ECSS-E-ST-70-41C Rev.1 DIR1
11 December 2024

Space engineering

Telemetry and telecommand packet utilization

This document is distributed to the ECSS community for Public Review.

Start of Public Review: 12 December 2024
End of Public Review: 28 February 2025

DISCLAIMER (for drafts)
This document is an ECSS Draft Standard. It is subject to change without any notice and may not be referred to as an ECSS Standard until published as such.

ECSS Secretariat
ESA-ESTEC
Requirements & Standards Section
Noordwijk, The Netherlands

Foreword

ECSS is a cooperative effort of the European Space Agency, national space agencies and European industry associations for the purpose of developing and maintaining common standards. Requirements in this Standard are defined in terms of what shall be accomplished, rather than in terms of how to organize and perform the necessary work. This allows existing organizational structures and methods to be applied where they are effective, and for the structures and methods to evolve as necessary without rewriting the standards.

This Standard has been prepared by the ECSS-E-ST-70-41C Rev.1 Working Group, reviewed by the ECSS Executive Secretariat and approved by the ECSS Technical Authority.

Disclaimer

Published by: ESA Requirements and Standards Section
ESTEC, P.O. Box 299,
2200 AG Noordwijk
The Netherlands

Change log

	Change log for Draft development
Previous steps	
DFR	Draft for Review (DFR) submitted to ES File name: *ECSS-E-ST-70-41C Rev.1 DIR1(30Sep2024) DFR_to_ECSS.docx*
ECSS-E-ST-70-41C Rev.1 DFR1 30 September 2024	Parallel Assessment for release for Public Review by E-70 TAAR and ES. 27 November 2024 – 11 December 2024 Draft released by E-70 TAAR on 3 December 2024
Current step	
ECSS-E-ST-32-02C Rev.2 DIR1 11 December 2024	Public Review 12 December 2024 – 28 February 2025
Next steps	
DIR + impl. DRRs	Draft with implemented DRRs
DIR + impl. DRRs	DRR Feedback
DIA	TA Vote for publication
DIA	Preparation of document for publication (including DOORS transfer for Standards)
	Publication
	Change log for published Standard (to be updated by ES before publication)
ECSS-E-70-41A 30 January 2003	First issue
ECSS-E-70-41B	never issued
ECSS-E-ST-70-41C 15 April 2016	Second issue
ECSS-E-ST-70-41C Rev.1 DFR1 30 September 2024	Second issue, Revision 1 The main purpose of the update of ECSS-E-70-41C (15 April 2026) was the need: • to remove the deficiencies from issue C and to inject lessons learned, • to improve the standard to meet the need for future missions, • to acknowledge the existence of new ECSS and CCSDS standards and to ensure consistency, • to implement the ECSS drafting rules applicable to all ECSS standards • to ensure consistency across the standard with regards to the PUS foundation model The main changes are: • evolution of request verification approach: ST[1] request verification • evolution of HK reporting: ST[3] housekeeping • deletion of service not used in current and future missions: ST[8] function management

	• standarization of services used in current and future missions: ST[26] parameter extraction, ST[27] critical packet log management • standarization of services for File Based Operatios: ST[24] file tranfer, ST[25] file data storage • configuration policy need formalized into requirements for applicable services
ECSS-E-ST-70-41C 15 April 2016	Third issue The main purpose of the update of ECSS-E-70-41C to ECSS-E-ST-70-41C Rev.1 was the need: • to remove the deficiencies from issue C and to inject lessons learned, • to improve the standard to meet the need for future missions, • to acknowledge the existence of new ECSS and CCSDS standards and to ensure consistency, • to implement the ECSS drafting rules applicable to all ECSS standards • to ensure consistency across the standard with regards to the PUS foundation model The main changes are: • evolution of request verification approach: ST[1] request verification • evolution of HK reporting: ST[3] housekeeping • deletion of service not used in current and future missions: ST[8] function management • standarization of services used in current and future missions: ST[26] parameter extraction, ST[27] critical packet log management • standarization of services for File Based Operatios: ST[24] file tranfer, ST[25] