (or Diagnostic-) Parameter Report Generation Mode

The requests to select the Periodic Generation Mode for a given housekeeping (or diagnostic) reporting definition are:

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 is changed to periodic mode. If the reporting definition is already in periodic mode, a standard error report is generated.

The requests to select the Filtered Generation Mode for a given housekeeping (or diagnostic) reporting definition are:

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	Threshold	N	Parameter #
Enumerated	Unsigned Integer	Unsigned Integer	Unsigned Integer	Enumerated
	Mission Optional	Mission Optional	Mission Optional	<--- Repeated N times -->

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 is systematically omitted if the Service uses a default value for the given housekeeping (or diagnostic) reporting definition.

Threshold

The minimum change to be detected between parameter samples in order to declare an event for the purpose of parameters report generation. This threshold applies to all unmasked parameters in the reporting definition. Its interpretation will be mission-specific, however it could represent for example a proportion (e.g. percentage) of the full-scale value of the parameter or the number of Least Significant Bits (LSBs) of the parameter which should be ignored.

This field is systematically omitted if the Service uses a default value for the given housekeeping (or diagnostic) reporting definition.

N The number of parameters which follow (it may take the value 0).

This field is systematically omitted if the Service does not support the notion of masked parameters or has a knowledge of the list of masked parameters to use for the given housekeeping (or diagnostic) reporting definition.

Parameter #

The parameters which are to be masked (i.e. not used) when the values of samples between successive collection intervals are compared.

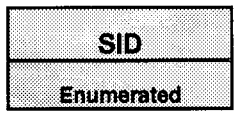
When this request is received, the parameter report generation mode for the indicated housekeeping (or diagnostic) reporting definition is changed to filtered mode and the new Service parameters supplied in the request are used with immediate effect. If the reporting definition is already in filtered mode, a standard error report is generated.

7.3.7 Reporting Masked Housekeeping- or Diagnostic- Parameters

The request to report the parameters which are masked (i.e. not used) for a given housekeeping (or diagnostic) reporting definition when the values of samples between successive collection intervals are compared is:

Report Masked Housekeeping-Parameters, (3,21)
Telecommand Packet, Application Data:

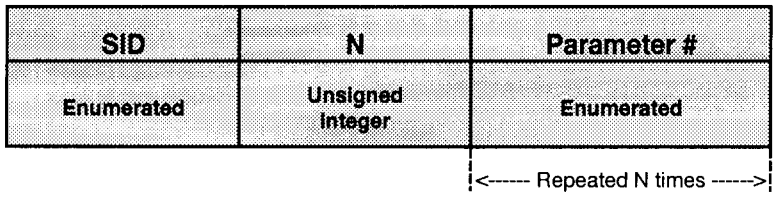
Report Masked Diagnostic-Parameters, (3,22)
Telecommand Packet, Application Data:



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

Masked Housekeeping Parameters Report, (3,23)
Telemetry Source Packet, Source Data:

Masked Diagnostic Parameters Report, (3,24)
Telemetry Source Packet, Source Data:

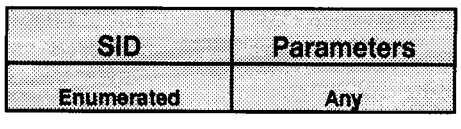


7.3.8 Reporting Housekeeping or Diagnostic Data

The Provider-initiated reports of the values of a set of housekeeping (or diagnostic) parameters are:

Housekeeping-Parameter Report, (3,25)
Telemetry Source Packet, Source Data:

Diagnostic-Parameter Report, (3,26)
Telemetry Source Packet, Source Data:



Parameters:

This field consists of a sequence of values of simply commutated housekeeping (or diagnostic) parameters followed by a sequence of fixed arrays of records and/or parameter values. The sequence of parameter values and arrays is the same as in the housekeeping- (or diagnostic-) parameter report definition request (see Sub-types 1 and 2 above).

The only authorised parameter types are those described in Chapter 23.

For ground processing purposes, the SID, together with the Application Process ID, Service Type and Sub-type implicitly identifies the structure of the parameter field.

7.4 Capability Sets

7.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Requests and Reports:

Sub-type	Service Request or Report
2	Define New Diagnostic-Parameter Report
4	Clear Diagnostic-Parameter Report Definitions
25	Housekeeping-Parameter Report
26	Diagnostic-Parameter Report

Table 7-1 Summary of Housekeeping & Diagnostic Data Reporting Minimum Capabilities

7.4.2 Additional Capability Sets

A Housekeeping and Diagnostic Data Reporting Service may also implement the following additional capability sets:

- report definitions control;
- report definitions reporting;
- sampling time offset reporting;
- filtered reports.

A Housekeeping and Diagnostic Data Reporting Service providing the report definitions control capability set implements *at least one* of the following pairs of capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
1	Define New Housekeeping-Parameter Report	→sub-type 3
3	Clear Housekeeping-Parameter Report Definitions	→sub-type 1
5	Enable Housekeeping-Parameter Report Data Collection	→sub-type 6
6	Disable Housekeeping-Parameter Report Data Collection	→sub-type 5
7	Enable Diagnostic-Parameter Report Data Collection	→sub-type 8
8	Disable Diagnostic-Parameter Report Data Collection	→sub-type 7

Table 7-2 Summary of Report Definitions Control Additional Capabilities

A Housekeeping and Diagnostic Data Reporting Service providing the report definitions reporting capability set implements *all* the following capabilities:

Sub-type	Service Request, Report or Capability
9	Report Housekeeping Parameters Report Definitions
10	Report Diagnostic Parameters Report Definitions
11	Housekeeping Parameters Report Definitions Report
12	Diagnostic Parameters Report Definitions Report

Table 7-3 Summary of Report Definitions Reporting Additional Capabilities

A Housekeeping and Diagnostic Data Reporting Service providing the sampling time offset reporting capability set implements *all* the following capabilities:

Sub-type	Service Request, Report or Capability
13	Report Housekeeping Parameter Sampling Time Offsets
14	Report Diagnostic Parameter Sampling Time Offsets
15	Housekeeping Parameter Sampling Time Offsets Report
16	Diagnostic Parameter Sampling Time Offsets Report

Table 7-4 Summary of Sampling Time Offset Reporting Additional Capabilities

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 the filtered reports capability set implements *all* the following capabilities:

Sub-type	Service Request, Report or Capability
17	Select Periodic Housekeeping Parameter Report Generation Mode
18	Select Periodic Diagnostic Parameter Report Generation Mode
19	Select Filtered Housekeeping Parameters Report Generation Mode
20	Select Filtered Diagnostic Parameters Report Generation Mode

Table 7-5 Summary of Filtered Reports Minimum Capabilities

A Housekeeping and Diagnostic Data Reporting Service providing the filtered reports capability set may implement *all* the following capabilities:

Sub-type	Service Request, Report or Capability
21	Report Masked Housekeeping Parameters
22	Report Masked Diagnostic Parameters
23	Masked Housekeeping Parameters Report
24	Masked Diagnostic Parameters Report

Table 7-6 Summary of Filtered Reports Additional Capabilities

8 Parameter Statistics Reporting Service

8.1 Scope

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

This service may be used during periods of ground non-coverage, e.g. as an alternative to housekeeping data reporting if the onboard storage capacity is limited.

Any number of onboard Application Processes may provide a single instance of this Service.

Telemetry source packets and telecommand packets relating to the Parameter Statistics Reporting Service are denoted by **Service Type = 4**.

8.2 Service Concept

The service provides the capability for reporting the maximum, minimum, mean and standard deviation values of a list of parameters. The ground 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 may be sampled at quite different frequencies, depending on the particular characteristics of the parameter (for example, a slowly varying analogue parameter such as a temperature may be sampled at a very low frequency). This sampling interval (per parameter basis) is specified when the parameter is added to the list.

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

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

Thus, if it desired (for example) to evaluate these parameter statistics during a period of ground non-coverage, the ground will 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.

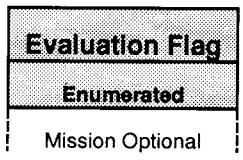
Because of the potentially long interval between reports, the packet time should indicate the report generation time and not the beginning of the evaluation interval.

8.3 Service Requests and Reports

8.3.1 Reporting the Parameter Statistics Results

The request is:

Report Parameter Statistics, (4,1)
Telecommand Packet, Application Data:



Evaluation Flag:

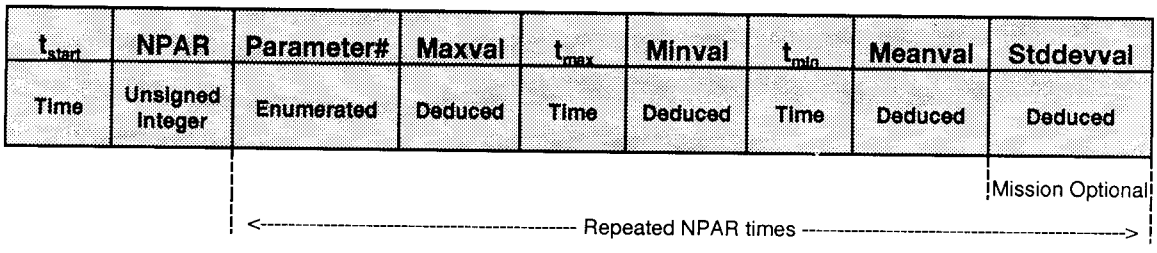
This indicates whether the evaluation of the parameter statistics is to be reset or not. Its possible values are "Reset" (value=1) and "Continue" (value=0).

This field is systematically omitted if the Service is designed always to reset the evaluation of the parameter statistics.

When this request is received, a report is generated which contains the current parameter statistics values. The evaluation of the parameter statistics is reset if the Evaluation Flag is "Reset".

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

Parameter Statistics Report, (4,2)
Telemetry Source Packet, Source Data:



t_{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.

8.3.2 Resetting the Parameter Statistics Reporting

The request is:

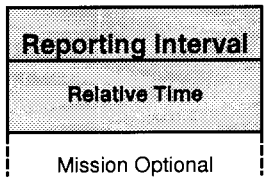
Reset Parameter Statistics Reporting, (4,3)
Telecommand Packet, Application Data: **None**

When this request is received, the evaluation of the parameter statistics is reset immediately, that is to say the current set of values is discarded and the evaluation starts again from scratch.

8.3.3 Selecting the Parameter Statistics Reporting Mode

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

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 must be greater than the sampling interval of any parameter currently in the parameter statistics list.

This field is systematically omitted if the Service uses a default value for the reporting interval.

When this request is received, the Service starts the periodic reporting and resetting of the parameter statistics using the specified reporting interval (if any). Note that whilst in periodic reporting mode, the ground can still explicitly request a report at any time.

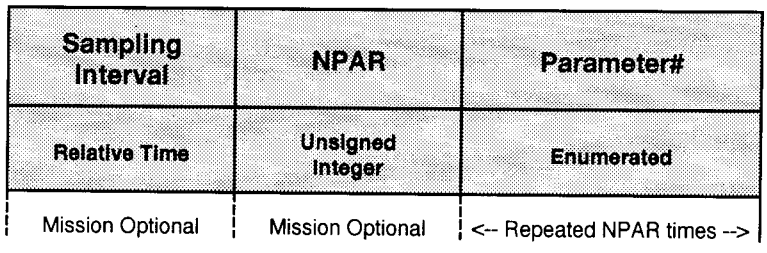
Disable Periodic Parameter Statistics Reporting, (4,5)
Telecommand Packet, Application Data: **None**

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

8.3.4 Adding Parameters to the Parameter Statistics List

The request is:

Add Parameters to Parameter Statistics List, (4,6)
Telecommand Packet, Application Data:



Sampling Interval:
The sampling interval to use for the set of parameters which follows. If the parameter statistics reporting mode is currently periodic, the sampling interval must be smaller than the reporting interval.

This field is systematically omitted if the Service knows which value to use.

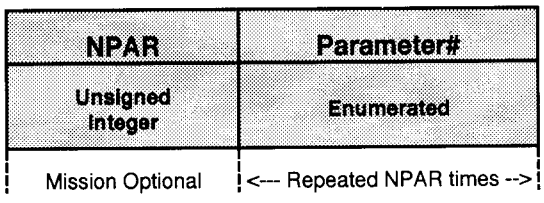
NPAR:
This field is systematically omitted if the Service can only add one parameter at a time.

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

8.3.5 Deleting Parameters from the Parameter Statistics List

The request is:

Delete Parameters from Parameter Statistics List, (4,7)
Telecommand Packet, Application Data:



NPAR:
This field is systematically omitted if the Service can only delete one parameter at a time.

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

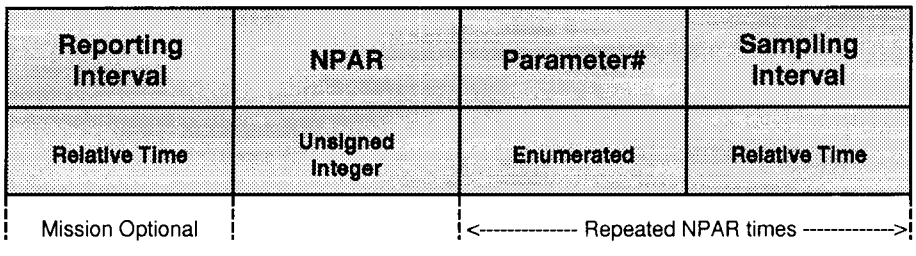
8.3.6 Reporting the Parameter Statistics List

The request is:

Report Parameter Statistics List, (4,8)
Telecommand Packet, Application Data: **None**

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

Parameter Statistics List Report, (4,9)
Telemetry Source Packet, Source Data:



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 is 0 seconds.

This field is systematically omitted if the Service does not support the concept of periodic reporting.

8.3.7 Clearing the Parameter Statistics List

The request is:

Clear Parameter Statistics List, (4,10)
Telecommand Packet, Application Data: **None**

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

8.4 Capability Sets

8.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Requests and Reports:

Sub-type	Service Request, Report or Capability
1	Report Parameter Statistics
2	Parameter Statistics Report
3	Reset Parameter Statistics Reporting

Table 8-1 Summary of Parameter Statistics Reporting Minimum Capabilities

8.4.2 Additional Capability Sets

The Parameter Statistics Reporting Service may also implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
4	Enable Periodic Parameter Statistics Reporting	→sub-type 5
5	Disable Periodic Parameter Statistics Reporting	→sub-type 4
6	Add Parameters to Parameter Statistics List	→sub-type 7 or sub-type 10
7	Delete Parameters from Parameter Statistics List	→sub-type 6
8	Report Parameter Statistics List	→sub-type 9
9	Parameter Statistics List Report	→sub-type 8
10	Clear Parameter Statistics List	→sub-type 6

Table 8-2 Summary of Parameter Statistics Reporting Additional Capabilities

9 Event Reporting Service

9.1 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.:

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

Some reports may combine more than one of these events; e.g. a report may declare that "Unit X has been switched off because its temperature was detected as 31°C, where the currently defined limit is 30°C".

Any number of onboard Application Processes may provide a single instance of this Service.

Telemetry source packets and telecommand packets relating to the Event Reporting Service are denoted by **Service Type = 5**.

9.2 Service Concept

This service provides the capability for the transmission of reports generated by any oonboardfunction to notify the ground of an event of operational significance. Four distinct levels of event report are provided:

i) normal/progress reports (used, for example, to notify the ground of an onboard autonomous action, which does not relate to a fault condition);

ii) error/anomaly report - low severity;

iii) error/anomaly report - medium severity;

iv) error/anomaly report - high severity.

9.3 Service Requests and Reports

Currently no User-initiated Service Requests are defined.

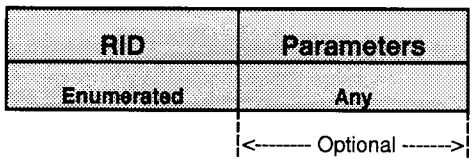
The different levels of provider-initiated event report are allocated different sub-types, to facilitate routing and/or ground processing. All reports have the same structure, as follows.

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:

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

Parameters:

The parameters field provides complementary information relating to the particular value of the Report ID. For these four Sub-types, the parameters field must comply with Structure Rules set #1 and each parameter must be one of the standard data types (see Chapter 23).

9.4 Capability Sets

9.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Reports:

Sub-type	Service Request or Report
1	Normal/Progress Report
2	Error/Anomaly Report - Low Severity
3	Error/Anomaly Report - Medium Severity
4	Error/Anomaly Report - High Severity

Table 9-1 Summary of Event Reporting Minimum Capabilities

9.4.2 Additional Capability Sets

There are no additional capability sets.

10 Memory Management Service

10.1 Scope

This Service relates to the management of the various memory blocks (which may be RAM, ROM, disk, etc.) which exist onboard 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.

Any number of onboard Application Processes may host a single instance of a Memory Management Service; however, the number of instances must ensure that any onboard changeable memory area can be loaded and that any onboard memory area can be dumped.

Telemetry source packets and telecommand packets relating to the Memory Management Service are denoted by **Service Type = 6**.

10.2 Service Concept

The memory management service affords quite basic dump, load and check facilities which do not require the maintenance of service state information. It is required only to read from, write to and checksum areas of, the memory blocks.

For example, the service need not know which memory areas are changeable, what are the sizes of the memory blocks or what types of information reside in particular parts of the memory blocks.

The integrity of the data to be loaded (or dumped) is ensured by systematically utilising the Packet Error Control field (see Section 4.4).

A "Memory ID" uniquely identifies each onboard memory block. It explicitly or implicitly denotes, for example, a physical onboard memory of an onboard processor or the virtual memory of an Application Process (which may be mapped onto physical onboard memories and disk spaces).

For a given onboard memory block, there is a unique addressing technique used for all memory load, dump and check requests and reports. This addressing technique consists either of a base reference plus an offset or an absolute address.

The memory management service may 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 Chapter 17) when the transaction for dumping (or loading) memory areas would exceed the maximum size of a single telemetry source packet (or telecommand packet) defined for the mission.

The following aspects are beyond the scope of this standard (nevertheless, decisions must be made during the design of the service for a given mission):

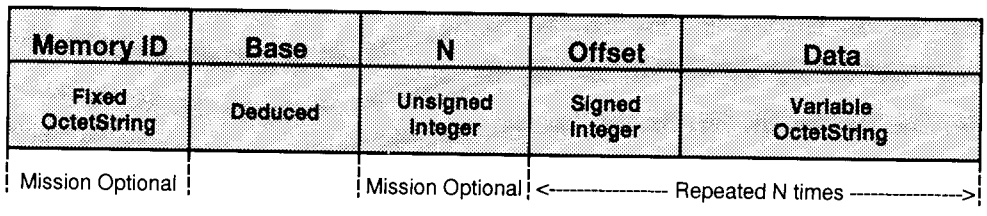
- the service may guarantee the consistency of the contents of the memory areas requested to be dumped (e.g. by ensuring that they are not written by any Application Process during the reading activity or by duplicating the contents);
- the service may guarantee that no Application Process will read an inconsistent set of data from the memory areas which are being modified by a loading activity;
- the service may limit the amount of memory data which may be loaded or dumped in a single transaction;
- the service may use a selected sub-set of capabilities of the Large Data Transfer Service (e.g. transfer without automatic re-transmission).

10.3 Service Requests and Reports

10.3.1 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 is:

Load Memory using Base plus Offsets, (6,1)
Telecommand Packet, Application Data:



Memory ID:
This identifies the destination memory block (note that it may be interpreted as a CharString 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 is systematically omitted if the Service only manages one onboard memory block.

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

The type of the Base field is one of the following:

- an unsigned integer;
- a symbolic base reference which is a fixed length character string.

The type and length may be deduced, for example, 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 section 23.4).

N: The number of data blocks to be loaded.

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

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

Data: A data block to be loaded (in increasing order of octet).

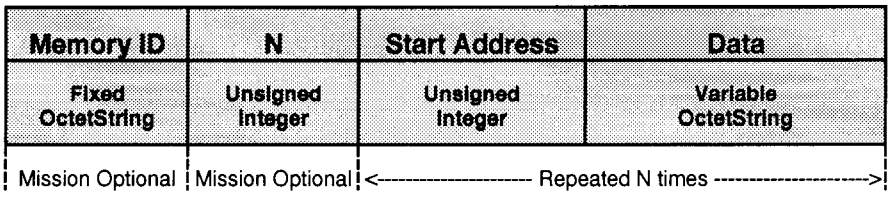
When the Service Provider receives this request:

- It calculates the checksum of the telecommand packet data (as specified in the Packet Data Header) and compares the result with the value in the Packet Error Control Field. The Service may be designed to skip the following steps if the checksum comparison fails.
- It writes each data block into the memory, at the specified offset from the base reference.
- If the completion of execution of the request is to be reported, the memory areas just written to are re-read, a checksum is calculated and compared with the value in the Packet Error Control Field. The result is reported.

10.3.2 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 is:

Load Memory using Absolute Addresses, (6,2)
Telecommand Packet, Application Data:



Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

N: This field is 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 octets, with the count starting from zero) within the memory block for loading the data.

The field length may be deduced in a similar way to the "Base" field of Sub-type 1 above.

When the Service Provider receives this request, it calculates the checksum of the received data, writes each data block to the memory at the specified start address and, if requested, re-reads the memory area just written to, calculates, compares and reports the checksum.

10.3.3 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 is:

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
Mission Optional		Mission Optional	<------ Repeated N times ------>	

Memory ID:

This field is systematically omitted if the Service only manages one onboard memory block.

N: This field is systematically omitted if the Service can only dump a single continuous area of memory (i.e. N=1).

Length:

The number of octets to be dumped.

When the Service Provider receives this request it reads each indicated area of the memory block, generates a report containing the contents of these areas and downlinks it (possibly 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	Data
Fixed OctetString	Deduced	Unsigned Integer	Signed Integer	Variable OctetString
Mission Optional		Mission Optional	<---------- Repeated N times ---------->	

Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

N: This field is systematically omitted if the Service can only dump a single continuous area of memory (i.e. N=1).

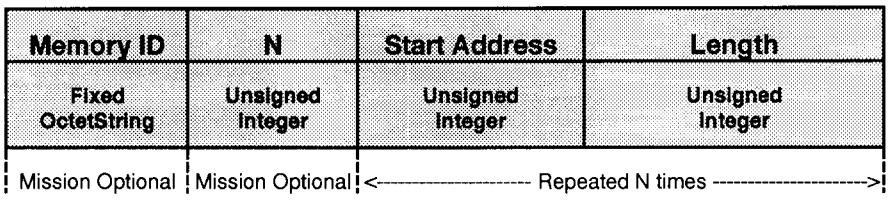
Data:
These are the values of the memory locations being dumped, starting from the octet position indicated by the initial request (in increasing order of octet).

The checksum of the telemetry source packet data is calculated and placed in the Packet Error Control Field. The type of checksum to be calculated is fixed for the Service Provider.

10.3.4 Dumping Memory using Absolute Addresses

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

Dump Memory using Absolute Addresses, (6,5)
Telecommand Packet, Application Data:

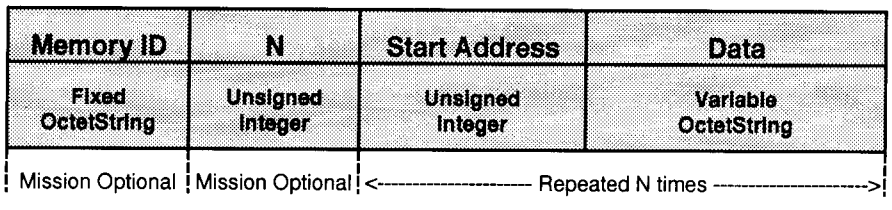


Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

N: This field is 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 reads each indicated area of the memory block, generates a report containing the contents of these areas and downlinks it (possibly using the Large Data Transfer Service).

Memory Dump using Absolute Addresses Report, (6,6)
Telemetry Source Packet, Source Data:



Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

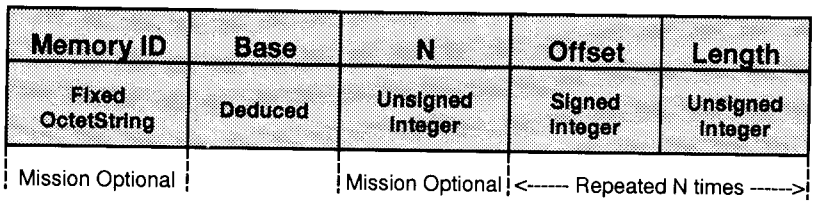
N: This field is systematically omitted if the Service can only dump a single continuous area of memory (i.e. N=1).

The checksum of the telemetry source packet data is calculated and placed in the Packet Error Control Field.

10.3.5 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 is:

Check Memory using Base plus Offsets, (6,7)
Telecommand Packet, Application Data:

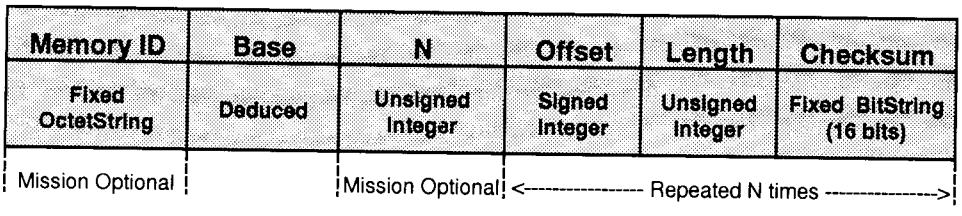


Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

N: This field is 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 reads and computes the checksum value of each indicated area of the memory block using the ISO standard 16-bit checksum algorithm defined in Appendix A.3. It then generates a report containing the checksum values computed.

Memory Check using Base plus Offsets Report, (6,8)
Telemetry Source Packet, Source Data:



Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

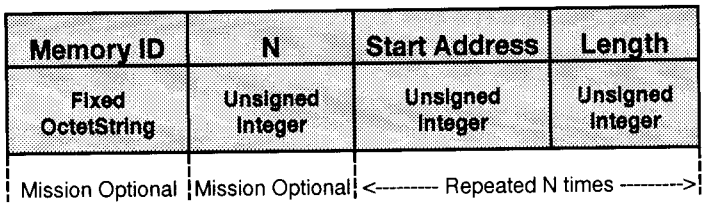
N: This field is systematically omitted if the Service can only check a single continuous area of memory (i.e. N=1).

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

10.3.6 Checking Memory using Absolute Addresses

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

Check Memory using Absolute Addresses, (6,9)
Telecommand Packet, Application Data:

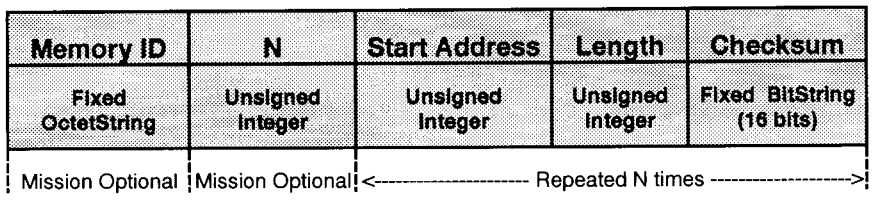


Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

N: This field is 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 reads and computes the checksum value of each indicated area of the memory block using the ISO standard 16-bit checksum algorithm defined in Appendix A.3. It then generates a report containing the checksum values computed.

Memory Check using Absolute Addresses Report, (6,10)
Telemetry Source Packet, Source Data:



Memory ID:
This field is systematically omitted if the Service only manages one onboard memory block.

N: This field is systematically omitted if the Service can only check a single continuous area of memory (i.e. N=1).

10.4 Capability Sets

10.4.1 Minimum Capability Set

All Service Requests and Reports are optional, however the Service will provide at least the capability to load and to dump memory areas using the addressing technique of the memory block(s) it manages. This implies that at least one of the "Load and Dump Memory using Base plus Offsets" or "Load and Dump Memory using Absolute Addresses" sub-services must be provided.

10.4.2 Additional Capability Sets

In addition to a minimum sub-service, the following additional capabilities may be implemented:

Sub-type	Service Request, Report or Capability	Implication(s)
1	Load Memory using Base plus Offsets	→sub-type 3
2	Load Memory using Absolute Addresses	→sub-type 5
3	Dump Memory using Base plus Offsets	→sub-type 1 and sub-type 4
4	Memory Dump using Base plus Offsets Report	→sub-type 3
5	Dump Memory using Absolute Addresses	→sub-type 2 and sub-type 6
6	Memory Dump using Absolute Addresses Report	→sub-type 5
7	Check Memory using Base plus Offsets	→sub-type 8
8	Memory Check using Base plus Offsets Report	→sub-type 7
9	Check Memory using Absolute Addresses	→sub-type 10
10	Memory Check using Absolute Addresses Report	→sub-type 9
	Support of symbolic base references	

Table 10-1 Summary of Memory Management Service Additional Capabilities

11 Task Management Service

11.1 Scope

A commonly encountered function of an onboard Application Process consists of supervising and coordinating the execution of one or more user-controllable software tasks (i.e. individually controllable through ground requests) which have their own thread of execution (i.e. they may execute simultaneously); for example, a task in an ADA program which is controllable from the ground or an active control loop dedicated to the handling of particular events.

All the user-initiated service activities defined in this standard are examples of functions of an Application Process whose execution can be requested from the ground. In practice, one or several software tasks will be directly or indirectly involved in the execution of these functions. Some of these tasks may be user controllable while others may not.

The Task Management Service provides standard service requests for managing the execution of the user-controllable tasks managed by an Application Process.

Any number of onboard Application Processes may provide a single instance of a Task Management Service.

Telemetry source packets and telecommand packets relating to the Task Management Service are denoted by **Service Type = 7**.

11.2 Service Concept

Each user-controllable task managed by an Application Process is uniquely identified by a "Task ID".

If identical tasks are managed by different Application Processes then they should have the same Task ID (for example, a single shared task managed by more than one Application Process or multiple instances of a task each with an independent context and managed by a different Application Process).

A task has an execution status which indicates whether it is currently "running", "suspended" or "stopped"; these states and the possible transitions between them are shown in Figure 11-1 below. The execution status of a task may be downlinked as part of the normal housekeeping data of the Application Process.

This Standard does not specify what the task execution context should be.

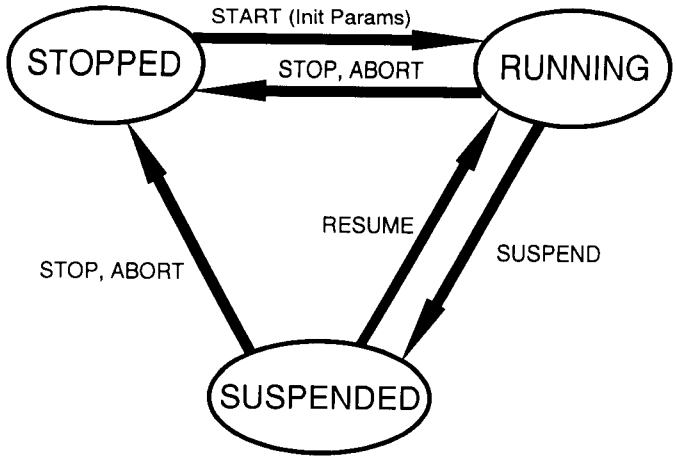


Figure 11-1 State Transition Diagram for a Software Task

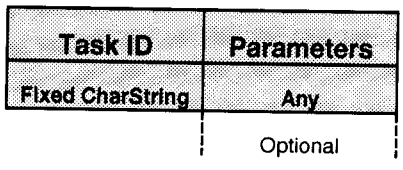
11.3 Service Requests and Reports

Currently no Service Reports are defined.

11.3.1 Starting a Task

The request is:

Start Task, (7,1)
Telecommand Packet, Application Data:



Task ID:

The Task ID, together with the Application ID in the packet header, implicitly defines the presence of, and the structure of, the Parameters field which follows. The length of the character string may be defined on a mission basis or per Application Process.

The internal structure of the Task ID is beyond the scope of this standard.

If identical tasks are managed by two or more Application Processes then their start requests must have an identical Parameters data structure.

Parameters:

A data structure compliant with the Structure Rules Set 1. The parameters are used to configure the specific instance of the execution of the task.

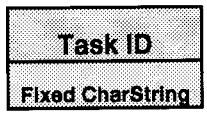
When this request is received, the onboard Application Process starts the specified software task using (or passing to the task) the specified task initiation parameters. The task status is then 'running'.

The request is ignored if the status of the task was "running" or "suspended".

11.3.2 Stopping a Task

The request is:

Stop Task, (7,2)
Telecommand Packet, Application Data:



When this request is received, the onboard Application Process stops the specified task (or asks the task to stop itself). The task status is then "stopped".

The request is ignored if the task status was "stopped".

11.3.3 Suspending a Task

The request is:

Suspend Task, (7,3)
Telecommand Packet, Application Data:
The same as for the "Stop a Task" Request.

When this request is received, the onboard Application Process suspends the specified task (or asks the task to suspend itself). The task status is then "suspended" and the task context is frozen.

The request is ignored if the status of the task was "stopped" or "suspended".

11.3.4 Resuming a Task

The request is:

Resume Task, (7,4)
Telecommand Packet, Application Data:
The same as for the "Stop a Task" Request.

When this request is received, the onboard Application Process resumes the specified task (or asks the task to resume itself). The task status is then "running". The initial task context is the one that was frozen when the task was suspended.

The request is ignored if the status of the task was "stopped" or "running".

11.3.5 Aborting a Task

The request is:

Abort Task, (7,5)
Telecommand Packet, Application Data:
The same as for the "Stop a Task" Request.

When this request is received, the onboard Application Process aborts the specified task.

Stopping a task may not always succeed (e.g. if the task is in an infinite software loop) whilst aborting a task always succeeds (it is an abrupt software interruption).

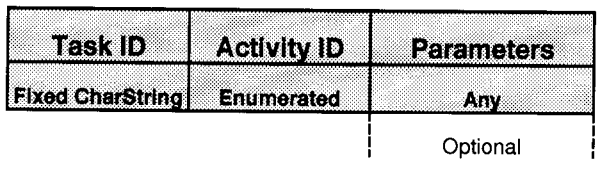
Stopping a task always results in the task context being put into a state which is appropriate for a subsequent run, whilst aborting a task may result in an inconsistent task context (e.g. the task program counter may be left at a random position).

The request is ignored if the task status was "stopped".

11.3.6 Performing an Activity within a Task

The request is:

Perform Activity within Task, (7,6)
Telecommand Packet, Application Data:



Activity ID:
This indicates which activity within the specified task is to be performed. Thus the Activity ID, together with the task ID and the application ID in the packet header, implicitly defines the presence of, and the structure of, the Parameters field which follows.

If identical logical tasks exists within two or more Application Processes then they should have the same set of activities.

Parameters:
Parameters relating to the activity to be performed. This is a data structure compliant with the Structure Rules Set 1.

When this request is received, the onboard Application Process indicates to the task which activity it has to perform and the parameters to be used. The task status is unchanged by the execution of the activity.

The request is ignored if the task status was "stopped" or "suspended".

11.4 Capability Sets

11.4.1 Minimum Capability Set

All service requests are optional. However, the Service will comprise, as a minimum, an opposing pair of task controls. This is shown in Table 11-1 below.

11.4.2 Additional Capability Sets

In addition to a minimum capability set, the following additional capabilities may be implemented:

Sub-type	Service Request, Report or Capability	Implication(s)
1	Start Task	→sub-type 2 and/or sub-type 5
2	Stop Task	→sub-type 1
3	Suspend Task	→sub-type 4
4	Resume Task	→sub-type 3
5	Abort Task	→sub-type 1
6	Perform Activity within Task	

Table 11-1 Summary of Task Management Service Additional Capabilities

12 Function Management Service

12.1 Scope

It is not always meaningful to "package" a set of possibly loosely related functions performed by an Application Processes in a Service. Yet it is normal engineering practice to organise these Application Process capabilities in well-defined functions (called "**Application Functions**" in the remainder of this chapter).

Some application functions may be permanently active (e.g. a mission critical control loop) while others may activated and de-activated from the ground (e.g. an application function to control and monitor payload equipment).

There may be a direct mapping between an application function and a mode of operation. That is to say, the activation of an application function may place a subsystem, payload or unit into a well-defined mode of operation.

When an application function is active, the ground may have the possibility to perform a number of activities in the context of the application function. For example, in a particular mode of operation of a payload, certain control actions may be available.

The Function Management Service provides standard Service requests for controlling the execution of the application functions and of their activities.

Any number of onboard Application Processes may provide a single instance of a Function Management Service.

Telemetry source packets and telecommand packets relating to the Function Management Service are denoted by **Service Type = 8**.

12.2 Service Concept

Each application function of an Application Process is uniquely identified by a "Function ID".

If identical application functions exist in different Application Processes then they should have the same Function ID.

An application function has an execution status which indicates whether it is currently "active" or "inactive". The execution status of an application function whose execution is not user controllable is always "active".

An application function may include a set of activities whose execution can be requested from the ground. Each such activity is uniquely identified by an "Activity ID".

This Standard does not specify what the application function execution context should be.

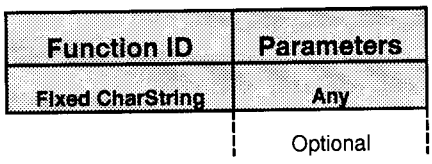
12.3 Service Requests and Reports

There are currently no Service Reports defined.

12.3.1 Activating a Function

The request is:

Activate Function, (8,1)
Telecommand Packet, Application Data:



Function ID:
The Function ID, together with the Application ID in the packet header, implicitly defines the presence of, and the structure of, the Parameters field which follows. 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.

If identical application functions exist in two or more Application Processes then their activation requests must have an identical parameters data structure.

Parameters:
A data structure compliant with Structure Rules Set 1. The parameters are used to configure the specific instance of execution of the application function.

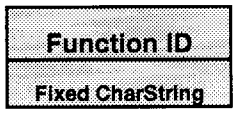
When this request is received, the specified application function is activated by using (or passing) the specified activation parameters. The application function status is then 'active'.

The request is ignored if the status of the application function was "active".

12.3.2 De-activating a Function

The request is:

De-activate Function, (8,2)
Telecommand Packet, Application Data:



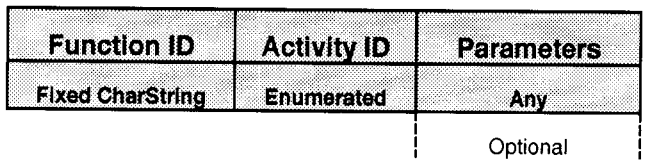
When this request is received, the specified application function is de-activated. The application function status is then "inactive".

The request is ignored if the application function has the "inactive" status.

12.3.3 Performing an Activity of a Function

The request is:

Perform Activity of Function, (8,3)
Telecommand Packet, Application Data:



Activity ID:

This indicates which activity of the specified application function is to be performed. Thus the activity ID, together with the function ID and the Application Process ID in the packet header, implicitly defines the presence of, and the structure of, the Parameters field which follows.

If identical application functions exists within two or more Application Processes then they should have the same set of activities.

Parameters:

Parameters relating to the activity to be performed. This is a data structure compliant with Structure Rules Set 1.

When this request is received, the Application Process indicates to the application function which activity it must perform and the parameters to be used. The application function status is unchanged by the execution of the activity.

The request is ignored if the application function has the "inactive" status.

12.4 Capability Sets

12.4.1 Minimum Capability Set

All service requests are optional. Indeed, an Application Process may have no application function or alternatively no application function with controllable Activities. Consequently, a Service implementation is only required to support either the pair of requests to control application functions or the request to control activities of application functions.

12.4.2 Additional Capability Sets

In addition to a minimum capability set, one of the following additional capabilities may be implemented:

ID	Service Request, Report or Capability	Implication(s)
1	Activate Function	→sub-type 2
2	De-activate Function	→sub-type 1
3	Perform Activity of Function	

Table 12-1 Summary of Function Management Service Additional Capabilities

13 Time Management Service

13.1 Scope

As described in Chapter 14, all satellites are required to regularly generate and downlink the satellite time reference in Time Reports for ground time correlation.

The Time Management Service provides the capability for the management of the time reference sampling and Time Report generation activities, in particular, the control of the rate of generation of the Time Reports. The Service may be required when a mission has varying requirements for the ground time correlation.

This issue of the standard does not cover the following aspects:

- the exchange(s) of time information between the satellite and the ground segment when, for example, drift or leap-second correction is needed because onboard Application Processes use CDS time (i.e. the satellite time / UTC time correlation established by the ground, as indicated in **[AD1]**);

- the exchange(s) of time information between the satellite and the ground segment which may be required when a GPS (Global Positioning System) time correlation is used onboard.

The specification of these time synchronisation aspects will be elaborated in the next issue of this standard.

Any number of onboard Application Processes may provide a single instance of the Time Management Service.

Telemetry source packets and telecommand packets relating to the Time Management Service are denoted by **Service Type = 9**.

13.2 Service Concept

A means of communicating the Time Report generation rate to the Time Reporting Service exists onboard.

When the Time Reporting Service is informed by the Time Management Service that a new generation rate is to be used, it uses this new generation rate from the next telemetry transfer frame of Virtual Channel 0 for which:

virtual channel frame count modulo new generation rate = 0.

Until then, it continues to use the existing generation rate.

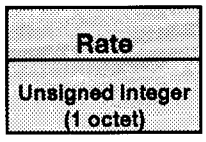
13.3 Service Requests and Reports

There are currently no Service Reports defined.

The request for changing the rate of generation of the time report is:

Change Time Report Generation Rate, (9,1)

Telecommand Packet, Application Data:



Rate:

This parameter determines the generation rate used to sample and downlink the satellite time. Its value must be in the range 0 to 8 inclusive, as defined in **[AD1]**. The corresponding generation rate is equal to once every 2^{Rate} telemetry transfer frames.

When this request is received, the Time Management Service informs the Time Reporting Service that a new time report generation rate is to be used.

13.4 Capabilities Set

13.4.1 Minimum Capability Set

The minimum capability set consist of the following Service Request:

Sub-type	Service Request, Report or Capability
1	Change Time Report Generation Rate

Table 13-1 Summary of Time Management Minimum Capabilities

13.4.2 Additional Capability Sets

There are no additional capability sets.

14 Time Reporting Service

14.1 Scope

As described in **[AD1]**, all satellites are required to maintain a satellite time reference which may be downlinked (via the Spacecraft Time Source Packet generated by means of the "Time Report" Service Report defined in this chapter) so that the ground segment is able to perform a correlation between satellite time and UTC (Universal Time Coordinated).

The Time Reporting Service provides the capability for the generation of Time Reports, such that the satellite time correlation procedures described in **[AD1]** can be performed.

Only one onboard Application Process (i.e. the Time Reporting Application Process with Application Process ID = 0) may provide a single instance of the Time Reporting Service.

Telemetry source packets relating to the Time Reporting Service are denoted by **Service Type = 10**.

14.2 Service Concept

The Service has access to the satellite time reference which is a free running counter. It also maintains the generation rate (1, 2, 4, 8, 16, 32, 64, 128 or 256) of the time report.

As defined in **[AD1]**, the Service samples the satellite time reference simultaneously with the occurrence of the leading edge of the first bit of the synchronisation marker of the telemetry transfer frame of Virtual Channel 0 for each frame for which:

virtual channel frame count modulo generation rate = 0.

The Service then downlinks this satellite time reference in a Spacecraft Time Source Packet at any time before the satellite time reference is next sampled.

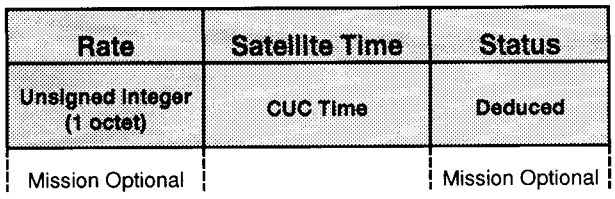
14.3 Service Requests and Reports

This Service has no User-initiated Service Requests.

The Time Report is the only provider-initiated Service Report defined for the Time Reporting Service.

As described in **[AD1]**, the Spacecraft Time Source Packet containing the Time Report has no Data Field Header.

Time Report, (10,1)
Telemetry Source Packet, Source Data:



Rate:
The length of the Rate field is as specified in **[AD1]**. Its value is related to the generation rate by the following formula: **generation rate =** 2^{Rate}.

The Rate field is 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 is the CCSDS Unsegmented Code (CUC) format with implicit or explicit P-Field. If the P-field is implicit, its value is given by the mission parameter <MISSION_TIME_CODE>.

Status:
This gives the status of the Time Reporting Service. The type and possible values of the Status parameter are not specified in this Standard. The type and physical format are deduced from other information (see Chapter 23).

The Status field may be systematically omitted if a mission implements a basic Time Reporting Service.

14.4 Capabilities Set

14.4.1 Minimum Capability Set

The minimum capability set consist of the following Service Report:

Sub-type	Service Request, Report or Capability
1	Time Report

Table 14-1 Summary of Time Reporting Minimum Capabilities

Note that transmission of the Spacecraft Time Source Packet to the ground is enabled and disabled using the Packet Transmission Control Service (see Chapter 18).

14.4.2 Additional Capability Sets

There are no additional capability sets.

15 Onboard Scheduling Service

15.1 Scope

The Onboard Scheduling Service provides the capability to command onboard Application Processes using telecommands pre-loaded onboard the satellite and released at their due time. To achieve this, the Service maintains an onboard command schedule and ensures the timely execution of telecommands contained therein.

Any number of onboard Application Processes may provide a single instance of an OonboardScheduling Service.

Telemetry source packets and telecommand packets relating to the Onboard Scheduling Service are denoted by **Service Type = 11**.

15.2 Service Concept

The Onboard Scheduling Service maintains a Command Schedule which contains telecommand packets and their associated scheduling information.

The Service User(s) can request the following activities:

- enable the scheduling of all, or a subset of, the telecommands in the Command Schedule (e.g. belonging to specified sub-schedules, to be sent to specified Application Processes, etc.);
- disable the scheduling of all, or a subset of, the telecommands in the Command Schedule;
- add telecommands to the Command Schedule;
- 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, etc.);
- time shift all, or a subset of, the telecommands in the Command Schedule or;
- report on all, or a subset of, the telecommands in the Command Schedule.

It shall not be possible for inconsistencies to exist as a direct result of the processing of a Service User request. Thus, the Onboard Scheduling Service will refuse to perform a request in its entirety if this would create an inconsistency in the Command Schedule. The Service User(s) is (are) responsible for ensuring that inconsistencies which can only be detected during the scheduling activity will never occur.

15.2.1 The Command Schedule

The Onboard Scheduling Service maintains a Command Schedule consisting of telecommand packets together with their scheduling attributes. The scheduling attributes of a telecommand indicate:

- the sub-schedule ID with which the telecommand is associated;
- the number of the interlock to be set by this telecommand, if any. The success or failure of execution of the telecommand will be associated with the specified interlock. Interlocking is restricted to commands belonging to the same sub-schedule;
- the number of the interlock on which the release of this telecommand is dependent, if any;
- whether the release of this telecommand is dependent on the success or on the failure of the telecommand to which it is interlocked;
- either the absolute onboard time at which the telecommand packet is to be released to its destination Application Process or a relative time (if supported).

Absolute times and relative times are always expressed in CUC format for the OonboardScheduling Service.

Relative time is defined with respect to a "Scheduling Event" which may be the starting of the Command Schedule, the starting of the sub-schedule containing the telecommand or the setting of the interlock to which the telecommand is interlocked (i.e. when the Scheduling Service is notified of the success or failure of the telecommand which sets the interlock).

The Scheduling Event for relative time may also be the occurrence of a mission specific event. However, this capability is beyond the scope of this Standard.

The Onboard Scheduling Service must 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 will be used for the derivation of the release times of the subsequent telecommands.

15.2.2 Telecommand Release Status

The Onboard Scheduling Service maintains appropriate information to determine whether a telecommand should be released or not at its due time.

The **release status** of a telecommand is 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 is either ***"disabled"*** or ***"enabled"***.

The release status of a telecommand is "enabled" if and only if the release of telecommands:

- from the Command Schedule and
- from the sub-schedule to which the telecommand belongs and
- to the destination Application Process of the telecommand

are enabled. The release status is "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 Application Process level) whose values determine the release status of the telecommand in accordance with Table 15-1 below.

Schedule	Sub-schedule	Application Process	Release 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

Table 15-1 Decision Table for the Release Status of a Telecommand

15.2.3 Telecommand Interlock Status

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

The **execution result** of a telecommand may be ***"success"*** or ***"failure"***.

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

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

The Scheduling Service explicitly indicates to the destination Application Process that the released telecommand sets an interlock so that, when the telecommand execution is complete, an execution report is passed back to the Scheduling Service.

On reception of this report, the Scheduling Service sets 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 is 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 is "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 is "locked".

This is illustrated in Figure 15-1 below (in which "1+" means "release depends on the successful execution of the telecommand setting interlock number 1" and "1-" means the opposite).

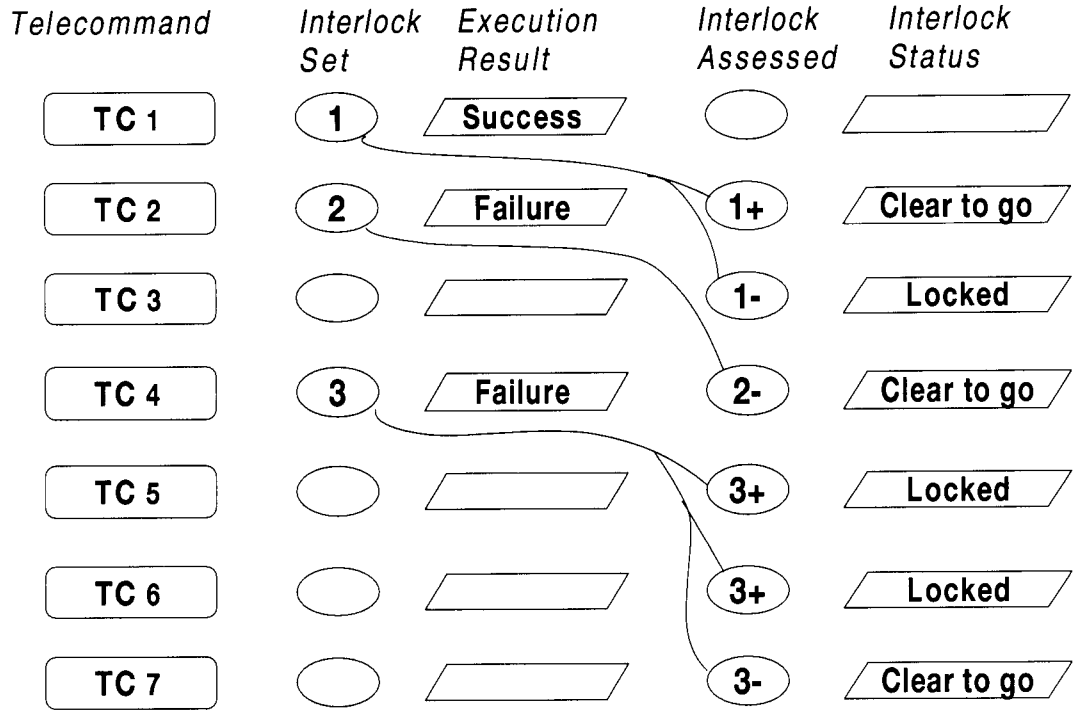


Figure 15-1 The Relation between Execution Result and Interlock Status

15.2.4 Scheduling Events

The Onboard Scheduling Service determines the times of occurrence of the various Scheduling Events used as the basis for relative release times, that is:

- the onboard CUC time at which the Command Schedule was enabled. It is unknown if the Command Schedule has not yet been enabled;

- the onboard 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. It is unknown if the sub-schedule is disabled;

- the onboard 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. It is unknown if the telecommand execution result is not yet known.

Note that 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 Onboard Scheduling Service will refuse to add or time-shift telecommands if this would result 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).

15.2.5 Auxiliary Information

The Onboard Scheduling Service also has access to other information required for the proper execution of its activities, for example:

- the maximum number of entries or maximum size of the Command Schedule;
- the maximum number of sub-schedules which can be simultaneously managed;
- the maximum number of interlocks which can be simultaneously managed;
- the list of sources from which the Service can receive telecommand packets to be scheduled;
- the list of onboard Application Processes to which the Service can release telecommand packets.

The Service uses this information for error detection and reporting.

15.2.6 The Scheduling Activity

The processing of a telecommand packet whose release time is due is always performed (e.g. even if the Command Schedule is disabled).

The corresponding service activity is:

- the telecommand is not 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 is the current time. If the telecommand sets an interlock, the execution result is set to "Failure";
- otherwise the telecommand is released. Where applicable, its execution result is then determined as indicated in Section 15.2.3 and the time of its Scheduling Event is determined as indicated in Section 15.2.4.

In addition to this primary activity, any onboard CUC time jumps are detected and reported.

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

15.3 Service Requests and Reports

15.3.1 Controlling the Command Schedule

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 (e.g. those telecommands with specified destination Application Processes).

Whilst a Service User may enable or disable telecommands originating from different sources, in practice there may be restrictions on which source(s) are allowed to do this.

Whilst the Service provides for the enabling, disabling or resetting of the Command Schedule as a whole, in practice there may be restrictions on which source(s) are allowed to exercise this control.

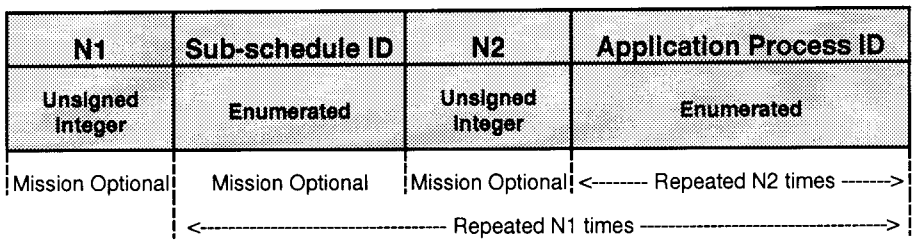
If an error is detected during the processing of a request, it does not affect the processing of the remainder of the request.

15.3.1.1 Controlling the Release of Telecommands

The Service Requests to enable or disable the release of selected telecommands are:

Enable Release of Selected Telecommands, (11,1)
Telecommand Packet, Application Data:

Disable Release of Selected Telecommands, (11,2)
Telecommand Packet, Application Data:



N1: This field is 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 is also omitted and the application data is simply an array of N2 Application Process IDs.

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

This field is systematically omitted if the Service does not support the concept of sub-schedules.

N2: This field is 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/disabled.

When the Service Provider receives this request, then:

- if **N1 = 0**, the schedule level controlling attribute of all telecommands (see Section 15.2.2) is set according to the request type. If it is an enable request, then the Command Schedule Scheduling Event time is also set;
- if **N1 > 0** and **N2 = 0**, the sub-schedule level controlling attribute of the telecommands from the specified sub-schedules is set according to the request type. If it is an enable request, then the Scheduling Event times of the sub-schedules are also 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 is set according to the request type.

Note that if N1 > 1 then there may be a mixture of empty arrays (N2 = 0) and non-empty arrays (N2 > 0).

15.3.1.2 Resetting the Command Schedule

The request is:

Reset Command Schedule, (11,3)
Telecommand Packet, Application Data: **None**

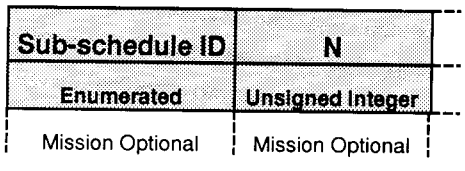
When the Service Provider receives this request:

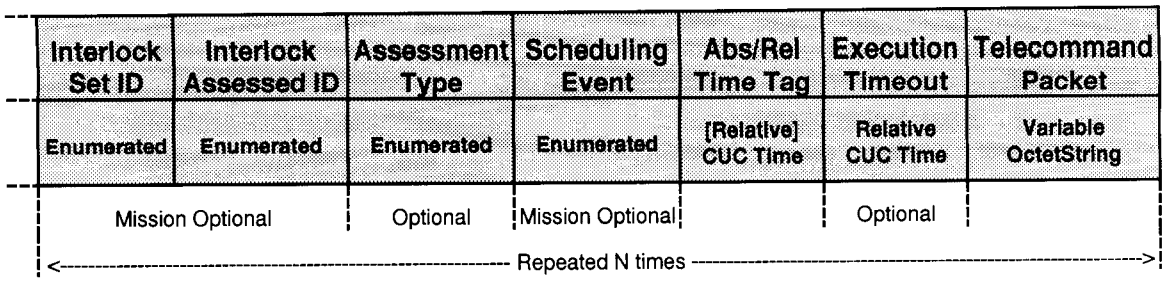
- it clears all entries in the Command Schedule. The Command Schedule is disabled, all sub-schedules are disabled and all Application Processes are enabled;
- it resets the interlock information (no interlock defined);
- it resets the Scheduling Event information (all Scheduling Event times are unknown).

15.3.2 Inserting Telecommands in the Command Schedule

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

Insert Telecommands in Command Schedule, (11,4)
Telecommand Packet, Application Data:





Sub-schedule ID:

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

This field is 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 is 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 to be set). The status of this interlock will be determined by the success or failure of the telecommand execution.

This field is 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 to be assessed).

This field is 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 to 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 is not present if the telecommand is not interlock dependent (or if the Interlock Assessed ID field is not present).

Scheduling Event:

This determines whether the release time of this telecommand is an absolute oonboardCUC 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 is relative to the time at which the Command Schedule is enabled.

If the Scheduling Event = "Sub-Schedule" (value = 2), the telecommand release time is 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 is relative to the time of notification to the Service of the success or failure of the telecommand which sets the interlock.

The Scheduling Event may take other mission-specific values.

This field is 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 onboard CUC time at which the telecommand packet is to be sent to its Application Process ID. The format and length of this field are uniquely defined for the onboard Application Process which provides the Service and are 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 is an onboard positive delta time which, when added to the time of occurrence of the event identified by the Scheduling Event, determines the absolute time at which the telecommand packet is to be sent to its destination Application Process. The format and length are the same as for the absolute time tag and are uniquely defined for the onboard Application Process which provides the Service.

Execution Timeout:

This is an onboard positive delta time which, when added to the time of release of the telecommand, determines the latest time at which the telecommand is expected to complete execution. This parameter is only present if the telecommand has a relative release time and sets an interlock.

If an execution completion report is not received by the Service within the timeout window, then the telecommand is deemed to have failed. Also, if the telecommand defines a Scheduling Event, then the Scheduling Event time is set to the timeout window upper bound and the absolute times of the related relative time telecommands become known (they would otherwise remain indefinitely in the Command Schedule).

The format and length of this field are the same as for the absolute time tag defined for the onboard Application Process which provides the Service.

Telecommand Packet:

This is a standard telecommand packet of any Service type and sub-type. It should be noted that this can be a telecommand destined for an onboard scheduling Service, either this one or a Service implemented in another Application Process, e.g. a request to enable or disable sub-schedules within this Command Schedule.

The source of the telecommand packet is indicated in the Source ID field of the packet header. This may be the ground system or another onboard Application Process.

If a telecommand to be added has a relative time with respect to an interlock, it is "linked" to the latest telecommand added to the Command Schedule which sets that interlock.

When this request is received, each telecommand in the request is processed in turn and, if no error is detected during its processing, it is added to the Command Schedule.

An error occurs, for example, if the Command Schedule is full, if the sub-schedule ID or one of the interlock IDs exceeds its maximum value, if the interlock IDs are equal, if the destination Application Process of the telecommand is not in the set of those allowed, if the time specification refers to the past, if the time specification is not supported by the Service, if the telecommand is interlock dependent and its release time falls within the execution window of its interlocking telecommand or if the telecommand has an "Interlock" relative time and no telecommand added since the last resetting of the Command Schedule sets the interlock.

15.3.3 Deleting Telecommands from the Command Schedule

There are several ways of deleting a subset of the telecommands from the Command Schedule. The subsets are specified according to selection criteria (e.g. those telecommands whose absolute schedule time falls within a specified time period).

The Scheduling Service will 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.

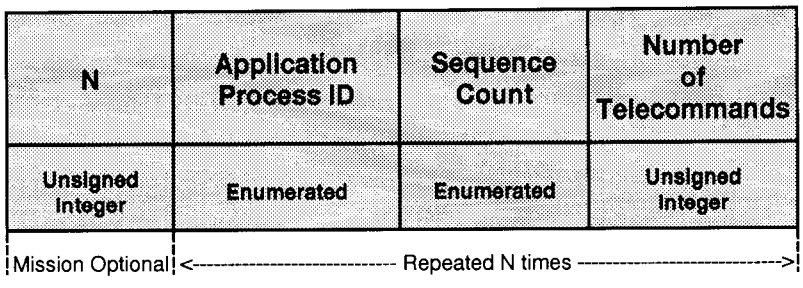
Whilst a Service User may delete telecommands originating from different sources, in practice there may be restrictions on which source(s) are allowed to do this.

If an error is detected during the processing of a request, nothing is deleted.

15.3.3.1 Deleting Telecommands

The request to delete sets of telecommands from the Command Schedule is:

Delete Telecommands, (11,5)
Telecommand Packet, Application Data:



N: This field is systematically omitted if the Service does not support the concept of "scatter delete" (i.e. N=1).

Sequence Count:
The identification of the first telecommand packet to be sent to the specified destination Application Process which is to be deleted.

The doublet (Application Process ID, Sequence Count) uniquely identifies a telecommand packet. These parameters correspond to the packet header fields of each telecommand packet as defined in Section 4.4.

Number of Telecommands:
The number of successive telecommand packets sent by the source to the specified destination Application Process which are to be deleted.

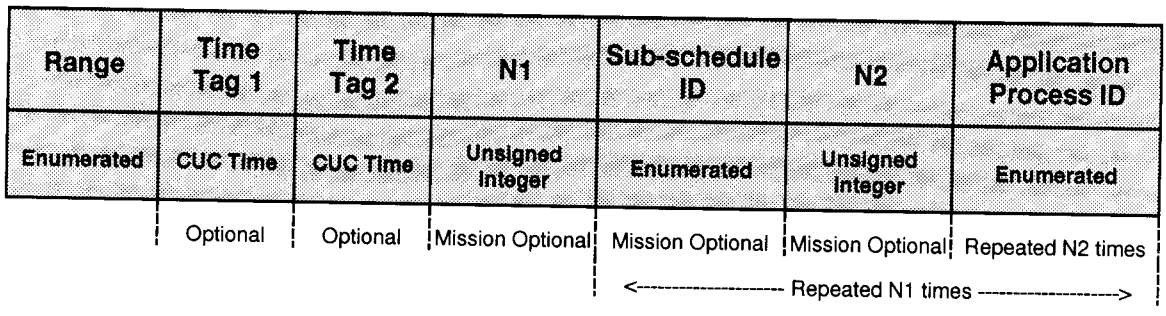
When this request is received, all telecommands which satisfy the selection criteria defined by the Application Process ID, Sequence Count and the Number of Telecommands are deleted. An error occurs if the first telecommand to be deleted is not found in the Command Schedule.

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.

15.3.3.2 Deleting Telecommands over a Time Period

The request is:

Delete Telecommands over Time Period, (11,6)
Telecommand Packet, Application Data:



Range:
This indicates whether 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
- less than or equal to Time Tag 1 if Range is ***"Before"*** (value = 2) or
- 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 is not present if Range is ***"All"***.

Time Tag 2:
The latest absolute time if Range is ***"Between"***. This parameter is not present if Range is not ***"Between"***.

N1: This field is 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 is also omitted.

Sub-schedule ID:
The identification of the sub-schedule(s) from which telecommands are to be deleted. By convention, the value 0 for Sub-schedule ID means "all sub-schedules".

N2: This field is systematically omitted if the Service does not support the selective deletion of telecommands in a time range (which is equivalent to having N2=0).

Application Process ID:
The identification of the destination Application Process(es) from which telecommands are to be deleted.

When this request is received, the following telecommands are deleted 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 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.

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*** is evaluated under the assumption that the execution duration of ***telecommand A*** is zero.

15.3.4 Time-Shifting of Telecommands in the Command Schedule

There are several ways of time-shifting a subset of the telecommands in the Command Schedule. The time-shift request contains the time offset to be added (which may be a positive or negative value) and specifies the group of telecommands to which this time-offset is to be applied.

The Scheduling Service will 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.

Whilst a Service User may time-shift telecommands originating from different sources, in practice there may be restrictions on which source(s) are allowed to do this.

If an error is detected during the processing of a request, nothing is time-shifted.

15.3.4.1 Time-Shifting Telecommands

The request to time-shift sets of telecommands in the Command Schedule is:

Time-Shift Telecommands, (11,7)
Telecommand Packet, Application Data:

Time Offset	N	Application Process ID	Sequence Count	Number of Telecommands
Relative CUC Time	Unsigned Integer	Enumerated	Enumerated	Unsigned Integer

Mission Optional <---------- Repeated N times ---------->

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).

N: This field is 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 are modified (by adding the specified time offset) for those telecommands which meet the selection criteria defined by the specified Application Process ID, Sequence Count and Number of Telecommands. An error occurs if the first telecommand to be time-shifted is not found in the Command Schedule.

In the case of a telecommand with relative release time, it is the relative time which is modified if its Scheduling Event has not yet occurred.

15.3.4.2 Time-Shifting Telecommands over a Time Period

The request is:

Time-Shift Telecommands over Time Period, (11,8)
Telecommand Packet, Application Data:

Range	Time Tag 1	Time Tag 2	Time Offset
Enumerated	CUC Time	CUC Time	Relative CUC Time

Optional Optional

N1	Sub-schedule ID	N2	Application Process ID
Unsigned Integer	Enumerated	Unsigned Integer	Enumerated

Mission Optional Mission Optional Mission Optional Repeated N2 times

<---------- Repeated N1 times ---------->

Range:
The Service may only support particular values of Range.

N1: This field is 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 is also omitted.

N2: This field is 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 means "all sub-schedules".

When this request is received, the release times of the following telecommands are 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.

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*** is evaluated under the assumption that the execution duration of ***telecommand A*** is zero.

15.3.5 Reporting of the Command Schedule Contents

There are several ways of reporting a subset of the telecommands in the Command Schedule. The requests indicate whether a summary or detailed report is to be produced.

The reports contain information on telecommands originating from all sources. The information is ordered according to the predicted times of telecommand release.

Only those telecommands are reported for which the time of release has not yet expired.

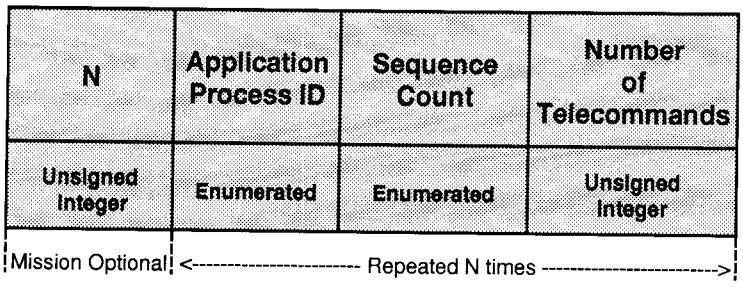
Note that telecommands may be released between the reception of the Service Request and the completion of the Service Report. However, the report contains a consistent view which reflects the situation at the report packet time.

15.3.5.1 Detailed Reporting of the Command Schedule

The request to obtain a detailed report on sets of telecommands in the Command Schedule is:

Report Command Schedule in Detailed Form, (11,9)

Telecommand Packet, Application Data:



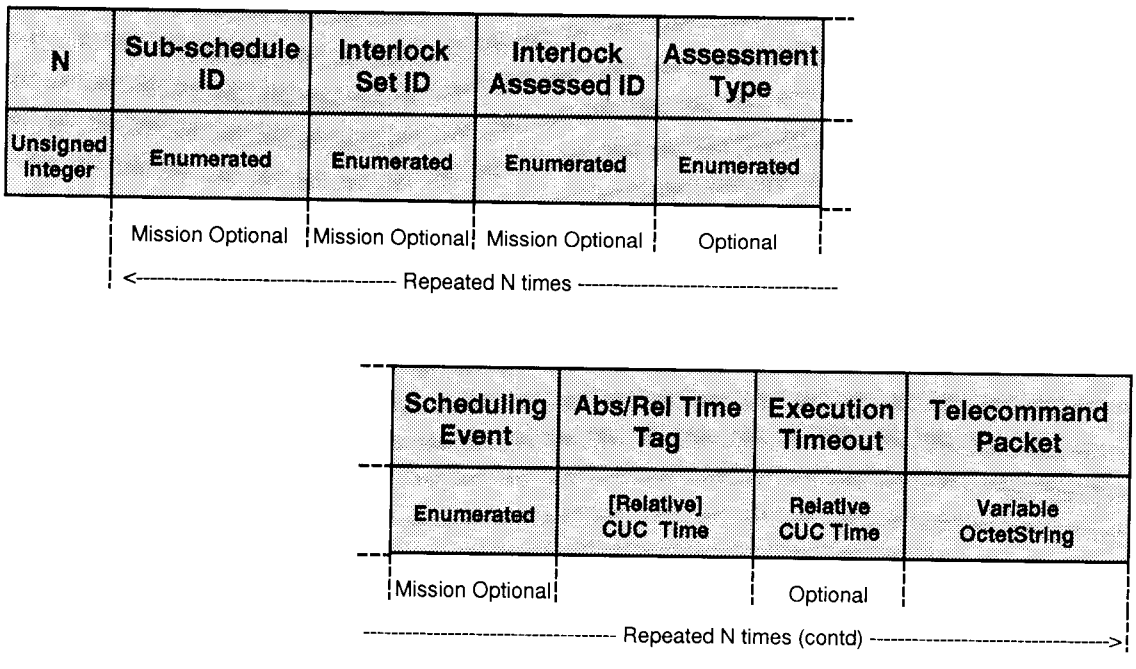
N: This field is systematically omitted if the Service does not support the concept of "scatter report" (i.e. N=1).

When this request is received, a Report is generated containing those telecommands in the Command Schedule which meet the selection criteria defined by the combination of Application Process ID, Sequence Count and Number of Telecommands. An error occurs if the first telecommand to be reported is not found in the Command Schedule.

The Report contains 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:



Sub-schedule ID:

This field is systematically omitted if the Service does not support the concept of sub-schedules.

Interlock Set ID, Interlock Assessed ID:
These fields are systematically omitted if the Service does not support the concept of interlocking.

Scheduling Event:
This field is systematically omitted if the Service does not support the concept of relative time.

Relative time is placed in the report where 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).

15.3.5.2 Summary Reporting of the Command Schedule

The request to obtain a summary report of sets of telecommands in the Command Schedule is:

Report Command Schedule in Summary Form, (11,12)

Telecommand Packet, Application Data:

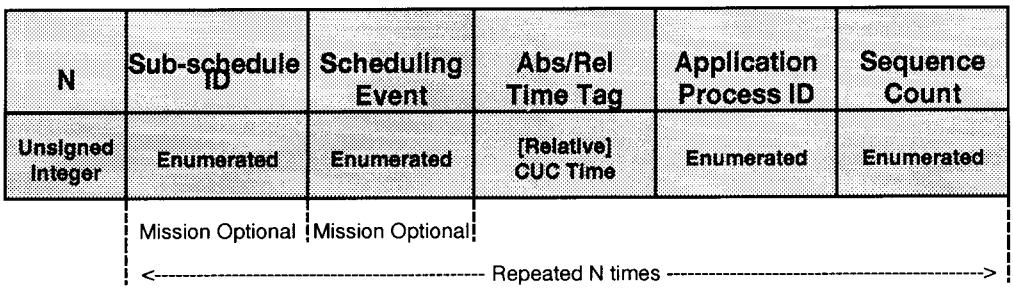
Same as for the "Report Command Schedule in Detailed Form" Service Request

When this request is received, a Report is generated containing those telecommands in the Command Schedule which meet the selection criteria defined by the combination of Application Process ID, Sequence Count and Number of Telecommands. An error occurs if the first telecommand to be reported is not found in the Command Schedule.

The Report contains 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:



15.3.5.3 Detailed Reporting of the Command Schedule over a Time Period

The request for a detailed report of selected part(s) of the Command Schedule over an absolute time period is:

Report Command Schedule in Detailed Form over Time Period, (11,11)
Telecommand Packet, Application Data:

Range	Time Tag 1	Time Tag 2	N1	Sub-schedule ID	N2	Application Process ID
Enumerated	CUC Time	CUC Time	Unsigned Integer	Enumerated	Unsigned Integer	Enumerated
	Optional	Optional	Mission Optional	Mission Optional	Mission Optional	Repeated N2 times
				<---------- Repeated N1 times ---------->		

Range:
The Service may only support particular values of Range.

N1: This field is 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 is also omitted.

N2: This field is 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 Section 15.3.5.1) is generated. The report covers 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 are not 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.

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*** is evaluated under the assumption that the execution duration of ***telecommand A*** is zero.

15.3.5.4 Summary Reporting of the Command Schedule over a Time Period

The request for a summary report of selected part(s) of the Command Schedule over an absolute time period is:

Report 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 section 15.3.5.2) is generated. The report covers the telecommands which meet the same selection criteria as defined in section 15.3.5.3.

15.4 Capability Sets

15.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Requests and Reports, but some of them only with particular combinations of parameters as specified in the sections above:

Sub-type	Service Request or Report
1	Enable Release of Telecommands
2	Disable Release of Telecommands
3	Reset Command Schedule
4	Insert Telecommands in Command Schedule

Table 15-2 Summary of Onboard Scheduling Minimum Capabilities

15.4.2 Additional Capability Sets

An Onboard Scheduling Service may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
	Support of Sub-Schedules	N1 in requests and reports
	Support of one or more types of relative times	Scheduling Event in requests and reports
	Support of the interlocking command concept	Interlock related fields in requests and reports
	Support of "scatter" delete, time-shift and report (for selected requests)	N in selected requests
	Support of telecommand selection strategy at sub-schedule level (for selected requests)	N1 in selected requests
	Support of telecommand selection strategy at Application Process level (for selected requests)	N2 in selected requests
5	Delete Telecommands	
6	Delete Telecommands over Time Period	
7	Time-Shift Telecommands	
8	Time-Shift Telecommands over Time Period	
9	Report Command Schedule in Detailed Form	→sub-type 10
10	Detailed Schedule Report	→sub-type 9 or sub-type 11
11	Report Command Schedule in Detailed Form over Time Period	→sub-type 10
12	Report Command Schedule in Summary Form	→sub-type 13
13	Summary Schedule Report	→sub-type 12 or sub-type 14
14	Report Command Schedule in Summary Form over Time Period	→sub-type 13

Table 15-3 Summary of Onboard Scheduling Additional Capabilities

16 Onboard Monitoring Service

16.1 Scope

The Onboard Monitoring Service provides the capability to monitor onboard parameters with respect to checks defined by the ground and reports any check transitions to the service user. To achieve this, the Service maintains a monitoring list and checks parameter samples according to the information contained therein.

Any number of onboard Application Processes may provide a single instance of an OonboardMonitoring Service.

Telemetry source packets and telecommand packets relating to the Onboard Monitoring Service are denoted by **Service Type = 12**.

16.2 Service Concept

A Monitoring List is maintained which contains the parameter monitoring information, drives the parameter monitoring activity and the generation of Out-of-Limit Reports.

The ground segment can modify or report the contents of the Monitoring List using Service Requests to:

- reset the Monitoring List;
- add parameters to, or delete parameters from, the Monitoring List;
- modify the monitoring information of parameters in the Monitoring List;
- enable or disable the monitoring of parameters in the Monitoring List;
- report the monitoring information for all parameters in the Monitoring List;
- report the set of parameters which are currently out-of-limits.

The ground can also modify attributes of the Onboard Monitoring Service which determine:

- whether the monitoring of parameters is enabled or disabled at Service level;
- the maximum reporting delay for the Out-of-limit Report. An Out-of-limit 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 Out-of-limit 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 datated).

16.2.1 The Monitoring List

The Onboard Monitoring Service maintains static monitoring information for each parameter to be monitored, which is provided by the ground by means of Service Requests. The **parameter monitoring information** specifies:

- the identification of the onboard parameter to be monitored;
- whether the monitoring of the parameter is enabled or disabled;
- the associated validity parameter (if any); this is a Boolean parameter whose value determines whether the parameter is to be monitored;
- the monitoring interval for the parameter, expressed in units of <DIAG_MIN_INTERV>.

The parameter monitoring information also includes a set of **check definitions**. A check definition provides the information required 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 type of check definition may be associated with a given parameter.

Thus, a check definition indicates:

- the nature of the check to be performed. This can be a Limit-check, a Delta-check or an Expected-value-check. A Delta-check can only be performed on parameters which can also be limit checked.

  For a Limit-check, a low-limit value and a high-limit value are specified. For a Delta-check, a minimum-delta value and a maximum-delta value are specified. For an Expected-value-check, an expected value is specified.

- a check selection parameter, which is a Boolean onboard parameter whose value determines whether the check against the limit pair (or expected value or delta threshold pair) is applied. If no selection parameter is provided, the check is always applied.

- a "filter". For a limit-check or an expected-value-check, this indicates the number of successive samples of the parameter which must fail (or succeed) the check in order to establish a new **checking status** for the parameter.

  For a delta-check, it is the number of consecutive delta values to be used to evaluate an average delta value and it is this average delta value which is checked against the delta check definition. A delta value is the arithmetic difference between successive samples of a parameter.

16.2.2 The Checking Activity and the Check State

The Onboard Monitoring Service maintains a **check state** corresponding to each check definition for each parameter to be monitored.

The check state includes information about the previous and current checking statuses of the parameter for the given check definition (see check filter above) and the time at which the transition to that checking status occurred. This information is downlinked when the ground requests a report of the parameters which are currently out-of-limit.

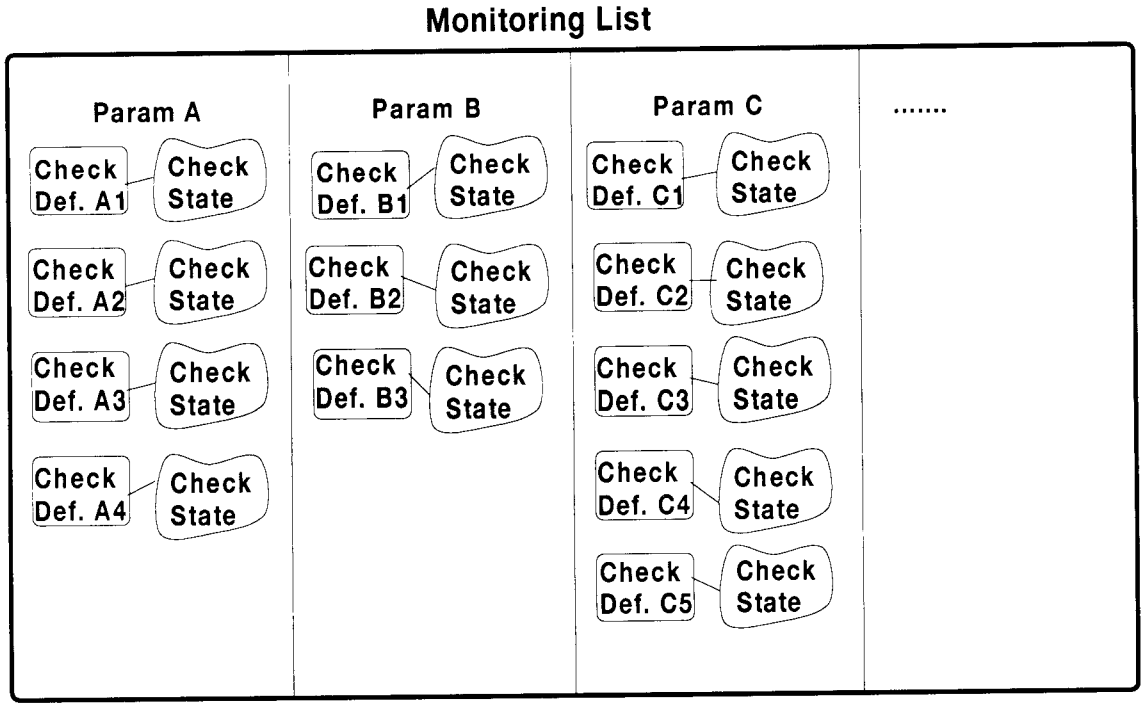


Figure 16-1 Parameter Check Definitions and Check States

A check definition is ***"enabled"*** and used for checking a parameter when:

- the monitoring of parameters is enabled at Service level **and**
- the monitoring of the parameter is enabled **and**
- 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 is ***"disabled"*** and is not used for checking the parameter.

Whenever a sample of the parameter is available for checking, the Service performs the following checking activity independently for each parameter check definition (and updates its check state accordingly):

- if the check definition is "disabled" then the new checking status immediately becomes either ***"Un-checked"***, ***"Invalid"*** or ***"Un-selected"*** 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 checking status of a parameter with respect to the check definition is "Un-checked" when the parameter is added to the monitoring list or if a new check definition for the parameter is added at a later time.

- if the check definition is "enabled" then the parameter sample is a valid sample for checking. It is checked against the limit pair or expected value, or it is used to elaborate a new average delta value which is 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 <filter> successive valid samples of the parameter (including the current one) have consistently failed (or consistently passed) the check, then the parameter is assigned a new checking status. The new checking status is 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"***.

  Note that if the previously determined checking status of a parameter with respect to a limit-check was "Within limits" then if successive samples are alternately "Below low limit" and "Above high limit", these are considered as "consistent" for the purposes of assigning a new checking status. However, if the last known checking status was, for example, "Above high limit" then a sequence of consecutive "Below low limit" samples is needed before a new checking status is assigned.

  For a delta-check, once <filter> successive valid samples of the parameter have been accumulated, a new mean delta value is evaluated and checked. The new checking status is set to the result of the check which is either ***"Below low threshold"***, ***"Above high threshold"*** or ***"Within thresholds"***.

- having elaborated a new checking status for the parameter, a comparison between the previous and new checking statuses is performed. If they differ, then a check transition is recorded (conceptually this is recorded in a Transition Reporting List, see below).

- when a check transition is detected, the transition time is recorded in the corresponding check state. This is the sampling time of the first parameter sample which was used to establish the new checking status.

The current checking statuses and associated transition times may be reported to the ground on request.

16.2.3 The Transition Reporting List

During the course of the monitoring activity, an ordered list of checking status transitions is established. Within this list, there may be more than one checking status transition for a given parameter (e.g. transition relating to different check definitions; transitions corresponding to different samples of the parameter etc.).

Each checking status transition in the list is characterised 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 is downlinked via an Out-of-Limit Report no later than the maximum reporting delay after the time of the first transition in the list. The list is emptied after downlink.

16.2.4 Auxiliary Information

It is assumed that the Onboard Monitoring Service has access to other information required for the detection of errors in the processing of service requests, for example:

- 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 allowable type(s) of check for each parameter which can be monitored (delta-check, limit-check or status check);
- the list of Boolean onboard parameters which can be accessed by the Application Process and can thus be used as validity parameter or check definition selection parameters.

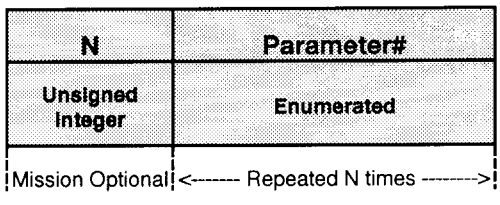
16.3 Service Requests and Reports

16.3.1 Controlling the Onboard Monitoring

It is possible to enable or disable the monitoring of parameters globally or to enable or disable the monitoring of a specified subset of parameters. The requests are:

Enable Monitoring of Parameters, (12,1)
Telecommand Packet, Application Data:

Disable Monitoring of Parameters, (12,2)
Telecommand Packet, Application Data:



N: The number of parameters whose monitoring is to be enabled/disabled. By convention, N = 0 means "Enable/disable monitoring at Service level".

This field is 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:

- if **N = 0**, it sets the Service-level monitoring status to "Enabled" or "Disabled", depending on the request sub-type.

  If "enable" is requested, the parameters in the Monitoring List whose parameter level monitoring status is "Enabled" start being monitored.

  If "disable" is requested, none of the parameters in the Monitoring List is any more monitored. Also, the checking status for all check transitions in the Monitoring List is set to "Un-checked" and their times of transition are set to the current time. The current content of the Transition Reporting List is not affected by these activities. If the list is not empty, its content is reported as usual via an Out-of-Limit Report.

- if **N > 0**, each parameter in the request is processed in turn and its parameter level monitoring status is set to "Enabled" or "Disabled", depending on the request sub-type.

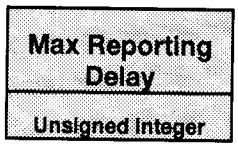
  If the monitoring of parameters is enabled at Service level and "enable" is requested, the monitoring of the parameters specified in the request starts immediately after the processing of the request.

  An error is flagged if the parameter is not in the list. However, the processing of the remaining parameters is not affected.

16.3.2 Changing the Maximum Reporting Delay

The request is:

Change Maximum Reporting Delay, (12,3)
Telecommand Packet, Application Data:



Max Reporting Delay:
The maximum reporting delay for the Out-of-limit Report, expressed in units of <DIAG_MIN_INTERV>.

When the Service Provider receives this request, the maximum reporting delay is recorded and used to determine when to downlink the Transition Reporting List.

16.3.3 Clearing the Monitoring List

The request is:

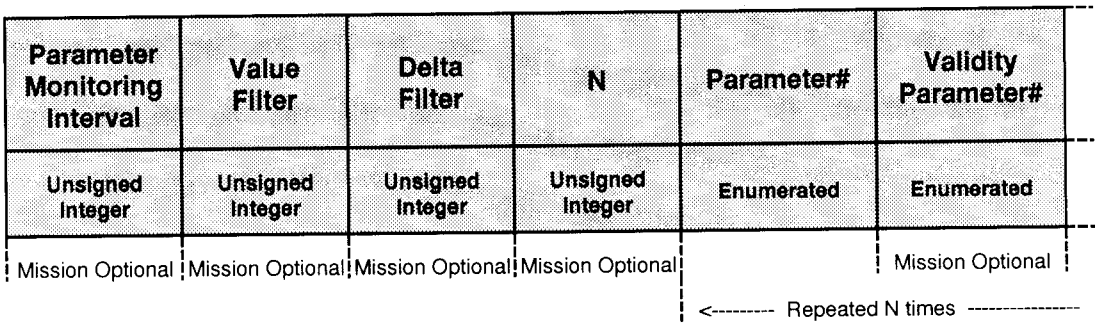
Clear Monitoring List, (12,4)
Telecommand Packet, Application Data: **None**

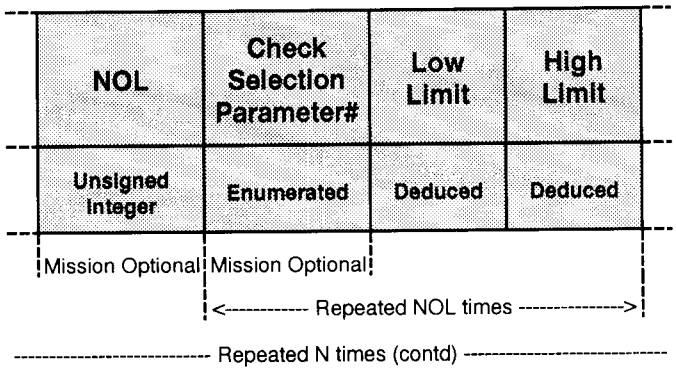
When the Service Provider receives this request, it sets the Service monitoring status to "Disabled" and clears all entries in the Monitoring List and in the Transition Reporting List.

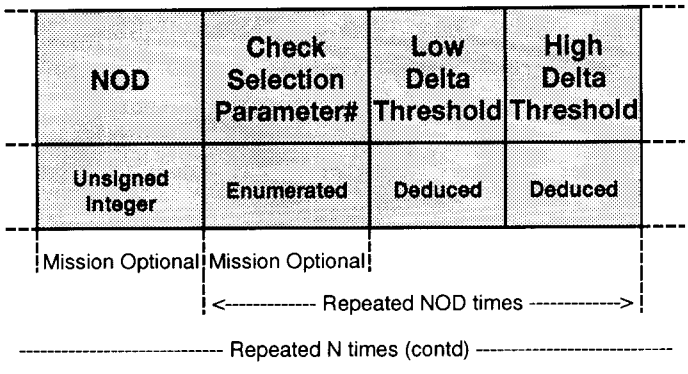
16.3.4 Adding Parameters to the Monitoring List

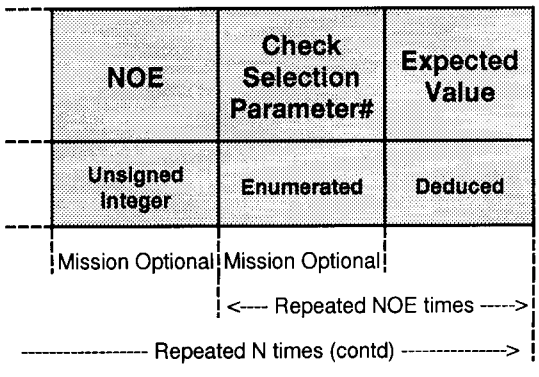
The request is:

Add Parameters to Monitoring List, (12,5)
Telecommand Packet, Application Data:









Parameter Monitoring Interval:
The monitoring interval for the parameters, in units of <DIAG_MIN_INTERV>.

This field is systematically omitted if the Service knows which value to use.

Value Filter:
The number of successive samples of the parameters needed to establish a new checking status for an Expected-value-check or a Limit-check.

This field is systematically omitted if the Service knows which value to use or if limit and expected value checks are not supported by the Service.

Delta Filter:
The number of successive samples of the parameters needed to establish a new checking status for a Delta-check.

This field is systematically omitted if the Service knows which value to use or if delta checks are not supported by the Service.

N:
The number of parameters to be added to the Monitoring List.

This field is 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 is always checked).

This field is 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 are systematically omitted if limit checks, delta checks and/or expected-value checks are not supported by the Service (i.e. NOL=0, NOD=0 and/or NOE=0 and the corresponding array is empty). The Service must 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 is always applied.

This field is systematically omitted, for a type of check, if the Service only supports one check definition of this type per parameter.

The type and format of the Low Limit, High Limit, Low Delta Threshold, High Delta Threshold or Expected Value are the same as the type and format of the values of the parameter to be monitored.

The number of parameters to be added to the monitoring list is limited to a maximum of <MONLIST_MAX_PARAMS>.

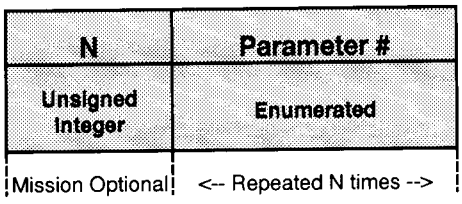
When the Service Provider receives this request, it adds the parameter monitoring information to the Monitoring List, initialises the check state of the check definitions and sets the parameter monitoring status to "Disabled".

If an error is detected during the processing of the monitoring information for a given parameter, this parameter is not added to the Monitoring List. This does not affect the processing of the remaining parameters. An error occurs, for example, if the Monitoring List is full, if the parameter is already in the list, if the parameter is not accessible, if the validity parameter or check selection parameter is not accessible or is not Boolean.

16.3.5 Deleting Parameters from the Monitoring List

The request to delete specified parameters from the monitoring list is:

Delete Parameters from Monitoring List, (12,6)
Telecommand Packet, Application Data:



N: This field is 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 processes each parameter in turn and removes its corresponding monitoring information, if any, from the Monitoring List (the entry becomes free).

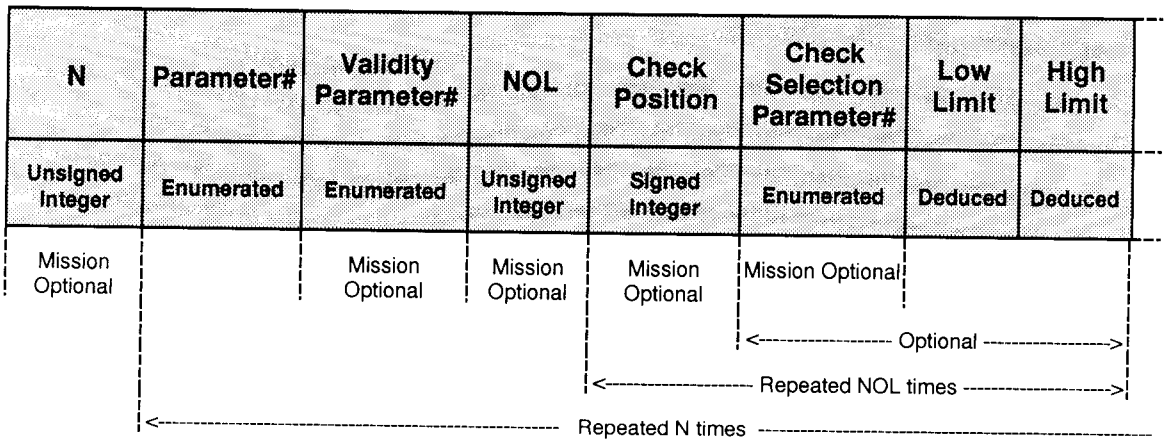
A standard error occurs if the parameter is not in the Monitoring List. This does not affect the deletion of the parameters which have an entry in the Monitoring List.

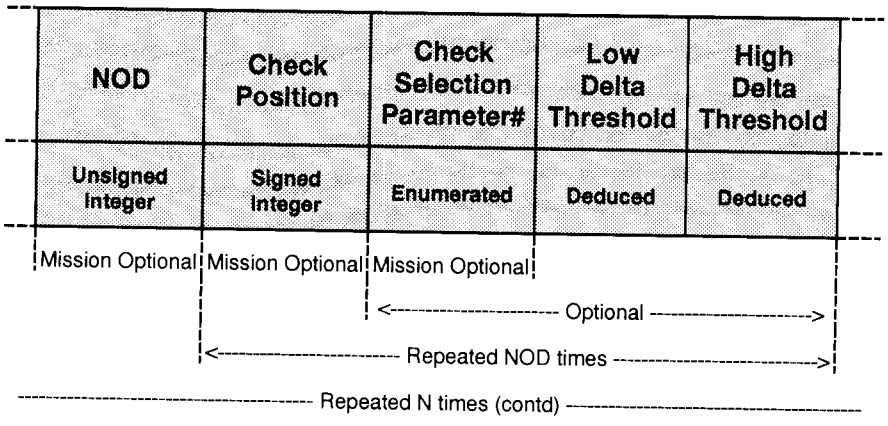
16.3.6 Modifying the Parameter Checking Information

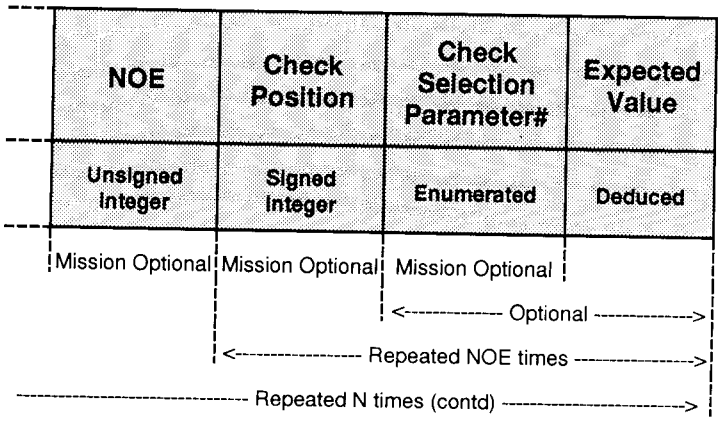
The request to modify the checking information for specified parameters is:

Modify Parameter Checking Information, (12,7)

Telecommand Packet, Application Data:







N:

This field is systematically omitted if the Service only supports modification of the monitoring information for one parameter at a time.

Validity Parameter #:

This field is systematically omitted if the Service does not support the concept of parameter validity.

NOL, NOD, NOE:
These fields are systematically omitted if limit checks, delta checks and/or expected value checks are not supported by the Service (i.e. NOL=0, NOD=0 and/or NOE=0 and the corresponding array is empty).

Check Position:
This indicates which check definition (for the given parameter) has to be deleted, added or replaced with a new check definition.

A positive Check Position value indicates that the corresponding check definition for the parameter is to be replaced by the new check definition which follows.

A negative Check Position value indicates that the corresponding check definition is to be deleted (the positions of succeeding check definitions are decremented). No check definition follows in this case. Any Service will refuse to delete the last remaining check definition of a parameter and will 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 is systematically omitted, for a 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 is systematically omitted, for a type of check, if the Service only supports one check definition of this type per parameter.

The number of parameters whose entry in the monitoring list is to be modified is limited to a maximum of <MONLIST_MAX_PARAMS>.

When the Service Provider receives this request, it processes the checking information for each parameter in turn and (if no error is detected) replaces, adds or deletes the specified check definitions. The check state for a new or modified check definition is initialised.

A standard error occurs 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 is unchanged if any error is detected but the processing of the remaining parameters is unaffected.

16.3.7 Reporting the Current Monitoring List Contents

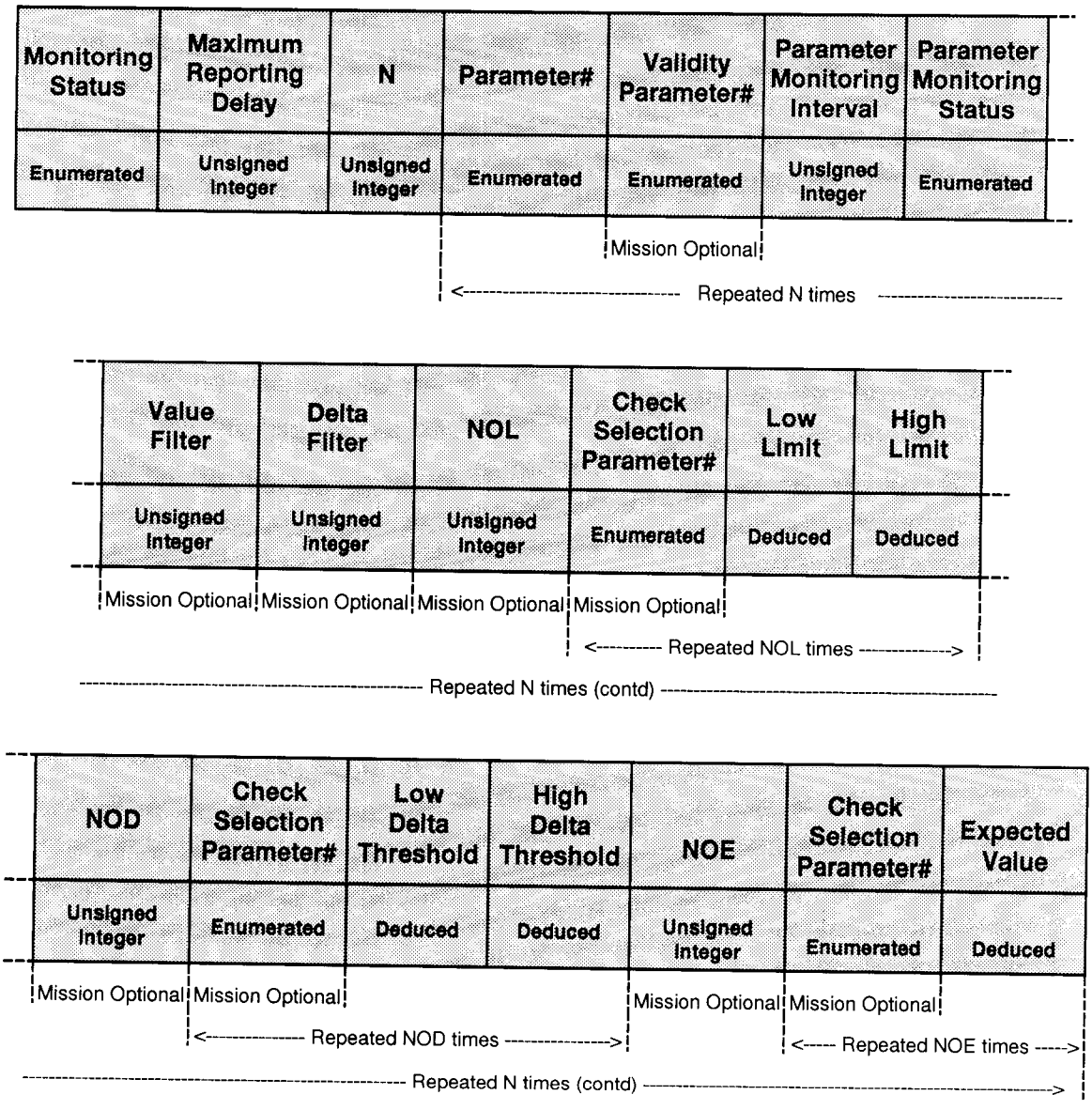
The request is:

Report Current Monitoring List, (12,8)
Telecommand Packet, Application Data: **None**

When the Service Provider receives this request, it issues a report with the current static contents of the Monitoring List.

Current Monitoring List Report, (12,9)

Telemetry Source Packet, Source Data:



Monitoring Status:

This indicates whether the overall monitoring is "enabled" (value = 1) or "disabled" (value = 0).

Validity Parameter #:

This field is 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 Filter:

This field is systematically omitted if expected value and limit checks are not supported by the Service.

Delta Filter:
This field is systematically omitted if delta checks are not supported by the Service.

NOL, NOD, NOE:
These fields are systematically omitted if limit checks, delta checks and/or expected-value checks are not supported by the Service (i.e. NOL=0, NOD=0 and/or NOE=0 and the corresponding array is empty).

Check Selection Parameter #:
This field is systematically omitted, for a type of check, if the Service only supports one check definition of this type per parameter.

16.3.8 Reporting the Current Parameters Out-of-limit List

The request is:

Report Current Parameters Out-of-limit List, (12,10)
Telecommand Packet, Application Data: **None**

When the Service Provider receives this request, it issues 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 "Un-expected Value". Thus a parameter may 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	Time

<---------- Repeated N times ---------->

Parameter value:
This gives the value of the telemetry parameter at the time the last checking status transition was detected. The format and length are uniquely defined for the parameter.

Limit Crossed:
This is the value of the Low Limit, High Limit, Low Delta Threshold, High Delta Threshold or Expected Value which has been crossed/violated. It has 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 overleaf.

Current Checking Status:
This indicates the current checking status of the parameter. The possible values are defined overleaf.

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 are:

"Expected Value", "Within Limits", "Within Thresholds":	value = 0
"Un-checked":	value = 1
"Invalid":	value = 2
"Un-Selected":	value = 3
"Un-expected 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 are (U->UV, I->UV, US->UV, EV->UV), the mnemonics appearing in the transition types mean:

EV:	Expected Value	US:	Un-Selected
U:	Un-checked	UV:	Un-expected Value
I:	Invalid		

For a limit check, the only reported transition types are (U->BL, U->AL, I->BL, I->AL, US->BL, US->AL, WL->BL, WL->AL, BL->AL, AL->BL), the mnemonics appearing in the transition types mean:

WL:	Within Limits	US:	Un-Selected
U:	Un-checked	BL:	Below Low Limit
I:	Invalid	AL:	Above High Limit

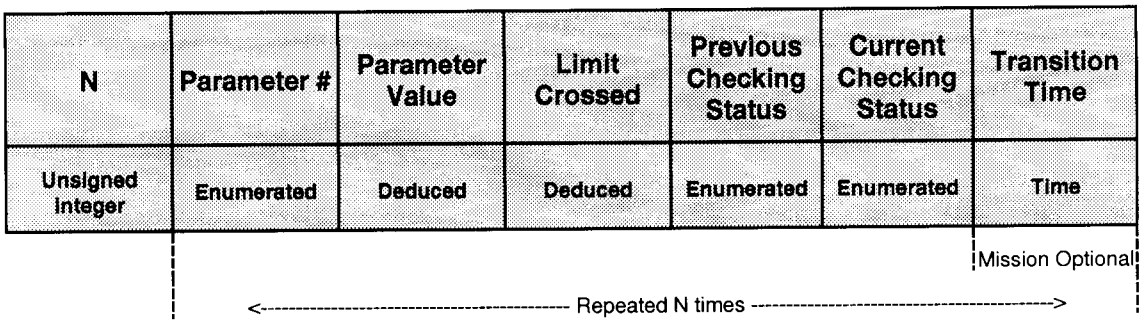
For a delta check, the only reported transition types are (U->BT, U->AT, I->BT, I->AT, US->BT, US->AT, WT->BT, WT->AT, BT->AT, AT->BT), the mnemonics appearing in the transition types mean:

WT:	Within Thresholds	US:	Un-Selected
U:	Un-checked	BT:	Below Low Threshold
I:	Invalid	AT:	Above High Threshold

16.3.9 Reporting the Out-of-Limit Transitions

The Out-of-limit Report is the only Provider-Initiated Report and contains the contents of the Transition Reporting List established since the last time an Out-of-limit Report was issued. It consists of:

Out-of-limit Report, (12,12)
Telemetry Source Packet, Source Data:



For an expected value check, the Previous Checking Status and Current Checking Status field values can be any of the following values (but must be different):

Un-checked, Invalid, Un-Selected, Expected Value or Un-expected Value

For a limit check, the Previous Checking Status and Current Checking Status field values can be any of the following values (but must be different):

Un-checked, Invalid, Un-Selected, 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 must be different):

Un-checked, Invalid, Un-Selected, Within Thresholds, Below Low Threshold or Above High Threshold

The values used for encoding the Previous Checking Status and Current Checking Status fields are as defined for the Current Parameter Out-of-limit List Report in the previous section (e.g. value = 0 for "Expected Value", "Within Limits" and "Within Thresholds").

The same Parameter# may appear several times in a Report.

The Transition Time field is systematically omitted if the maximum reporting delay supported by the Service is always smaller than the required precision with which the transition time is to be known (e.g. it may only be needed if the maximum reporting delay is large compared with the parameter monitoring intervals for some parameters).

The Out-of-limit Report is 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 is emptied immediately after the Report has been generated.

16.4 Capability Sets

16.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Requests and Reports:

Sub-type	Service Request, Report or Capability
1	Enable Monitoring of Parameters
2	Disable Monitoring of Parameters
12	Out-of-limit Report

Table 16-1 Summary of Onboard Monitoring Service Minimum Capabilities

16.4.2 Additional Capability Sets

The Onboard Monitoring Service may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
	Support of limit, expected value or delta checks (minimum one type to be supported)	NOL, NOE or NOD and corresponding array in requests and reports
	Support of validity parameter concept	Corresponding field in requests and reports
	Support of several check definitions of a type of check	NOL, NOE or NOD and corresponding array in requests and reports
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	→sub-type 9
9	Current Monitoring List Report	→sub-type 8
10	Report Current Parameters Out-of-limit List	→sub-type 11
11	Current Parameters Out-of-limit List Report	→sub-type 10

Table 16-2 Summary of Onboard Monitoring Service Additional Capabilities

17 Large Data Transfer Service

17.1 Scope

The Large Data Transfer Service is a supporting service used by the ground or other Services to transfer large **service data units** in a controlled manner. The choice as to whether or not to use the Large Data Transfer Service is made by the initiator of a given service data unit.

Some scenarios where the Large Data Transfer Service may be used are:

i) the Onboard Scheduling Service may be requested to report to the ground the current contents of the Command Schedule, whose size may exceed the maximum size of a single telemetry source packet;

ii) a very large area of onboard memory may have to be loaded using telecommand packets with a mission-specific maximum size.

The Large Data Transfer Service provides a common transfer mechanism for all Services thus avoiding the proliferation of service-specific solutions. A service data unit is passed to the Large Data Transfer Service which splits it into parts (as needed) and transmits each part within a single telemetry source packet (onboard to ground) or a single telecommand packet (ground to onboard).

The Large Data Transfer Service may have to fulfil different operational requirements for the data transfer. These requirements include:

- the capability to determine when the transfer is complete;
- the capability to identify and selectively re-transmit those parts of the data which are lost during a transfer;
- the capability to pass the data parts in order, and without duplication, to the recipient.

Any number of Application Processes may provide a single instance of a Large Data Transfer Service.

Telemetry source packet and telecommand packet relating to the Large Data Transfer Service are denoted by **Service Type = 13**.

17.2 Service Concept

17.2.1 Introduction

The Large Data Transfer Service provides 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** onboard 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 onboard.

In the remainder of this chapter, 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 is as follows: when a service data unit or portion of it is passed to the sending sub-service of the Large Data Transfer Service, it performs the downlink (uplink) in the following manner:

- it splits the original Service Data Unit (SDU) in fixed size parts and sends the parts one by one, and in order. Each part is transmitted within a single telemetry source packet (telecommand packet) as illustrated in Figure 17-1 below:

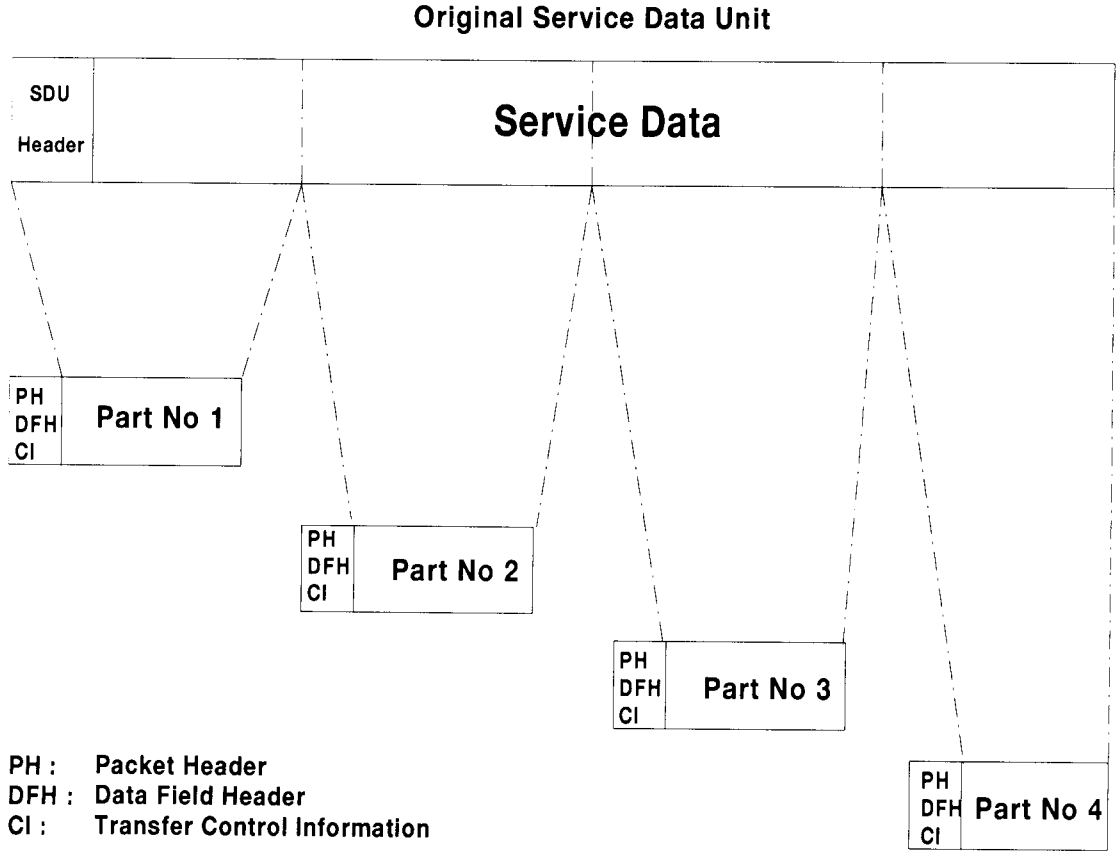


Figure 17-1 The splitting of a Service Data Unit in Parts to be Downlinked (Uplinked)

- the sending and receiving sub-services use the same definition of the size of the parts (***Part Size***). The parts have equal size (except possibly the last part), which must be less than the maximum size of the Source Data field of a telemetry source packet (Application Data field of a telecommand packet).

17.2.2 Service Data Unit to be Downlinked (Uplinked)

For the Large Data Downlink, the service data unit which an onboard Service passes to the sending sub-service must 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 onboard receiving sub-service must consist of either a telecommand packet or an "extended" telecommand packet.

In the remainder of this chapter, 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) is as follows:

Unit Type	Packet Header	Data Field Header	Packet Data
Enumerated	Record	Record	Any

Unit Type:
This parameter indicates whether what follows is a standard packet (value = 0) or an extended packet (value = 1).

An extended packet has the same structure as a packet, but the size of its packet data field may exceed the maximum size defined for a packet (i.e. if the total length of the Packet Data plus the Data Field Header exceeds 65535 octets).

For an extended packet, the value of the Packet Length field in the Packet Header is 0.

Packet Header:
This is a standard packet header, but the interpretation of the packet length field value differs 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 subservice receives a **"Large Data Unit ID"**, which unambiguously identifies the service data unit to be transferred. This Large Data Unit ID is only needed and used by the sub-services of the Large Data Transfer Service onboard 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 onboard Application Process.

The detailed specification of the mechanism used to pass large data units between an oonboardService Provider and a Large Data Transfer Service is beyond the scope of this standard. However, depending on the storage capacity available to the Large Data Transfer Service or on design constraints, the following approaches may be used.

For a downlink operation, the data may be passed by the sending Service to the onboard Large Data Transfer Service either as a complete service data unit or as a set of data blocks (in which case the Large Data Transfer Service informs the sending Service when the next block may be passed or when a downlink problem occurs). The service data unit may even be available in a storage/memory area directly accessible by the Large Data Transfer Service.

For an uplink operation, the data received by the onboard Large Data Transfer Service may be passed to the destination Service either as a complete data unit or as a set of data blocks or it may even be written directly into a storage/memory area accessible by the destination Service.

17.2.3 Large Data Downlink (Uplink) Protocol

The following downlink (uplink) protocol aspects complement the basic protocol concept defined in section 17.2.1:

- the sending sub-service may have an attribute (***Window Size***) defining the number of parts which can be downlinked prior to receiving an acknowledgement from the receiving sub-service.

  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 by more than the value of "Window Size". In other words, a part can only be sent if it falls within the sliding window defined by the last acknowledged part and the window size.

  Several window sizes may be used by a given Large data Transfer Service. For example, a large window size may 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 sends the parts of the Service Data Unit at the highest frequency allowed under the prevailing operational constraints.

- when a receiving sub-service receives an error-free part, this becomes 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 uplink window will the receiving end send a notification of the "last successfully received part". This allows the sending sub-service to slide the downlink window.

  If a sliding window is not used, the notification of successful reception is only mandatory on completion of the service data unit reception.

  In both situations, this ensures that the sending sub-service can inform the originating Service (the ground) of the success or failure of the downlink (uplink). **Note that only when the destination application on the ground (destination Service onboard) 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.**

- when the sending sub-service has sent the last part which can be sent in the current sliding window or the last part of the service data unit, it initiates an "acknowledgment timer". If it does not receive a notification of "last successfully received part" or an abort notification within a specified timeout interval, then it assumes that the downlink (uplink) has failed and notifies the originating Service (the ground) accordingly.

- when the first, or an intermediate, part (which may be a re-transmitted part) is received by a receiving sub-service, a "reception timer" is initiated. If the receiving sub-service does not receive a subsequent part within a specified timeout interval, then it assumes that the downlink (uplink) has failed and the reception activity is locally aborted. The destination application on the ground (destination Service oonboard is notified accordingly.

- if the receiving sub-service receives an erroneous part or detects a gap in the reception sequence it adds 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 to automatically request the re-transmission of these parts or simply to inform the ground (the destination Service) of the errors in the transferred data.

- 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 will the receiving sub-service send notifications of the failed parts.

  The sending sub-service then re-transmits (though not necessarily immediately) the notified failed parts. The successful retransmission of a failed part updates the set of failed parts and may determine a new "last successfully received part".

  If a sliding window is not used, notification of failed parts is not required. However, if the receiving sub-service does send such a notification, then the sending sub-service is required to retransmit the failed parts.

- at any time during the downlink (uplink) activity, the originating (destination) Service or the ground segment can abort the large data transfer operation.

If the sending sub-service is able to perform several overlapping data transfers, there is 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.

17.3 Service Requests and Reports

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.

17.3.1 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 are:

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
Mission Optional		

Large Data Unit ID:
This uniquely identifies the Service data unit which is the subject of the transfer.

This field is systematically omitted if the Service only supports the downlink (or uplink) of one Service Data Unit at a time. This applies to all requests and reports in this chapter.

Sequence Number:
This is a unique identification for this part of the Service data unit. By convention, the sequence number of the first part is 1. The sequence number is then 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 is always equal to the Part Size in use.

Downlink: The onboard sending sub-service uses the report to send the first part of the data to be downlinked.

Uplink: The ground segment uses the request to send the first part of the data to be uplinked.

On reception of the part, the receiving sub-service updates its context and initiates a reception timer.

17.3.2 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 are:

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 is always equal to the Part Size in use.

Downlink: The sending sub-service uses this report to send an intermediate part of the data to be downlinked.

Uplink: The ground segment uses this request to send an intermediate part of the data to be uplinked.

The sending sub-service initiates 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 updates its context and initiates a reception timer.

17.3.3 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 are:

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 is less than or equal to the Part Size in use.

Downlink: The sending sub-service uses this report to send the last part of the data to be downlinked.

Uplink: The ground segment uses this request to send the last part of the data to be uplinked.

The sending sub-service always initiates an acknowledgement timer.

On reception of the part, the receiving sub-service updates its context and immediately notifies the sending sub-service of what is the "last successfully received part" at this point.

17.3.4 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 retransmission has been requested by the receiving end are:

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 is the same as the Sequence Number used for the initial transfer.

Downlink: The sending sub-service uses this report to retransmit a part of the data which was not properly received by the ground and which was requested for retransmission.

Uplink: The ground segment uses this request to retransmit a part of the data which was not properly received by the onboard receiving sub-service.

The sending sub-service initiates an acknowledgement timer if this part is the last part which can be sent in the current window.

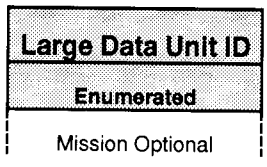
On reception of the part, the receiving sub-service updates its context and initiates a reception timer.

17.3.5 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 are:

Downlink Abort Report, (13,4)
Telemetry Source Packet, Source Data:

Abort Reception of Uplinked Data, (13,13)
Telecommand Packet, Application Data:



Downlink: The sending sub-service uses this report to indicate to the ground that the large data downlink operation has been aborted (e.g. by the originating Service).

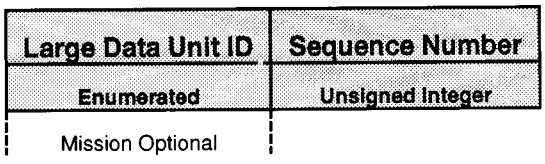
Uplink: The ground segment uses this request to indicate to the onboard receiving sub-service that the large data reception activities should be aborted. The onboard receiving sub-service informs the destination Service of the abort.

17.3.6 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 are:

Uplink Reception Acknowledgement Report, (13,14)
Telemetry Source Packet, Source Data:

Downlink Reception Acknowledgement, (13,5)
Telecommand Packet, Application Data:



The receiving sub-service uses this notification to indicate to the sending sub-service that it has successfully received all parts of the large Service data unit up to and including the part with the indicated sequence number.

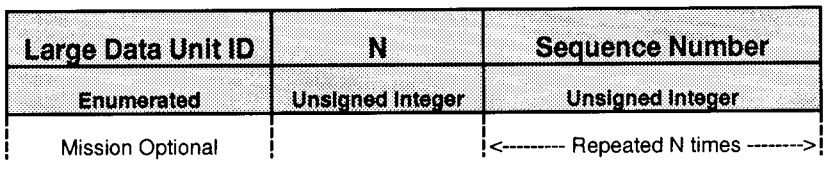
If a sliding window is used, this may enable it to be moved forward and the sending activity to be resumed.

17.3.7 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 that specified parts were not received or were erroneously received are:

Unsuccessfully Received Parts Report, (13,15)
Telemetry Source Packet, Source Data:

Repeat Parts, (13,6)
Telecommand Packet, Application Data:



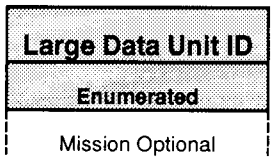
On reception of this notification, the sending sub-service records the information. It uses this information to retransmit the parts if a sliding window is used or if it is an onboard sending sub-service. A ground sending sub-service may decide not to retransmit the parts if a sliding window is not used.

17.3.8 Transfer Abort Initiated by the Receiving End

The report (during an uplink operation) and the request (during a downlink operation) are:

Reception Abort Report, (13,16)
Telemetry Source Packet, Source Data:

Abort Downlink, (13,8)
Telecommand Packet, Application Data:



On reception of this notification, the sending sub-service informs the originating Service (the ground) that the downlink (uplink) operation is to be aborted.

17.4 Capability Sets

17.4.1 Minimum Capability Sets

Each sub-service (sending and receiving) is optional, but a given implementation of the Large Data Transfer Service must support at least one sub-service.

The minimum capability set for an onboard Large Data Transfer Service providing the sending sub-service (Data Downlink operation) consists of the following Service Requests and Reports:

Sub-type	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

Table 17-1 Summary of Sending Sub-service (Downlink) Minimum Capabilities

The minimum capability set for a Large Data Transfer Service providing the receiving sub-service (Data Uplink operation) consists of the following Service Requests and Reports:

Sub-type	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

Table 17-2 Summary of Receiving Sub-service (Uplink) Minimum Capabilities

17.4.2 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.

A Large Data Transfer Service providing the sending sub-service (Data Downlink operation) may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
6	Repeat Parts	→sub-type 7
7	Repeated Part Report	→sub-type 6
	Use of a "Sliding Window"	→sub-type 7

Table 17-3 Summary of Sending Sub-service (Downlink) Additional Capabilities

A Large Data Transfer Service providing the receiving sub-service (Data Uplink operation) may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
12	Accept Repeated Part	→sub-type 15
15	Unsuccessfully Received Parts Report	→sub-type 12
	Use of a "Sliding Window"	→sub-type 12

Table 17-4 Summary of receiving Sub-service (Uplink) Additional Capabilities

18 Packet Transmission Control Service

18.1 Scope

The Packet Transmission Control Service provides the capability to control the transmission to the ground of telemetry source packets issued by other onboard Services. A generic transaction is provided to control the transmission of packets (i.e. Service Reports) selected at the level of Type and/or Sub-type. Other transactions are provided for control of individual housekeeping, diagnostic and report packets. The current transmission status of selected types of packets can also be reported.

Each onboard Application Process provides a single instance of the Packet Transmission Control Service so that the transmission of any type of telemetry source packets originating from it can be controlled.

Telemetry source packets and telecommand packets relating to the Packet Transmission Control Service are denoted by **Service Type = 14**.

18.2 Service Concept

The Packet Transmission Control Service maintains the knowledge of which packets generated by the Services of the Application Process should or should not be transmitted to the ground.

The transmission of packets may be "enabled" and "disabled" at the level of:

- a type of packet;
- a sub-type of packet;
- a housekeeping packet definition, a diagnostic packet definition or a filtered report definition (see Chapters 7 and 9).

The transmission of packets with a given type and sub-type is "enabled" if and only if the packet type and the packet sub-type are enabled (i.e. if the type is not in the set of disabled types and the sub-type is not in the set of disabled sub-types for that type).

In addition, the transmission of housekeeping (or diagnostic or report) packets is "enabled" if and only if the packet type, the packet sub-type and the housekeeping packet definition (or the diagnostic packet definition or the filtered report definition) are enabled.

Conceptually, this is as if each such packet definition has three independent controlling attributes (at type level, at sub-type level and at packet structure definition level) whose values determine the transmission status of the packets in accordance with Table 18-1 overleaf.

Type	Sub-Type	Definition (SID)	Transmission 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

Table 18-1 Decision Table for the Transmission Status of a Packet

The Packet Transmission Control Service maintains the transmission status for all types and sub-types of packet generated by the Services of its Application Process and reports them to the ground on request.

The Packet Transmission Control Service maintains the transmission status for all housekeeping packet definitions, diagnostic packet definitions and filtered report definitions of its Application Process and reports them to the ground on request.

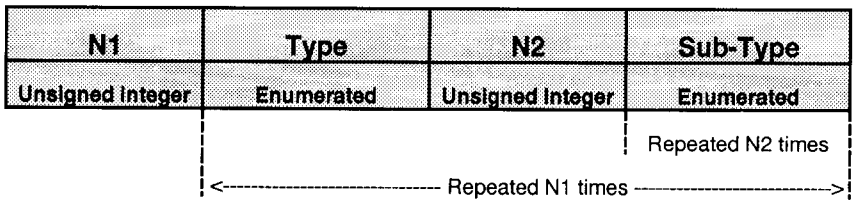
18.3 Service Requests and Reports

18.3.1 Controlling the Transmission of Specified Telemetry Source Packets

The requests to enable or disable the transmission of telemetry source packets of specified type and sub-type from the destination Application Process are:

Enable Transmission of Telemetry Source Packets, (14,1)
Telecommand Packet, Application Data:

Disable Transmission of Telemetry Source Packets, (14,2)
Telecommand Packet, Application data:



Type:
The telemetry source packet Service Type.

Sub-Type:
The telemetry source packet Service Sub-Type for the specified Service Type.

When the Service Provider receives this request:

- if **N1 = 0** and "enable" is requested, all types of telemetry source packets are removed from the set of disabled types and all sets of disabled sub-types are emptied; if "disable" is requested, all types of telemetry source packets are placed in the set of disabled types;

- if **N1 > 0** and **N2 = 0** and "enable" is requested, the specified types of telemetry source packets are removed from the set of disabled types and their sets of disabled sub-types are emptied; if "disable" is requested, the specified types of telemetry source packets are added to the set of disabled types;

- if **N1 > 0** and **N2 > 0** and "enable" is requested, the specified sub-types of telemetry source packets are removed from the set of disabled sub-types for the specified type; if "disable" is requested, the specified sub-types of telemetry source packets are added to the set of disabled sub-types for the specified type.

Note that if N1 > 1 then there may be a mixture of empty (N2 = 0) and non-empty (N2 > 0) arrays.

An error is flagged if a specified type or sub-type of packet is not generated by the Application Process. However, the processing of the other types and sub-types in the request is unaffected.

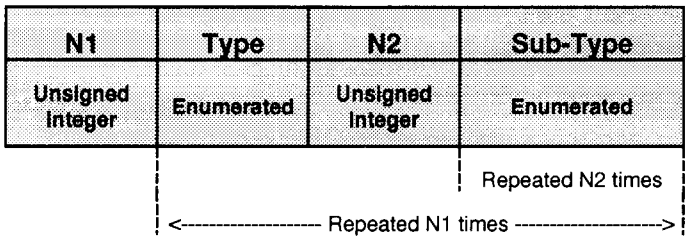
18.3.2 Reporting the List of Active Telemetry Source Packets

The request to report the list of telemetry source packet types and sub-types from the Application Process with an "Enabled" transmission status is:

Report Active Telemetry Source Packets, (14,3)
Telecommand Packet, Application Data: **None**

When this request is received, the active types and sub-types of telemetry source packet from the Application Process are determined and a report is generated.

Active Telemetry Source Packets Report, (14,4)
Telemetry Source Packet, Source Data:



If **N1 = 0**, this means that no type nor sub-type of packet is active.

If **N1 > 0**, this means that the specified types of packet are active.

If **N2 = 0** for a type of packet, this means that none of the sub-types of this type is active.

If **N2 > 0** for a type of packet, this means that the specified sub-types of this type are active.

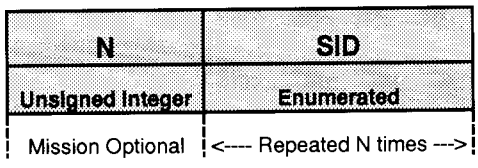
Note that if N1 > 1 then there may be a mixture of empty (N2 = 0) and non-empty (N2 > 0) arrays.

18.3.3 Controlling the Transmission of Specified Housekeeping Packets

The requests are:

Enable Transmission of Housekeeping Packets, (14,5)
Telecommand Packet, Application Data:

Disable Transmission of Housekeeping Packets, (14,6)
Telecommand Packet, Application data:



N: The number of housekeeping packet definitions to be enabled or disabled.

This field is systematically omitted if the Service only supports the control of one housekeeping packet at a time (i.e. N=1).

SID: The Structure Identifier identifying a housekeeping packet definition.

When this request is received, the specified housekeeping packet definitions are removed from (or added to) the set of disabled housekeeping packet definitions (depending on whether it is an "enable" or "disable" request).

An error is flagged if a specified housekeeping SID is not defined for the Application Process. However, this does not affect the processing of the other housekeeping SIDs in the request.

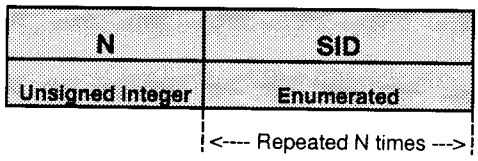
18.3.4 Reporting the List of Active Housekeeping Packets

The request to report the list of housekeeping packet definitions of the Application Process with an "Enabled" transmission status is:

Report Active Housekeeping Packets, (14,7)
Telecommand Packet, Application Data: **None**

When this request is received, the list of active housekeeping packet definitions from the Application Process is determined and a report is generated.

Active Housekeeping Packets Report, (14,8)
Telemetry Source Packet, Source Data:

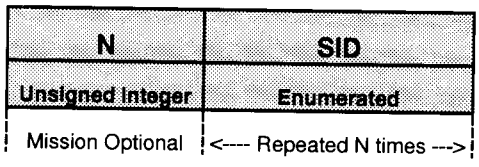


18.3.5 Controlling the Transmission of Specified Diagnostic Packets

The requests are:

Enable Transmission of Diagnostic Packets, (14,9)
Telecommand Packet, Application Data:

Disable Transmission of Diagnostic Packets, (14,10)
Telecommand Packet, Application data:



N: The number of diagnostic packet definitions to be enabled or disabled.

This field is systematically omitted if the Service only supports the control of one diagnostic packet at a time (i.e. N=1).

SID: The Structure Identifier identifying a diagnostic packet definition.

When this request is received, the specified diagnostic packet definitions are removed from (or added to) the set of disabled diagnostic packet definitions (depending on whether it is an "enable" or "disable" request).

An error is flagged if a specified diagnostic SID is not defined for the Application Process,. However, this does not affect the processing of the other diagnostic SIDs in the request.

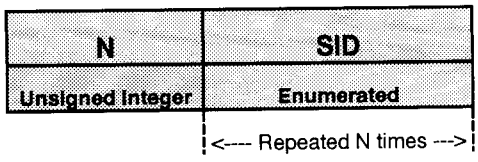
18.3.6 Reporting the List of Active Diagnostic Packets

The request to report the list of diagnostic packet definitions of the Application Process with an "Enabled" transmission status is:

Report Active Diagnostic Packets, (14,11)
Telecommand Packet, Application Data: **None**

When this request is received, the list of active diagnostic packet definitions from the Application Process is determined and a report is generated.

Active Diagnostic Packets Report, (14,12)
Telemetry Source Packet, Source Data:

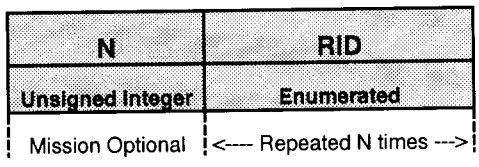


18.3.7 Controlling the Transmission of Specified Report Packets

The requests to enable or disable the transmission of specified report packets from the Application Process are:

Enable Transmission of Report Packets, (14,13)
Telecommand Packet, Application Data:

Disable Transmission of Report Packets, (14,14)
Telecommand Packet, Application data:



N: The number of report packet definitions to be enabled or disabled.

This field is systematically omitted if the Service only supports the control of one report packet at a time (i.e. N=1).

RID: The Report Identifier identifying a report packet definition.

When this request is received, the specified report packet definitions are removed from or added to the set of disabled report packet definitions (depending on whether it is an "enable" or "disable" request).

An error is flagged if a specified report RID is not defined for the Application Process. However, this does not affect the processing of the other report RIDs in the request.

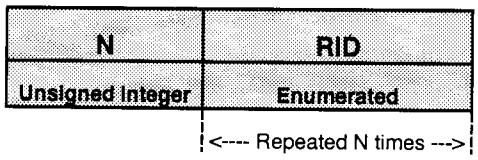
18.3.8 Reporting the List of Active Report Packets

The request to report the list of report packet definitions of the Application Process with an "Enabled" transmission status is:

Report Active Report Packets, (14,15)
Telecommand Packet, Application Data: **None**

When this request is received, the list of active report packet definitions from the Application Process is determined and a report is generated.

Active Report Packets Report, (14,16)
Telemetry Source Packet, Source Data:



18.4 Capability Sets

18.4.1 Minimum Capability Set

The capability to control the transmission of a given type of definition of a status reporting packet (i.e. a housekeeping, diagnostic or report packet) is only required if the corresponding type of definition can be generated by the Application Process.

Thus, the minimum capability set consists of the following Service Requests and Reports:

Sub-type	Service Request, Report or Capability
1	Enable Transmission of Telemetry Source Packets
2	Disable Transmission of Telemetry Source Packets

Table 18-2 Summary of Packet Transmission Control Minimum Capabilities

18.4.2 Additional Capability Sets

A Packet Transmission Control Service may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
3	Report Active Telemetry Source Packets	→sub-type 4
4	Active Telemetry Source Packets Report	→sub-type 3
5	Enable Transmission of Housekeeping Packets	→sub-type 6
6	Disable Transmission of Housekeeping Packets	→sub-type 5
7	Report Active Housekeeping Packets	→sub-type 8 and sub-type 5
8	Active Housekeeping Packets Report	→sub-type 7
9	Enable Transmission of Diagnostic Packets	→sub-type 10
10	Disable Transmission of Diagnostic Packets	→sub-type 9
11	Report Active Diagnostic Packets	→sub-type 12 and sub-type 9
12	Active Diagnostic Packets Report	→sub-type 11
13	Enable Transmission of Report Packets	→sub-type 14
14	Disable Transmission of Report Packets	→sub-type 13
15	Report Active Report Packets	→sub-type 16 and sub-type 13
16	Active Report Packets Report	→sub-type 15

Table 18-3 Summary of Packet Transmission Control Additional Capabilities

19 Onboard Storage and Retrieval Service

19.1 Scope

The Onboard Storage and Retrieval Service is a supporting service used by other oonboardServices to selectively store the Service Reports which they generate in order to give the ground the possibility to request the retrieval and downlink of the selectively stored data.

The Onboard Storage and Retrieval Service consists of two parts:

- a **Packet Selection sub-service**: selection and transfer of telemetry source packets for storage in Packet Stores;
- a **Storage and Retrieval sub-service**: storage and retrieval of telemetry source packets from Packet Stores.

Any number of onboard Application Processes may provide a single instance of an OonboardStorage and Retrieval Service.

Telemetry source packets and telecommand packets relating to the Onboard Storage and Retrieval Service are denoted by **Service Type = 15**.

19.2 Service Concept

19.2.1 General

The Onboard Storage and Retrieval Service is used if a mission requires onboard storage of telemetry source packets with the capability to selectively store the packet types and sub-types in different Packet Stores.

For example, for missions with intermittent coverage, packets of high operational significance (e.g. anomaly report packets) which are generated during a period of non-coverage could be stored in a dedicated Packet Store so that they can be retrieved first during the next period of coverage.

The Service may 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 cyclic Packet Stores.

One or more packet types and sub-types generated by one or more Application Processes may be selected for storage in a given Packet Store managed by a given Application Process.

The Application Process which is responsible for selecting which telemetry source packets should be sent for storage, either to another Application Process or to its own Storage and Retrieval sub-service, provides the Packet Selection sub-service.

The Application Process responsible for managing the Packet Store(s) provides the Storage and Retrieval sub-service.

A Packet Store is uniquely identified by a "Store ID". The Packet Selection sub-service knows implicitly (e.g. by virtue of its design) which Application Process provides the Storage and Retrieval sub-service which manages a Packet Store to which it will transfer packets.

The definition of the storage selection used by a given Packet Selection sub-service may either be predefined or changeable by the ground.

Packets are ordered according to their time of arrival at the Storage and Retrieval sub-service.

Telemetry source packets stored in a Packet Store are downlinked on request. The downlink request may specify that all, or a subset (e.g. within a specified packet range), of the stored packets be downlinked.

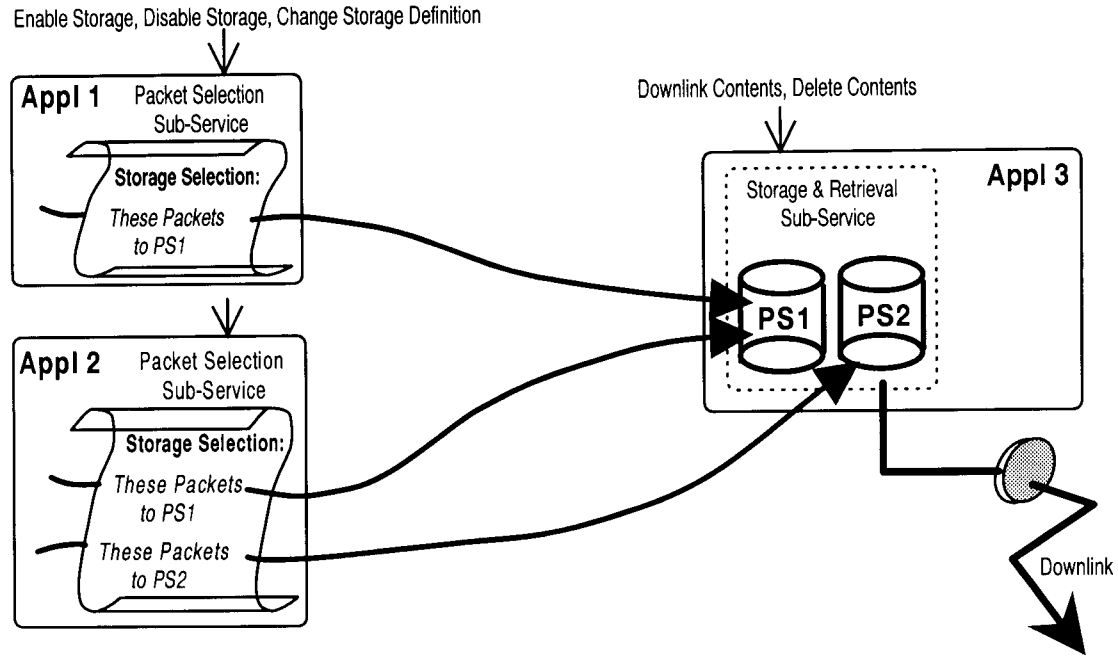


Figure 19-1 An Example of a Centralised Storage and Retrieval Approach

The Storage and Retrieval sub-service may (optionally) time stamp each telemetry source packet it receives with the time of reception/storage. Where this service is offered, the ground can request the downlink of packets from a Packet Store which were stored after, before or between specified times.

The Packet Storage and Retrieval sub-service may use the Large Data Transfer Service to downlink the requested packets, if appropriate. The Service Data Unit containing the data to be downlinked is constructed by retrieving from the Packet Store those packets which meet the selection criteria specified by the ground. The position within the Service Data Unit corresponds 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.

19.2.2 Packet Store Types

Each Packet Store has attributes, which are accessed by the Storage and Retrieval sub-service which manages it, and which indicate:

- whether the storage strategy is cyclic 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 **cyclic** Packet Store is full, any subsequently received packet over-writes partly or fully the oldest packet(s) in the list. A Packet Store used to provide a "lost recovery packet" capability must be cyclic in nature.

When a **bounded** Packet Store is full, any subsequently received packet is ignored. The contents of the Packet Store, or at least the oldest part of it, must be explicitly deleted in order to free storage space and to allow the resumption of storage of new packets. This type of deletion (i.e. erasing the contents of a Packet Store) may also be performed on a cyclic 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 may not be reliable. If, for any reason, packets cannot be stored, the Service is not required to 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 cyclic Packet Store are being downlinked, the Storage and Retrieval sub-service may not always 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 is given to the downlink activity, i.e. the overwriting pointer must always stay behind the retrieval pointer. In other words, new packets are not 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 are received directly by the ground, while the oldest packets in the Packet Store would otherwise be lost if they were not protected from being overwritten.

The Storage and Retrieval sub-service maintains status data about each Packet Store which is downlinked on a regular basis (e.g. through housekeeping parameters reports, as defined in Chapter 7) to provide insight into the storage and retrieval load and activities. For example, this could include the percentage of filling of the Packet Store, the storage time of the oldest packet in the Packet Store or the amount of data not yet retrieved from the Packet Store.

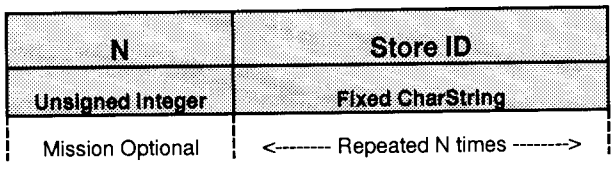
19.3 Service Requests and Reports

19.3.1 Controlling the Storage in Specified Packet Stores

The requests are:

Enable Storage in Packet Stores, (15,1)
Telecommand Packet, Application Data:

Disable Storage in Packet Stores, (15,2)
Telecommand Packet, Application data:



N: The number of packet stores to be controlled. By convention, N = 0 means "all Packet Stores".

This field is systematically omitted if the Service only supports one Packet Store (this is equivalent to having N=0).

Store ID:
An onboard Packet Store is uniquely identified by a "Store ID". This is a character string which may explicitly or implicitly indicate, for example, an access path to a physical onboard 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 receives this request, it starts or stops (depending on whether it is an "enable" or "disable" request) sending the relevant packets to the Application Processes managing the specified Packet Stores.

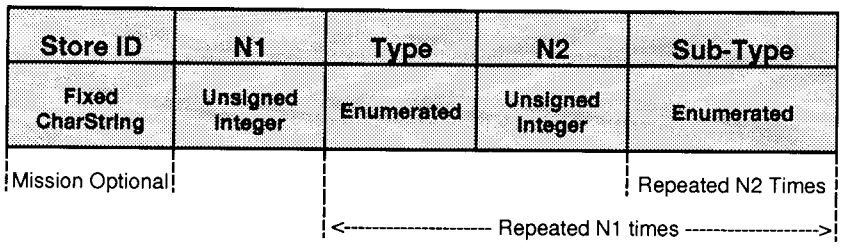
19.3.2 Modifying the Definition of a Storage Selection Criteria

The storage selection definition used by an Application Process to send packets for storage in a given Packet Store consists of the identification of the type and sub-type of the relevant packets. It is possible to add packet types and sub-types to (or remove from) a storage selection definition.

The requests to modify the storage selection definition for a specified Packet Store are:

Add Packet Types & Sub-Types to Storage Selection Definition, (15,3)
Telecommand Packet, Application Data:

Remove Packet Types & Sub-Types from Storage Selection Definition, (15,4)
Telecommand Packet, Application Data:



Store ID:
This field is systematically omitted if the Service only supports one Packet Store.

Type: A telemetry source packet Service Type.

Sub-Type:
A telemetry source packet Service Sub-Type of the specified Service Type.

Note that the above data structure is very similar to the that for controlling the transmission of packets defined in Section 18.3.1.

When the packet selection sub-service receives this request:

- if **N1 = 0**, all types of telemetry source packet from the Application Process are added to, or removed from, the list of packet types to be stored in the specified Packet Store (depending on the type of request);

- if **N1 > 0** and **N2 = 0**, the specified types of telemetry source packet from the Application Process are 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);

- if **N1 > 0** and **N2 > 0**, the specified sub-types of telemetry source packet from the Application Process are added to, or removed from, the list of packet sub-types (for the specified type) to be stored in the specified Packet Store (depending on the type of request).

The request has 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 sub-types are already selected for storage).

Note that if N1 > 1 then there may be a mixture of empty (N2 = 0) and non-empty arrays (N2 > 0).

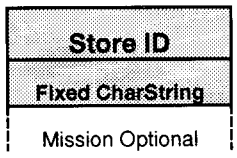
The current content of the Packet Store is not affected by the request and, if storage is enabled, packets start or stop to be appended to the Packet Store immediately after the request is processed.

The creation, deletion or modification of Packet Stores (together with their attributes, e.g. size, whether cyclic or bounded) is beyond the scope of this Standard.

19.3.3 Reporting a Storage Selection Definition

The request is:

Report Storage Selection Definition, (15,5)
Telecommand Packet, Application Data:

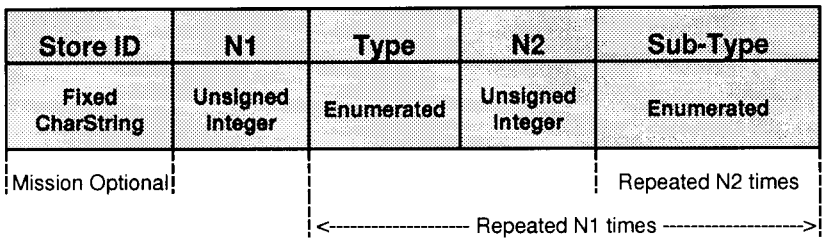


Store ID:
This field is systematically omitted if the Service only supports one Packet Store.

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 is read and a report is generated.

Note that since the destination Application Process uses a local storage selection definition for selecting which of its packets to send to a Packet Store (which may be managed by another Application Process), what is downlinked may 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 is systematically omitted if the Service only supports one Packet Store.

If all the types of packet generated by an Application Process are selected for storage in the Packet Store then N1 = 0 (and no parameters follow).

If all the sub-types of a given type of packet generated by an Application Process are selected for storage in the Packet Store, then N2 = 0 (and no sub-types follow).

19.3.4 Downlinking the Contents of a Packet Store for a Specified Packet Range

The request is:

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
Mission Optional	Mission Optional	<---------- Optional ---------->		<---------- Optional ---------->	

Store ID:
This field is systematically omitted if the Service 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 are to 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 is systematically omitted if the Service only supports one way of specifying the packet range. Note that the Service 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 are to 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 are present if Packet Set is not "All". The fields Application Process ID 2 and Source Sequence Count 2 are present if Packet Set is "Between".

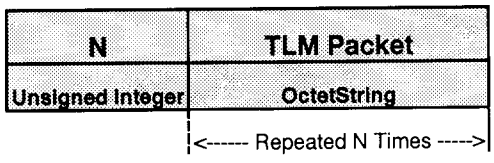
When this request is received by the Application Process which provides the Storage and Retrieval sub-service, the contents of the specified Packet Store falling within the specified packet range is downlinked. Whatever the value of Packet Set, the retrieval ends at the latest when the last packet stored 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 starts 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 ends 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).

The retrieved packets are placed in a report following their storage order and downlinked (using the Large Data Transfer Service, if appropriate).

Packet Store Contents Report, (15,8)
Telemetry Source Packet, Source Data:

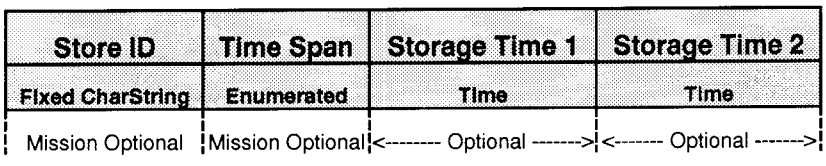


TLM Packet:
A telemetry source packet of any Type retrieved from the Packet Store.

19.3.5 Downlinking the Contents of a Packet Store for a Specified Time Period

The request is:

Downlink Packet Store Contents for Time Period, (15,9)
Telecommand Packet, Application Data:



Store ID:
This field is 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 are to 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);
- greater than or equal to Storage Time 1 if Time Span is "After" (value = 3).

This field is systematically omitted if the Service only supports one way of specifying the packet range. Note that the Service 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 are uniquely defined for the Application Process for which the request is destined.

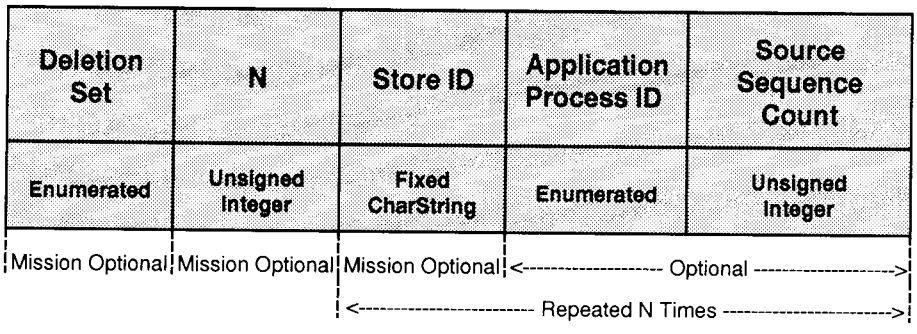
Storage Time 1 is present if Time Span is not "All". Storage Time 2 is present if Time Span is "Between".

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 are downlinked. Whatever the value of Time Span, the retrieval ends at the latest when the last packet stored at the time of reception of the request has been downlinked. The packets are placed in a ***Packet Store Contents Report*** (as defined in the previous section) and downlinked (using the Large Data Transfer Service, if appropriate).

19.3.6 Deleting the Contents of Specified Packet Stores up to Specified Packets

The request is:

Delete Packet Stores Contents up to Specified Packets, (15,10)
Telecommand Packet, Application Data:



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 is to be deleted. If Deletion Set is "Before" (value = 1), it is the set of packets up to the specified packet.

This field is systematically omitted if the Service only supports one way of specifying the part of the Packet Store to be deleted.

N: By convention, N = 0 means "Delete the full contents of all Packet Stores".

This field is systematically omitted if the Service only supports one Packet Store (equivalent to having N=1).

Store ID:
This field is systematically omitted if the Service 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 are to 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 are deleted. The deletion ends 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 are deleted.

If a packet defining the upper boundary of the packet range is not in the Packet Store, the deletion ends with the last packet from the same Application Process before the missing packet (or nothing is 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 may be recorded in the freed space of the Packet Store.

19.3.7 Deleting the Contents of Specified Packet Stores up to a Specified Storage Time

The request is:

Delete Packet Stores Contents up to Specified Storage Time, (15,11)
Telecommand Packet, Application Data:

End Time	N	Store ID
Time	Unsigned Integer	Fixed CharString
	Mission Optional	<----- Repeated N Times ----->

End Time:
The absolute time defining the upper boundary (inclusive) of the packet range to be deleted. The format and length of this parameter is uniquely defined for the Application Process for which the request is destined.

N: By convention, N = 0 means "All Packet Stores".

This field is systematically omitted if the Service 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 are deleted. The deletion ends 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 may be recorded in the freed space of the Packet Store.

19.4 Capability Sets

19.4.1 Minimum Capability Sets

The minimum capability set for a Service providing the Packet Selection sub-service consists of the following Service Requests and Reports:

Sub-type	Service Request or Report
1	Enable Storage in Packet Stores
2	Disable Storage in Packet Stores

Table 19-1 Summary of Packet Selection Sub-service Minimum Capabilities

The minimum capability set for a Service providing the Storage and Retrieval sub-service consists of the following Service Requests and Reports:

Sub-type	Service Request or Report
8	Packet Store Contents Report (thus 7 and/or 9 also)

Table 19-2 Summary of Storage and Retrieval Sub-service Minimum Capabilities

19.4.2 Additional Capability Sets

A Service providing the Packet Selection sub-service may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
3	Add Packet Types & Sub-Types to Storage Selection Definition	→sub-type 4
4	Remove Packet Types & Sub-Types from Storage Selection Definition	→sub-type 3
5	Report Storage Selection Definition	→sub-type 6
6	Storage Selection Definition Report	→sub-type 5

Table 19-3 Summary of Packet Selection Sub-service Additional Capabilities

A Service providing the Storage and Retrieval sub-service may implement the following additional capabilities:

Sub-type	Service Request, Report or Capability	Implication(s)
	Support of Cyclic Packet Store	
	Support of Bounded Packet Store	→sub-type 10 or sub-type 11
	Time stamping of stored packets	→sub-type 9
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	

Table 19-4 Summary of Storage and Retrieval Sub-service Additional Capabilities

20 Onboard Traffic Management Service

20.1 Scope

The specification of this Service will be contained in Issue 2 of this standard.

The Onboard Traffic Management Service provides all functions for (among other things):

i) the allocation of onboard routes and the reporting of onboard routing information, e.g. as used for the mapping of APIDs to MAPs;

ii) reporting by the PAC on any problems concerned with the re-assembly of telecommand packets from telecommand segments;

iii) the routine reporting of packet bus management parameters, such as average/peak loading, numbers of packet retransmissions etc.;

iv) the detection and reporting of onboard traffic problems of any nature, e.g. overflow in the distribution of telecommand packets or of telemetry source packets on the packet bus;

v) ground control over onboard traffic problems.

Telemetry source packets and telecommand packets relating to the Onboard Traffic Management Service are denoted by **Service Type = 16**.

21 Test Service

21.1 Scope

The Test Service provides the capability to activate test functions implemented onboard and to report the results of such tests.

Each Application Process provides a single instance of the Test Service.

Telemetry source packets and telecommand packets relating to the Test Service are denoted by **Service Type = 17**.

21.2 Service Concept

It is foreseen that the majority of test functions will be mission-specific in nature. The only generic test identified for this standard is an end-to-end "connection test" between the ground and the Application Process.

The function exercised by the connection test Service Request is the generation of a corresponding one-shot Service Report by the Application Process. The reception on the ground of the Service Report will 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).

21.3 Service Requests and Reports

The request to perform an end-to-end connection test is:

Perform Connection Test, (17,1)
Telecommand Packet, Application Data: **None**

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

Link Connection Report, (17,2)
Telemetry Source Packet, Source Data: **None**

21.4 Capability Sets

21.4.1 Minimum Capability Set

The minimum capability set consists of the following Service Requests and Reports:

Sub-type	Service Request or Report
1	Perform Connection Test
2	Connection Test Report

Table 21-1 Summary of Test Service Minimum Capabilities

21.4.2 Additional Capability Set

There are currently no additional capabilities defined.

22 Summary of Service Requests and Reports

22.1 Introduction

The table below 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 Sub-type (ST) for Requests and Reports which constitute the Minimum Capability Set for a given Service is indicated in the table by a shaded background. Note that 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 chapter of this Standard.

ST	Service Requests	ST	Service Reports
Telecommand Verification Service - 1			
		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 Complete Report - Success
		8	Telecommand Execution Complete Report - Failure
Device Command Distribution Service - 2			
1	Distribute On/Off Commands		
2	Distribute Register Load Commands		
3	Distribute CPDU Commands		
4	Distribute Command in TC Segment		
Housekeeping and Diagnostic Data Reporting - 3			
1	Define New Housekeeping-Parameter Report		
2	Define New Diagnostic-Parameter Report		
3	Clear Housekeeping-Parameter Report Definitions		

ST	Service Requests	ST	Service Reports
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	14	Housekeeping-Parameter Sampling Time Offsets Report
15	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 Masked Housekeeping-Parameter	22	Masked Housekeeping-Parameter Report
23	Report Masked Diagnostic-Parameter	24	Masked Diagnostic-Parameter Report
		25	Housekeeping-Parameter Report
		26	Diagnostic-Parameter Report
Parameter Statistics Reporting - 4			
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		
Event Reporting - 5			
		1	Normal/Progress Report
		2	Error/Anomaly Report - Low Severity

ST	Service Requests	ST	Service Reports
		3	Error/Anomaly Report - Medium Severity
		4	Error/Anomaly Report - High Severity
Memory Management Service - 6			
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
Task Management Service - 7			
1	Start Task		
2	Stop Task		
3	Suspend Task		
4	Resume Task		
5	Abort Task		
6	Perform Activity within Task		
Function Management Service - 8			
1	Activate Function		
2	De-activate Function		
3	Perform Activity of Function		
Time Management Service - 9			
1	Change Time Report Generation Rate		
Time Reporting Service - 10			
		1	Time Report
Onboard Scheduling Service - 11			
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		
7	Time-Shift Telecommands		
8	Time-Shift Telecommands over Time Period		
9	Report Command Schedule in Detailed Form	10	Detailed Schedule Report
11	Report Command Schedule in Detailed Form over Time Period	(10)	

ST	Service Requests	ST	Service Reports
12	Report Command Schedule in Summary Form	13	Summary Schedule Report
14	Report Command Schedule in Summary Form over Time Period	(13)	
Onboard Monitoring Service - 12			
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	Out-of-limit Report
Large Data Transfer Service - 13			
Data Downlink Operation			
		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		
Data Uplink Operation			
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
Packet Transmission Control Service - 14			
1	Enable Transmission of Packets		
2	Disable Transmission of Packets		
3	Report Active Packets	4	Active Packets Report
5	Enable Transmission of Housekeeping Parameters Reports		

ST	Service Requests	ST	Service Reports
6	Disable Transmission of Housekeeping Parameters Reports		
7	Report Active Housekeeping Parameters Reports	8	Active Housekeeping Parameters Reports Report
9	Enable Transmission of Diagnostic Parameters Reports		
10	Disable Transmission of Diagnostic Parameters Reports		
11	Report Active Diagnostic Parameters Reports	12	Active Diagnostic Parameters Reports Report
13	Enable Transmission of Event Reports		
14	Disable Transmission of Event Reports		
15	Report Active Event Reports	16	Active Event Reports Report
Onboard Storage and Retrieval Service - 15			
	Packet Selection Sub-service		
1	Enable Storage in Packet Stores		
2	Disable Storage in Packet Stores		
3	Add Packet Types & Sub-Types to Storage Selection Definition		
4	Remove Packet Types & Sub-Types from Storage Selection Definition		
5	Report Storage Selection Definition	6	Storage Selection Definition Report
	Storage and Retrieval Sub-service		
7	Downlink Packet Store Contents for Packet Range	8	Packet Store Contents Report (thus 3 and/or 5 also)
9	Downlink Packet Store Contents for Time Period	(8)	
10	Delete Packet Stores Contents up to Specified Packets		
11	Delete Packet Stores Contents up to Specified Storage Time		
Onboard Traffic Management Service - 16			
No service sub-types defined for the moment.			
Test Service - 17			
1	Perform Connection Test	2	Connection Test Report

Table 22-1 Summary of Requests and Reports for PUS Standard Services

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 field containing a parameter value or a structured field. A structured field contains several parameter fields organised 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 Section 23.5.

A structured field has a **structured type** indicating the high level organisation 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 Section 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 chapter form a subset (sometimes with explicit restrictions or extensions) of the simple and structured types defined in **ISO 8824 "Specification of Abstract Syntax Notation One (ASN.1)"**.

The ASN.1 notation is not used in this document. 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 chapter defines the physical encoding rules for each simple and structured type, that is their permitted lengths and the internal format used to encode values.

These encoding rules differ from the rules defined in **ISO 8825 "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)"** for several reasons, the main reason being the generally applicable mission constraints imposed by the space link bandwidth.

This standard does not recommend a particular length (and physical encoding format) for each field in the Packet Data Field of the packets. A mission has to 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 a N-bit field:

> The first bit in a field, starting from the left, is defined to be "Bit-0" and will be 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 document 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 are possible variations in the format and length of the values.

Each permitted 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 is used whenever an explicit definition of a parameter field is required (e.g. to specify the physical format selected by a mission; for parameter interpretation on the ground). Although the parameter code has not been used within the packets defined to date, there may be future packet structures where this definition is required. 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 chapter):

PC =

PTC	PFC
Enumerated	Enumerated

Where:

PTC = Parameter Type Code }
PFC = Parameter Format Code } PC = Parameter Code

The only parameter types allowable for telemetry and telecommand parameters are those defined in Sections 23.5.1 to 23.5.9 below. For each type of parameter, a list of standard parameter formats is defined in this Standard. Missions should strive to adopt these standard parameter formats, but other mission-specific parameter formats may 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 is actually allowed to take may 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 indicate the parameter types of the parameter fields in the corresponding packet structures. A mission selects the actual physical format by specifying the Parameter Code of each parameter field found in each packet type and sub-type.

For example, the format of the Time field in the Packet Data Header of the telemetry source packets will normally be selected on the basis of the time resolution required for the monitoring of an Application Process.

The actual physical format for a parameter field may be:

- **Mission Defined**: The physical format of the parameter field is unique for the mission. The parameter field may appear in different packet types and sub-types and/or may appear in a packet sent to or generated by different Application Processes; in all cases its physical format is the same.

- **Application Process Defined**: The physical format of the parameter field is defined per Application Process. The encoding of the parameter field is identical in all packet types and sub-types sent to or generated by the Application Process.

- **Service Instance Defined**: The physical format of the parameter field is defined per Service provided by an Application Process. The encoding of the parameter field is identical in all packet sub-types received or generated by a Service provided by a given Application Process (this may reflect, for example, the fact that several organisations implement the onboard software).

- **Request/Report Defined**: The physical format of the parameter field in a packet is deduced from the value(s) of one or several preceding parameter fields in the packet (including the packet type and sub-type). It may additionally depend on the value(s) of one or several mission parameter(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 may also be Mission Defined, Application Process Defined, Service Instance Defined or Request/Report Defined. The specification of the packet structures defined in this Standard highlights which parameter fields have their types defined in this way by indicating that their type is "**Deduced**".

The presence or absence of a parameter field specified as "**Mission Optional**" in the structure of a packet may be Mission Defined, Application Process Defined or Service Instance Defined.

The presence or absence of a parameter field specified as "**Optional**" in the structure of a packet is Request/Report Defined.

A mission may require that a parameter field starts on a byte boundary. This may be achieved by ensuring that the preceding parameter field ends on a byte boundary (increasing its size as needed) or by introducing filler bits between two parameter fields.

When a mission tailors the packet structures to suit its needs, it should take into account the field tailoring restrictions and defaults defined in **[RD5]** ESA PSS-07-201, "Packet Data Description Standard".

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 = 1,2,4,8,12,16,24 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 un-ordered. Each value has a meaning which is interpreted as a character string value. An error code is a typical example (e.g. 0 means "un-checked", 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 is encoded with Bit-0 being the MSB and Bit-N-1 the LSB.

The possible formats of an Unsigned Integer are:

Format Code	Format Definition	Lowest Value	Highest Value
0 <= PFC <= 12	PFC+4 bits, unsigned	0	$2^{PFC+4}-1$ (15 to 65535)
PFC = 13	3 octets, unsigned	0	$2^{24}-1$ (16777215)
PFC = 14	4 octets, unsigned	0	$2^{32}-1$ ($\approx 4.3*10^{9}$)
PFC = 15	6 octets, unsigned	0	$2^{48}-1$ ($\approx 2.8*10^{14}$)
PFC = 16	8 octets, unsigned	0	$2^{64}-1$ ($\approx 18.5*10^{19}$)

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 is positive following the unsigned integer convention. If Bit-0 = 1, the value is negative and the field is the 2's complement of the positive value.

The possible formats of a Signed Integer are:

Format Code	Format Definition	Lowest Value	Highest Value
0 <= PFC <= 12	PFC+4 bits, signed	-2^{PFC+3} (-8 to -32768)	$2^{PFC+3}-1$ (7 to 32767)
PFC = 13	3 octets, signed	-2^{23} (-8388608)	$2^{23}-1$ (8388607)
PFC = 14	4 octets, signed	-2^{31} ($\approx -2.15*10^{9}$)	$2^{31}-1$ ($\approx 2.15*10^{9}$)
PFC = 15	6 octets, signed	-2^{47} ($\approx -1.4*10^{14}$)	$2^{47}-1$ ($\approx 1.4*10^{14}$)
PFC = 16	8 octets, signed	-2^{63} ($\approx -9.2*10^{18}$)	$2^{63}-1$ ($\approx 9.2*10^{18}$)

23.5.5 Real Parameter (PTC = 5)

PFC = 1: 4 octets simple precision format
PFC = 2: 8 octets double precision format

The simple-precision format and the double-precision format are defined in "**IEEE Standard for Binary Floating-Point Arithmetic**", IEEE Std 754-1985. The salient features of their definitions are repeated here for completeness.

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

$$(-1)^{S} \times 2^{E} \times (b_0 \bullet b_1 b_2 .....b_{p-1})$$

where

$b_0 \bullet b_1 b_2 ... b_{p-1}$ means $\frac{b_0}{2^0} + \frac{b_1}{2^1} + \frac{b_2}{2^2} + ... + \frac{b_{p-1}}{2^{p-1}}$

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 sub-fields:

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 $\bullet b_1 b_2 .... b_{p-1}$ with p=24 (single precision) or p=53 (double precision)

The encoded value of a single-precision real parameter is constituted as follows:

Sign	Exponent	Fraction
1 bit	8 bits	23 bits

The encoded value of a double-precision real parameter is constituted as follows:

Sign	Exponent	Fraction
1 bit	11 bits	52 bits

The value of a single-precision parameter is:

If exponent = 255 and fraction <> 0	value = Not a Number
If exponent = 255 and fraction = 0	value = $(-1)^{sign} \times \infty$
If 0 < exponent < 255	value = $(-1)^{sign} \times 2^{exponent-127} \times (1.fraction)$
If exponent = 0 and fraction <> 0	value = $(-1)^{sign} \times 2^{-126} \times (0.fraction)$
If exponent = 0 and fraction = 0	value = 0

The value of a double-precision parameter is:

If exponent = 2047 and fraction <> 0	value = Not a Number
If exponent = 2047 and fraction = 0	value = $(-1)^{sign} \times \infty$
If 0 < exponent < 2047	value = $(-1)^{sign} \times 2^{exponent-1023} \times (1.fraction)$
If exponent = 0 and fraction <> 0	value = $(-1)^{sign} \times 2^{-1022} \times (0.fraction)$
If exponent = 0 and fraction = 0	value = 0

In the cases where Exponent = 0 and Fraction <> 0, the values are called denormalised.

The range of possible values and precision for a real parameter are as follows:

Simple precision

$1.12*10^{-38}$ <= |value| <= $3.40*10^{38}$ (precision $1.15*10^{-7}$)

Double precision

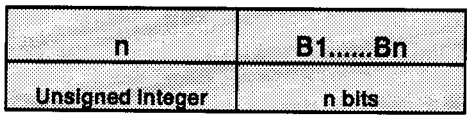
$2.22*10^{-308}$ <= |value| <= $1.79*10^{308}$ (precision $2.22*10^{-16}$)

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 specific.

A variable-length bit-string parameter is of the form:



Where:
B1....Bn are bits.
n indicates the number of bits which follows.

A fixed-length bit-string parameter is of the form:

Where:
B1....Bn 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 is application specific.

A variable-length octet-string parameter is of the form:

n	O1	O2	- - - - - - -	On
Unsigned Integer	octet	octet		octet

Where:

O1....On are octets.
n indicates the number of octets which follows.

A fixed-length octet-string parameter is of the form:

O1	O2	- - - - - - -	On
octet	octet		octet

Where:

O1....On 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**). A visible character is represented by its ASCII code on one octet. The meaning and interpretation of a value is application specific.

A variable-length character-string parameter is of the form:

n	C1	C2	- - - - - - -	Cn
Unsigned Integer	ASCII	ASCII		ASCII

Where:

C1....Cn are ASCII character codes on 8 bits.
n indicates the number of ASCII character codes which follows.

A fixed-length character-string parameter is of the form:

C1	C2	- - - - - - -	Cn
ASCII	ASCII		ASCII

Where:

C1....Cn are ASCII character codes.
n is the number of ASCII character codes and is equal to PFC.

23.5.9 Time Parameter (PTC = 9) and Relative Time Parameter (PTC = 10)

PFC = 0: Full definition of time format (CUC or CDS), i.e. including the p-field
PFC = 1: 2 bytes day CDS format without μseconds field (parameter field is 6 octets)
PFC = 2: 2 bytes day CDS format with μseconds field (parameter field is 8 octets)

3≤ PFC ≤18: CUC format with ((PFC+1) / 4) bytes of coarse time and ((PFC+1) modulo 4) bytes of fine time.

The CUC and CDS time formats are defined in **[AD4]**.

The value of a Time parameter field is a number of seconds and fractions of second from a given Agency epoch (i.e. it denotes an absolute time). It is a positive time offset.

The value of a Relative Time parameter field is a number of seconds and fractions of a second from the occurrence time of an event whose identification may be derived from other parameters in the packet (e.g. identifying a type of onboard event) or a number of seconds and fractions of a second between two absolute times. It can be a positive or negative time offset.

CDS Format

The CDS Format with μseconds (PFC = 2) is as follows:

Day	msec of day	μsec of msec
2 octets	4 octets	2 octets

The value of Day is an unsigned integer in the range 0 to 2^{16}-1.

Relative Time cannot be specified by means of a CDS Format.

CUC Format

The full CUC Format is as follows:

C1	C2	C3	C4	F1	F2	F3
1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet

For a Time parameter, the time in seconds from the given Agency epoch is given by:

$$t = C1*256^3 + C2*256^2 + C3*256 + C4 + F1*256^{-1} + F2*256^{-2} + F3*256^{-3}.$$

For a Relative Time parameter, a positive time offset is given by:

$$t = C1*256^3 + C2*256^2 + C3*256 + C4 + F1*256^{-1} + F2*256^{-2} + F3*256^{-3}$$

where C1 is in the range 0 to 127.

A negative time offset is expressed as the "2's complement" of the corresponding positive time offset.

23.5.10 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 sections. 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 and/or from the value(s) of mission parameters.

For example, consider 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. It would be described as illustrated below:

Parameter#	Parameter Value
Enumerated	Deduced

In this example, successive packets generated by an Application Process may 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, are defined by a structured type. The structured types of fields are defined through rules specifying how to organise 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 specialised packet structures within this Standard follow these rules. This set is referred to as **Structure rules set number 1** and is defined below.

23.6.2 Structure Rules (Set number 1)

The structured types are

- **Record** Type: it 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: it defines a field which is composed of zero, one or more components where all the components are of identical type (simple or structured).

  An Array may have a fixed number of entries (Fixed Array) or a variable number of entries (Variable Array).

- **Choice Structure** Type: it defines a field which is composed of one component whose actual type (simple or structured) is deduced from the value(s) of other preceding parameter field(s) in the packet and/or from the value(s) of mission parameters.

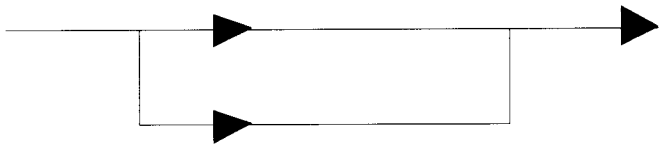
  A Deduced parameter field (see Section 23.5.10) is a particular case of a Choice Structure field.

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 may also be defined by a mission).

The structure rules utilise the following diagram conventions:

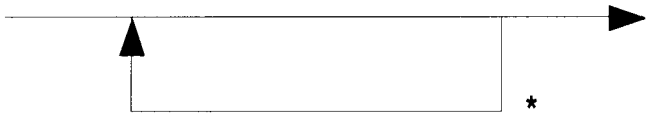
An oval denotes a definition of a simple component.

A box denotes a component built from simple or structured components.

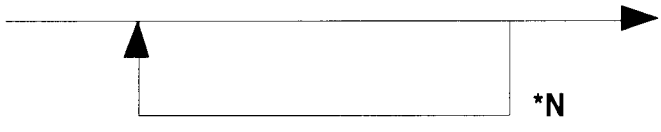


An exclusive "OR".

A repetition of components a fixed number of times, which is known implicitly.



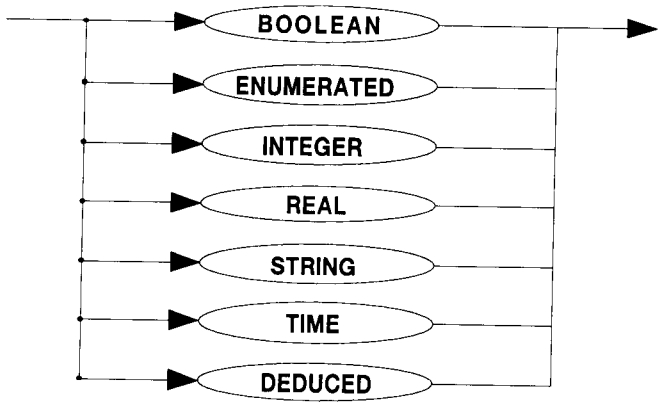
A repetition of the same component a fixed number of times, which is known implicitly.



A repetition of the same component N times

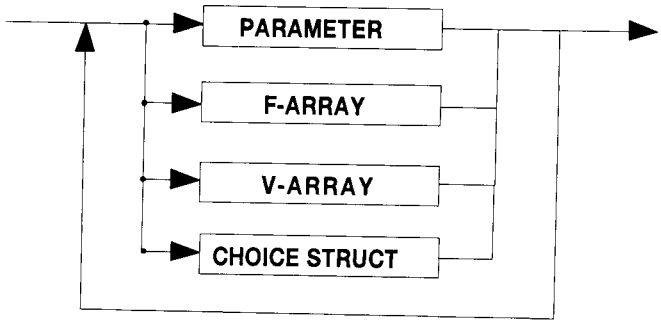
Structure Rules (Set number 1)

PARAMETER



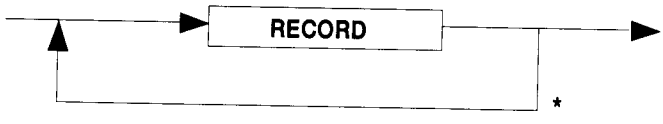
RECORD

A record is a sequence of distinct parameters, arrays or choice structures, where the structure of the record is known implicitly.



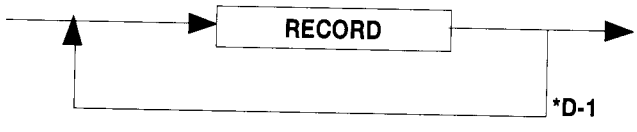
F-ARRAY

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. Note that the return loop operates D - 1 times. A Variable Array can only be empty if it is explicitly stated in its definition.



CHOICE-STRUCTURE

An instance of a "Choice Structure" can be of any type (simple or structured).

CHOICE-STRUCTURE (Contd)

A "Choice Structure" cannot be fully represented with the same diagram conventions since the actual type for an instance of the choice structure in a packet is deduced from the value(s) of one or several preceding parameter field(s) in the packet and/or from the value(s) of one or several mission parameters (called *"choice parameters"*). However, within a given instance of a packet, the Choice Structure is either a Parameter, a Record, a V-Array, a F-Array or even nothing.

A choice parameter could, for example, be an explicit sampling time or indicate an event to which the values in the Choice Structure relate.

There is a direct correspondence between the value(s) of the set of choice parameter(s) and the deduced Choice Structure instance (e.g. a look-up table could be used).

For example, a Choice Structure depending on a preceding parameter field P could 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 Section 23.6.3).

The type of a choice parameter can only be Boolean, Enumerated, Signed or Unsigned Integer or CharString. It can also be Deduced, in which case its actual type must depend on the value(s) of some of the other choice parameters.

The choice parameter(s) need not be placed consecutively in the packet nor just before the Choice Structure (i.e. there may be other fields separating them).

If an Array entry contains a Choice Structure, then the entry number may also be used to deduce the Choice Structure instance.

Nesting and Identification Rules:

NIR1: By definition, in a packet all components which are not components of any Array are at level 1. Thus 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 applies to the array entry number which is considered to be an implicit component of the array if it is used for deducing a Choice Structure instance.

The implications of the two foregoing rules are indicated below each example presented in Section 23.6.4.

NIR3: A parameter field in a packet may contain:

- the value of a *concrete parameter* (e.g. the sampled value of an onboard measurement or the value of a threshold to be set). A concrete parameter has a uniquely identifying name and its meaning and interpretation is uniquely defined for a mission.

- the value of a *formal parameter*. A formal parameter field has a meaning and interpretation which is defined with respect to one or more component(s). For 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 Section 23.5.10, 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 it must be at level 1 or 2 (thus all parameters at level 3 and above are formal).

This implies that the supercommutated values of a concrete parameter cannot be scattered "randomly" in a packet but must be components of one, and only one, Array. Also, a concrete parameter may be sub-commutated if it appears in the definition of a single Choice 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 specialised meaning in each case (e.g. if it appears in a Choice Structure providing information on an event, the specialised meaning may be "value of concrete parameter *when event occurred*"). This really means that the packet contains formal parameter fields which hold the "special purposes" values of the concrete parameter (these values may be scattered "randomly" in these packet).

NIR6: The last (or only) choice parameter of a Choice Structure must be at the same level or at one level less than the level of its Choice Structure. The first choice parameter of a Choice Structure must be at level 1 or 2, and there must be at least one choice parameter at each level between the level of the first choice parameter and the level of the last choice parameter.

This rule denotes the fact that an array hierarchy provides information which relates to a functional and/or physical hierarchy for which the choice parameter(s) define an access and interpretation path.

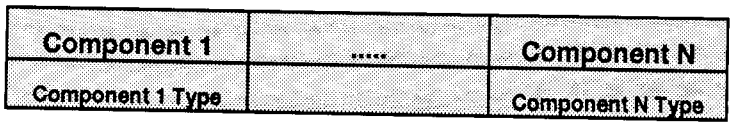
23.6.3 Encoding of the Structured Fields

The rules for the construction of structured types which are described in the previous section do not specify how the resulting construct is physically encoded in the packets.

This section 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 are placed consecutively in the packet, by means of their own physical encoding format:



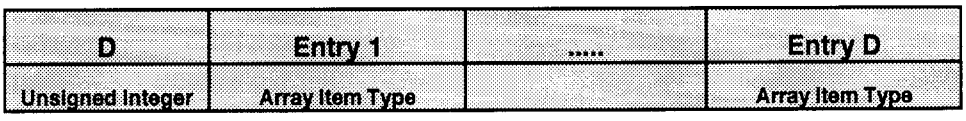
A parameter field is bit-wise aligned with the last bit of the parameter field which precedes it and occupies exactly the number of bits defined for its physical encoding (as defined by its PFC). A specific number of filler bits (see Section 23.4) may be introduced before the first component and between two successive components.

The value of a variable-length parameter type occupies exactly the actual number of bits required 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 are placed consecutively in the packet and are bit-wise aligned. Filler bits may be introduced before the array and between the successive entries of the array.

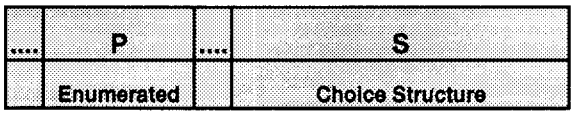
The set of entries of a Variable Array is preceded by an unsigned integer parameter field whose value (D) is the number of entries in the Variable Array:



Choice Structure Type

The Choice Structure is placed in the packet by means of the encoding format of its actual structure instance. It is bit-wise aligned with the preceding field in the packet. Filler bits may be introduced before the Choice Structure (it applies to all structure instances).

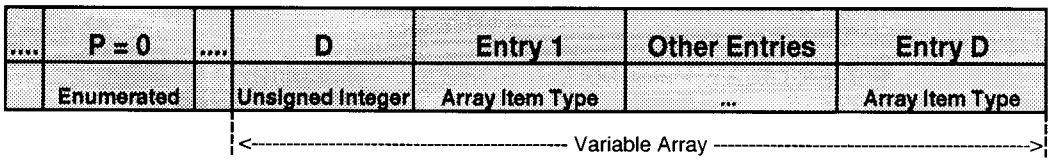
For example if a Choice Structure S depends on the value of a preceding choice parameter P, the generic structure is:

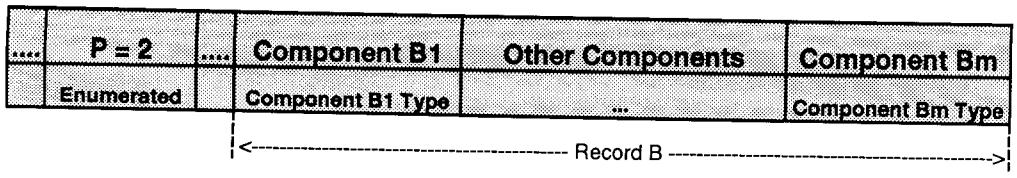


The actual variants of this Choice Structure could be

- a Variable Array when P = 0,
- a Record A when P = 1 and
- a Record B when P = 2.

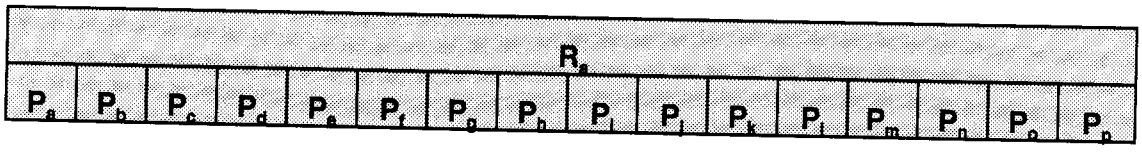
In this case the 3 resulting possible packet structures would be:





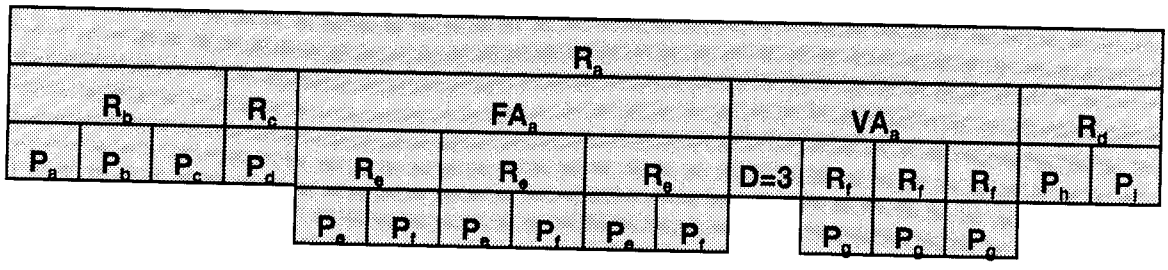
23.6.4 Examples of Structure Rules Set #1

A simple example of the use of the Structure Rules is a classical structure of the parameters field of a housekeeping packet containing no supercommutated parameters, viz:



In the above example, R_a and its components (P_a to P_p) are at level 1.

On the other hand, the following structure utilises the Array constructs defined in Structure Rules Set #1:



P = Parameter; R = Record; FA = Fixed array; VA = Variable array.

It should be noted that, in the second example, R_b and R_c could have been combined into one record without changing the lower level structure. However, it may often be useful to define separate records corresponding to different sets of information which may appear elsewhere in different structures.

In the above example, R_a, R_b, R_c, R_d (and their components), FA_a, VA_a and D are at level 1 while R_e and R_f (and their components) are at level 2.

Appendix A EXAMPLES

A.1 Examples/Explanations for Operational Requirements

The examples/explanations below are denoted in the same manner as the requirements of Chapter 3 to which they correspond.

TC-1 This requirement prohibits a telecommand packet which requests both the starting of an Onboard Schedule ***and*** the dumping of an unrelated area of onboard memory.

TC-6 Category 1 commands may also be sent from the ground using two (or three) independent telecommand packets; this will need to be decided on a case-by-case basis.

TC-10 These CPDU commands are also referred to as "High-priority" commands.

TC-11 There may be commands which should not be sent to a particular unit unless that unit is switched ON. These shall be inhibited on the ground unless the housekeeping telemetry indeed indicates that the unit is ON.

TC-12 During testing, there may be commands which should not be sent owing to environmental (e.g. vacuum) or physical constraints. Where there are no particular restrictions on the use of these commands in orbit, special measures shall be taken within the ground system to protect these commands from being sent.

TC-15 If a safety mode can be triggered by any one (or by any two or more) of several conditions, for example:

- sun sensors 1 to n showing Sun outside permitted zone;
- gyros 1 to m health monitor showing "FAIL";
- gyros 1 to p indicating a rate which exceeds a preset threshold;

then it shall be possible to remove (de-select) any of the sensor inputs which are tested, without affecting the overall functioning of the safety mode. This capability is required so that any recurrent indication, which is known to be spurious, can be by-passed.

TCTRANS-2
It shall not be necessary for the ground to introduce artificial delays in any uplink procedure to allow the onboard processing of telecommand packets or the generation of telecommand verification packets.

SSREP-4.1 Whether or not power is being drawn by a given unit shall be determined from a current sensor and not from monitoring a change in the temperature of the unit.

SSREP-6.2/6.3
In general this frequency shall be compatible with the natural frequencies of the phenomenon under observation, e.g.:

- in the case of a control loop with a time response of 10s the sensor telemetry should be sampled every 1s;

- for thermal parameters with a slow time response, a sampling interval of a few minutes is acceptable.

Furthermore, a telemetry parameter which verifies a telecommand execution shall be generated within a periodicity less than <TC_VERIF_DELAY>.

SSREP-19 In an earlier ESA mission, the capability existed to set up a "Diagnostic mode" as defined within this document, using a special Application Process within the Onboard Computer of the data handling subsystem. However, there were serious limitations in using this mode to over-sample AOCS parameters, since these could only be accessed via the AOCS micro-processor. The micro-processor operated on a 2-second software cycle, with one process collecting AOCS telemetry. The normal housekeeping telemetry did not require a sampling interval shorter than this 2-second interval. Although the diagnostic mode of the data handling subsystem could provide a sampling interval of 31.25 ms, the AOCS telemetry oversampling interval was, in practice, restricted to 2 seconds as a result of the inability to by-pass the AOCS micro-processor. This requirement ensures that ***all*** telemetry parameters are available to the Diagnostic mode process(es) at a sufficiently high rate.

TIMING-3.2/3.3

Note that there may be several mission parameters (<PARAM_ABS_SAMPL_TIME>, <PARAM_REL_SAMPL_TIME>) defined for the mission, for different classes of parameter. However in the most stringent case, a typical value of 10 milliseconds might be expected.

SWMAN-1.2

In some implementations, resetting an Application Process may be achieved by resetting the processor which hosts the Application Process.

SWMAN-2 Communication with an onboard function or task shall not be achieved, for example, by means of general-purpose memory write (memory load) and memory read (memory dump) packets.

SWMAN-4 The term "task" is used here in a very broad sense. It can be any entity which can be independently controlled, e.g. an ADA task in an Application Process, an operating system process etc. It could also be an application controlled by an Application Manager responsible for the supervision of all applications, or a hardware process, or a subsystem.

OBSCH-2 The interlocking option can be used to provide conditional branching within the Onboard Schedule, as follows:

If, for instance, it is desired to send commands B, C and D following successful execution of command A, but if command A fails execution, then commands E, F and G shall be sent instead, then this can be achieved by loading the following, for example:

t_1	A	
t_2	B	INT (A OK)
t_3	C	INT (A OK)
t_4	D	INT (A OK)
t_2	E	INT (A FAILED)
t_3	F	INT (A FAILED)
t_4	G	INT (A FAILED)

where INT (A OK) means that the command is sent only if the destination Application Process returns a code which indicates that command A has been executed successfully. If this code is not received then the command is presumed to have failed execution (A FAILED). Note that it is important that the commands interlocked to command failure (e.g. E, F, G above) should appear **at the same time or later than** commands interlocked to the success (e.g. commands A, B, C above) of the same command. Otherwise, there is chance that the interlocking command could register successful in the intervening period.

The same effect would be achieved by loading the following (this example indicates the potential use of Onboard Schedule control packets within the Onboard Schedule itself):

Sub-schedule 1

t_1	A	
t_2 - 5 sec	Start Sub-schedule 2	INT (A OK)
t_2 - 5 sec	Start Sub-schedule 3	INT (A FAILED)

Sub-schedule 2

t_2	B
t_3	C
t_4	D

Sub-schedule 3

t_2	E
t_3	F
t_4	G

STORE-1 In general, this requirement will not apply to payload data, although there may be User-generated requirements for onboard storage of payload data.

ROUTE-1 This implies that the identification of the Services and functions to which a packet relates, as well as its operational priority/criticality, must use information within the packet data field (in practice, the Type, Sub-type, FID etc. are used for this purpose).

ROUTE-2 The objective of this requirement is essentially to facilitate the implementation of telecommanding security measures on the ground. In a telescience scenario, for example, telecommands from an experimenter to his payload may be "screened" purely on the basis of APID.

ROUTE-4 This requirement could be fulfilled by the use of different MAPs for payload and platform commands.

ROUTE-5 The use of different virtual channels permits differentiation between different data-transmission bandwidths and/or priorities. Some typical examples are:

- different VCs for real-time data and for the playback of data generated onboard at an earlier time and stored onboard (deferred data);
- different VCs for different deferred packets (e.g. for different storage units) in order to ensure high-priority access to critical report packets;
- different virtual channels allocated to different payloads to ensure "guaranteed" bandwidth for real-time payload data.

A.2 Specification of the Cyclic Redundancy Code (CRC)

A.2.1 General Specification

The Packet Error Control Field provides the capability for detecting errors which may 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 subsections, 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 are generally expressed as follows:

a) The generator polynomial is:

$$g(x) = x^{16} + x^{12} + x^5 + 1$$

b) Both encoder and decoder are initialised to the "all-ones" state for each packet.

c) PCS generation is performed over the data space "D" as shown in Figure A.2-1 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 are generally expressed as follows:

- the proportion of all errors in the data that will not be detected is approximately 1.53×10^{-5};
- an error in the data affecting an odd number of bits will always be detected;
- an error in the data affecting exactly two bits, no more than 65535 bits apart, will always be detected;
- if an error in the data affects an even number of bits (greater than or equal to 4), the probability that the error will not be detected is approximately 3×10^{-5} for a data length of 4096 octets. The probability increases slightly for larger data lengths and decreases slightly for smaller data lengths;
- a single error burst spanning 16 bits or less of the data will always be detected. Note that 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 utilise it for correction.

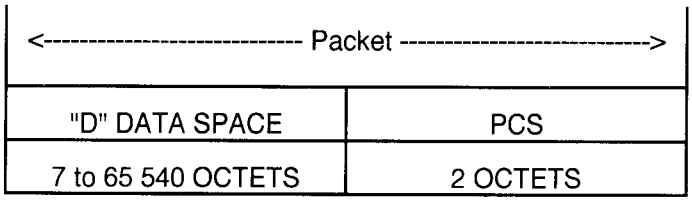


Figure A.2-1 Standard Packet Check Sequence Generation

A.2.2 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:

$$PCS = [X^{16} \cdot M(X) + X^{(n-16)} \cdot L(X)] \text{ MODULO } G(X)$$

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:

$$L(x) = \sum_{i=0}^{15} x^i \quad \text{(all "1" polynomial of order 15)}$$

G(X) is the CCIT Recommendation V.41 generating polynomial given by:

$$G(X) = X^{16} + X^{12} + X^5 + 1$$

+ is the modulo 2 addition operator (Exclusive OR).

Note that the encoding procedure differs from that of a conventional cyclic block encoding operation in that:

- The $X^{(n-16)} \cdot L(X)$ 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.2.3 Decoding Procedure

The error detection syndrome, S(X) is given by:

$$S(X) = [X^{16} \cdot C^*(X) + X^n \cdot L(X)] \text{ MODULO } G(X)$$

where: $C^*(X)$ is the received block in polynomial form.

S(X) is the syndrome polynomial which will be zero if no error has been detected.

A.2.4 Possible Realisation of a CRC Encoder - Decoder

This subsection describes two possible 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.2.4.1 Encoder

Figure A.2-2 shows a possible 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 will 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 required check bits will appear in succession at the output.

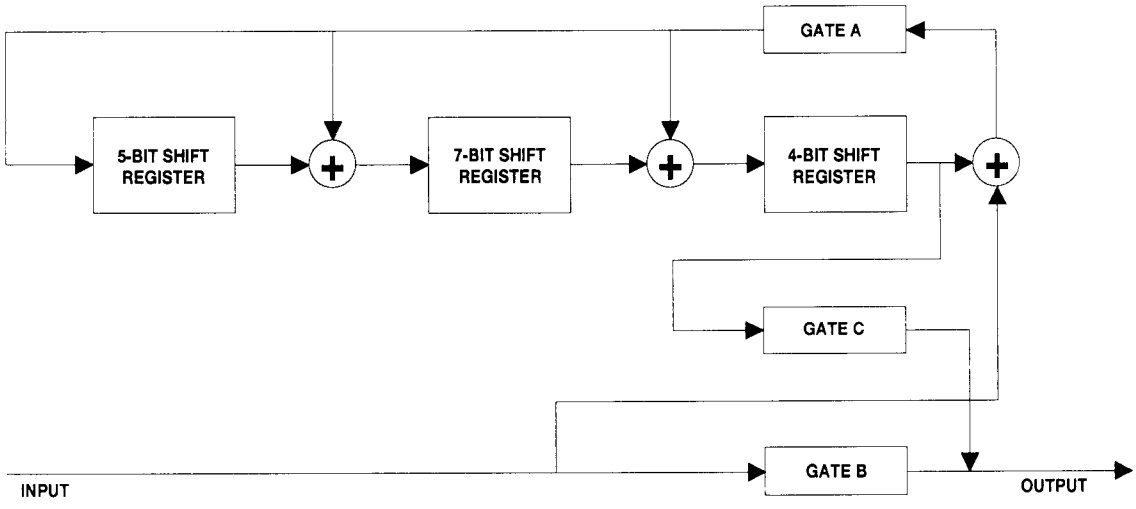


Figure A.2-2 ENCODER

A.2.4.2 Decoder

Figure A.2-3 shows a possible 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 will be "zero". Non-zero contents indicate an erroneous packet.

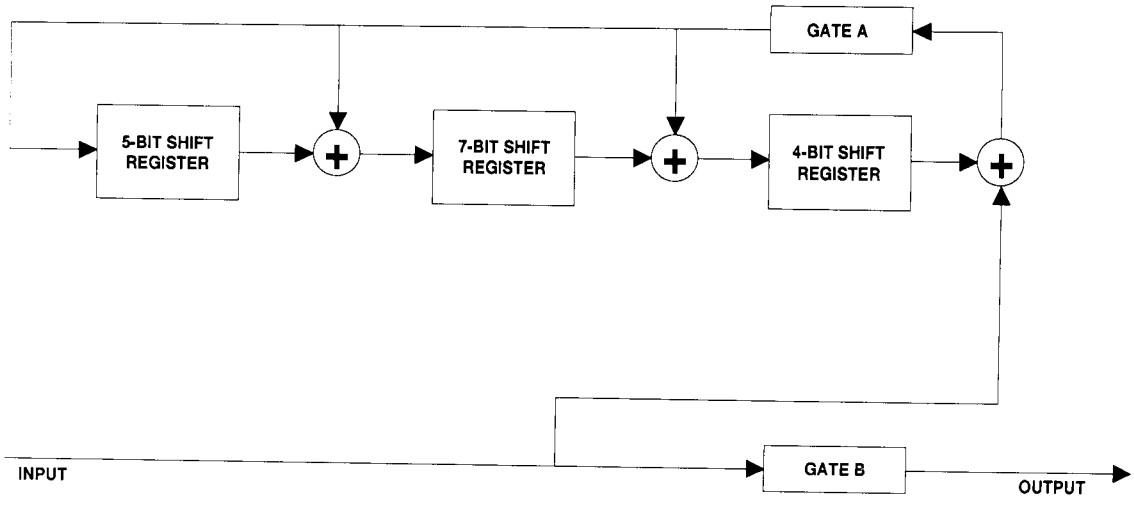


Figure A.2-3 DECODER

A.2.5 Verification of Compliance

The binary sequences defined in this section are provided to the designers of packet systems as samples for early testing, so that they can 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	F2 34

A.2.6 Software Implementation

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 required 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.

Functions required to generate the CRC placed at the end of a packet

Crc FUNCTION

The Crc function calculates the CRC for one byte in serial fashion and returns the value of the calculated CRC checksum.

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.

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) & OxFF00) ^ 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[5] = 0x00;  indata[6] = 0xEC;  indata[7] = 0x95;  indata[8] = 0x55

    Chk = 0xFFFF;   /* Reset syndrome to all ones    */

    for (j=0; j,8; j++)  {
       Chk = Crc(indata[j], Chk);              /* Un-optimized 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 o)\n", Chk);

  }
```

A.3 Specification of the ISO Checksum

A.3.1 General Specification

The standard error detection encoding/decoding procedure, which is described in detail in the following subsections, produces a 16-bit checksum (the "ISO Checksum") using integer arithmetic (see **[RD7]**, **[RD8]**).

The ISO checksum procedure can be easily implemented in software on typical processors using a compact and efficient algorithm. In contrast to the CRC algorithms (see Appendix A.2), 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 is to be used to check the contents of an onboard memory area using the services of the Memory Management Service (see Chapter 10). All octets of the onboard memory area are processed in turn and the calculated ISO checksum value is 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 may 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 are processed in turn and the calculated ISO checksum value is 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 are generally expressed as follows:

- the proportion of all errors in the data that will not be detected is approximately 1.54×10^{-5} i.e. the checksum detects virtually the same proportion of all errors as does the CRC;

- a single bit in error will always be detected;

- 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;

- an error in the data affecting exactly two bits, no more than 2040 bits apart, will always be detected;

- the probability that a single error burst spanning 16 bits or less of the data will not be detected is approximately 1.9×10^{-7}. Note that 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 utilise it for correction.

A.3.2 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^{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.3.3 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:

1. Initialise C_0 and C_1 to zero

2. 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:

$$C_0 = \Sigma_{i=1}^{N} B_i$$

$$C_1 = \Sigma_{i=1}^{N} (N - i + 1) * B_i$$

3. Calculate an intermediate ISO checksum value as:

$$CK_1 = -(C_0 + C_1)$$

$$CK_2 = C_1$$

4. If $CK_1 = 0$, then $CK_1 = 255$

5. If $CK_2 = 0$, then $CK_2 = 255$

6. Place the resulting values of CK_1 and CK_2 in their destination fields.

A.3.4 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:

1. If either, but not both, checksum octets contains the value zero, then report Error-Detected.

2. Initialise C_0 and C_1 to zero.

3. Process each octet of the data to be checked sequentially from i = 1 to N+2 by

$$C_0 = C_0 + B_i$$

$$C_1 = C_1 + C_0$$

4. 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.3.5 Verification of Compliance

The binary sequences defined in this section 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	1D 0F
00 00 00	CC 9C
AB CD EF 01	04 A2
14 56 F8 9A 00 01	7F D5

A.4 Use of Service Type 2, Device Command Distribution

The following examples show how OLYMPUS Mode 5 commands (distributed via the OBDH Bus) could be accommodated within PUS Service Type 2 (Device Command Distribution):

A. High Level ON/OFF Command, currently 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, Sub-type 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. Memory Load Command, currently 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, Sub-type 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	

Appendix B MISSION PARAMETERS

The following set of telemetry and telecommand related parameters shall be assigned values (where applicable) for each mission:

<ANOM_REPORT_INTERV>
The interval of time over which input data to anomaly detection functions shall be reported within anomaly report packets. The time interval shall be centred on the anomaly time of occurrence. In general, this mission parameter will be specified on a per anomaly basis.

<ANOM_RESP_TIME>
The minimum response time for ground to react to anomalies detected from the telemetry with the generation of a telecommand. Only applicable for short, well-defined intervals during critical mission phases and for pre-agreed contingencies and anomaly conditions.

<APPL_TIME_CODE>
This parameter 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 parameter for each onboard Application Process.

<DIAG_MIN_INTERV>
The minimum sampling interval for the onboard sampling of parameters in Diagnostic mode.

<GRND_RESP_TIME>
The response time for control loops involving the ground. There may be several such parameters for a given mission.

<MISSION_TIME_CODE>
This parameter defines the value of the p-field for the Time 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 fixed status checks which can be specified for the onboard monitoring of a parameter. This mission parameter shall be specified separately for each Application Process which supports onboard parameter monitoring.

<MONLIST_MAX_PARAMS>
The maximum number of parameters which can be monitored at any given time by an onboard 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 parameter shall be specified separately for each Application Process which supports onboard parameter monitoring.

<NUM_SOURCE_BITS>
For telecommand packets, this defines the number of bits of the Sequence Count field which are used to denote the Source of a telecommand packet.

<NUM_SUB_SCHEDS>
The number of distinct sub-schedules supported by an Onboard Command Schedule. Sub-schedules can be independently stopped and started and commands within the same sub-schedule can be interlocked.

<PARAM_ABS_SAMPL_TIME>
The accuracy with which it shall be possible to determine the absolute (onboard) sampling time of a telemetry parameter.

<PARAM_REL_SAMPL_TIME>
The accuracy with which it shall be possible to determine the relative sampling time of any two parameters, which may be telemetred in different packets.

<PKT_RETR_DELAY>
The maximum allowable time delay for the ground to retrieve packets generated at an earlier time and stored onboard. There may be several such mission parameters relating to packets of different priority.

<PKT_STORAGE_TIME>
The time for which telemetry source packets shall be stored onboard, for later dumping to ground, over and above the longest time interval without ground coverage. Only applicable for missions with discontinuous ground coverage.

<PKTS_NUM_STORED>
The number of event-generated packets stored onboard for subsequent retrieval from ground, for missions with continuous ground coverage.

<PSLIST_MAX_PARAMS>
The maximum number of parameters whose statistical values can be evaluated at any given time by an onboard 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 parameter shall be specified separately for each Application Process ID which supports this sub-Service.

<RELATIVE_TIMES_SUPPORTED>
This indicates if commands may be released from the Onboard Scheduling Service on the basis of Relative Time (as opposed to Absolute Time). There may be as many such mission parameters as there are instances of the Onboard Scheduling Service.

<TCPKT_MAX_LENGTH>
The maximum length of a telecommand packet, which may be less than the maximum length defined by the ESA Packet Telecommand Standard.

<TC_VERIF_DELAY>
The maximum delay between the execution of a telecommand and its verification within the telemetry.

<TIME_CORREL_ACCUR>
The accuracy with which onboard time shall be correlated with ground time.

<TIME_INVERS_MAX>

The maximum allowable time inversion between the receipt on ground of the telemetry source packet containing a parameter and the telemetry source packet containing any auxiliary information needed by the ground to process the parameter.

<TMPKT_MAX_LENGTH>

The maximum length of a telemetry source packet, which may be less than the maximum length defined by the ESA Packet Telemetry Standard.

Appendix C GLOSSARY OF TERMS

ADDITIONAL CAPABILITY SET

An Additional Capability Set denotes a set of Service capabilities and/or Service Requests and Reports which may optionally be supported by a given implementation of a Service.

APPLICATION PROCESS

This is an entity which is capable of generating telemetry source packets and receiving telecommand packets and which is uniquely identified by one, and only one, Application Process ID (APID). APIDs may not be dynamically changed during a mission. Provided that these basic requirements are met, an Application Process may be implemented in software, firmware or hardware.

There are also no restrictions on the mapping between Application Processes and the usual functional subdivision of a satellite into subsystems and payloads. In a relatively simple satellite, there may be a centralised Application Process which serves a number of "dumb" platform subsystems and payloads in terms of the collection of housekeeping data, the distribution of device commands, Onboard Scheduling, Onboard monitoring etc. In a more complex satellite, each subsystem and payload might be served by its own independent Application Process.

In a purely software implementation, a given microprocessor may host one or several Application Processes. However, it is also possible that a given Application Process (presumably a rather complex one) could be distributed across two or more microprocessors.

CATASTROPHIC HAZARDOUS TELECOMMAND

A telecommand which, if executed at the wrong time or in the wrong configuration, could cause loss of the space segment or disabling injury or loss of life in the case of a manned mission.

CONFIGURATION-DEPENDENT TELECOMMAND

A telecommand is configuration-dependent if it shall only be sent if a particular configuration of a subsystem or payload prevails, or alternatively, shall ***not*** be sent under certain defined conditions.

CONTROL LOOP

A Control Loop comprises the mechanisms to maintain a parameter, or set of parameters, within prescribed bounds. It is normally constituted 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 must be high and the control response time short, then these control loops are implemented onboard (e.g. attitude control).

CRITICAL HAZARDOUS TELECOMMAND

A telecommand which, if executed at the wrong time or in the wrong configuration, could cause major damage to the space segment leading to significant degradation to the mission or non-disabling personnel injury in the case of a manned mission.

DEVICE TELECOMMAND

A telecommand which is routed to and executed by onboard hardware, e.g. a relay switching telecommand, a telecommand to load an onboard register etc..

HIGH-LEVEL TELECOMMAND

This corresponds to a telecommand issued to an onboard Application Process, which will result in lower-level telecommands being issued to other Application Processes or directly to hardware.

HIGH-LEVEL TELEMETRY

This corresponds to telemetry derived from the low-level telemetry by an onboard Application Process.

LOW-LEVEL TELECOMMAND

This is a telecommand which is issued at a lower level (in the limit, a Device Command).

LOW-LEVEL TELEMETRY

This corresponds to the lowest level of readable onboard information.

MARGINAL HAZARDOUS TELECOMMAND

A telecommand which, if executed at the wrong time or in the wrong configuration, could cause minor damage to the space segment or minor non-disabling injury in the case of a manned mission.

MEMORY

This term is used generically to cover any onboard memory area, whether pure memory or storage memory, such as disk, tape or bubble-memory.

MINIMUM CAPABILITY SET

The minimum capability set denotes the set of Service capabilities and Service Requests and Reports which must be supported by all implementations of a Service.

OPTIONALLY CONFIRMED

All Service User initiated activities defined in this Standard are optionally confirmed. This means that the user invoking an activity indicates which level of acknowledgment (e.g. for telecommand verification) is required for each instance of activity invocation.

PARAMETER

A parameter is the lowest level of elementary data item onboard and, therefore, must have a unique interpretation. Traditionally, each distinct telemetry parameter has been assigned a "Parameter ID" for ground identification purposes. Within this standard, the concept of "Parameter Number (#)" has also been introduced for the onboard "identification" of Parameters; there is a one-to-one correspondence between Parameter ID and Parameter #.

PARAMETER VALIDITY

It may only be meaningful to interpret a given telemetry parameter if certain well-defined conditions prevail. For example, the angular output of a gyro may only have a valid engineering meaning if the power to the gyro is "on"; at other times the output may be random (or at best should not be relied upon). Such a parameter is deemed **Conditionally Valid**, with its **Validity** determined from the power status.

PROTECTION SYSTEM

Those onboard functions (implemented either in hardware or software) which are provided to monitor sensor or logic readings and, based on their outputs, either direct or processed, to reconfigure the satellite system or subsystem into a "safe" configuration. Subsequent analysis and recovery action will normally be performed on the ground.

SATELLITE STATUS

This consists of all the information necessary to assess the operational status of the satellite at a given time, i.e. all the information needed to determine all the criteria driving operational decisions.

SERVICE

A Service denotes a set of onboard functions which provide the conceptually related Service capabilities which can be controlled and monitored by a ground system through a well-defined set of Service Requests and Reports.

SERVICE ACTIVITY

A Service Activity denotes a precise set of actions performed by a Service Provider, whose execution may either be requested by a Service User (**user initiated activity**) or be part of the continuous activities required to accomplish the Service (**continuous activity**).

SERVICE CONCEPT

The Service Concept describes the continuous activities and the state information to be maintained by a Service.

SERVICE PROVIDER

A Service Provider denotes the onboard Application Process which executes the service activities and which is the destination of the Service Requests and the source of the Service Reports.

SERVICE REPORT

A Service Report denotes a 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**.

SERVICE REQUEST

A Service Request denotes a data exchange between a Service User (the request initiator) and a Service Provider to request the execution of a particular activity.

SERVICE USER

The Service User(s) denote the ground system(s) which initiates the Service Requests and receives the Service Reports.

SOFTWARE FUNCTION

An Application Process may perform one or more software **functions** which can be invoked from the ground. This is the "normal" way in which the activities of (i.e. the Services provided by) an Application Process will be invoked. This document defines a number of standard Services. For a given mission, some of these services may be implemented within all Application Processes; some within only one Application Process; others may not be implemented at all. Mission-specific Services may also be implemented; for example to control the operation of a payload or an AOCS.

SOFTWARE TASK

Functions are ultimately achieved by the operation of one or more software **tasks** (usually referred to as processes) running under the control of an Application Process. Under some circumstances, it may be required for the ground to directly control the software tasks.

SUB-SCHEDULE

A distinct sub-set of an Onboard Schedule in the sense that it can be independently started and stopped. Commands within the same sub-schedule can be interlocked to each other.

SUBSYSTEM/PAYLOAD

This is any combination of units within the satellite platform or the payload which fulfils a well-defined and usually self-contained set of onboard functions.

SUPPORTING SERVICE

This is a Service which can only be used 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.

TELECOMMAND

This can be a telecommand packet or a device command sent within a telecommand transfer frame from the ground.

TELECOMMAND FUNCTION

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

VITAL TELECOMMAND

A telecommand which is not hazardous, but which, if not executed at the foreseen time, could cause a significant degradation to the mission.

Appendix D GLOSSARY OF ABBREVIATIONS

ABM	Apogee Boost Motor
ACK	Acknowledgement
AD	Applicable Document
AOCS	Attitude and Orbit Control Subsystem
APID	Application Process ID
ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
CDS	CCSDS Day Segmented
CLCW	Command Link Control Word
CLTU	Command Link Transfer Unit
CPDU	Command Pulse Distribution Unit
COES	Committee for Operations and EGSE Standards
CRC	Cyclic Redundancy Code
CUC	CCSDS Unsegmented Code
EGSE	Electrical Ground Support Equipment
ESA	European Space Agency
FID	Function Identification
H/W	Hardware
HK	Housekeeping (telemetry)
ICD	Interface Control Document
ID	Identification, Identifier
ISO	International Standards Organisation
LEO	Low-Earth Orbit
LEOP	Launch and Early Orbit Phase
LSB	Least Significant Bit
MAP	Multiplexed Access Point
MSB	Most Significant Bit
ms	millisecond
NASA	National Aeronautics and Space Administration
PAC	Packet Assembly Control(ler)
PC	Parameter Code
PCS	Packet Check Sequence
PEC	Packet Error Control
PFC	Parameter Format Code
PSS	Procedures, Specifications, Standards

PTC	Parameter Type Code
PUS	Packet Utilisation Standard
RAM	Random Access Memory
RF	Radio-Frequency
RID	Report Identification
ROM	Read-Only Memory
s	Second
S/C	Spacecraft
S/W	Software
SCS	Satellite Control System
SID	Structure Identification
TBD	To Be Determined
TC	Telecommand
TM	Telemetry
VCi	Virtual Channel i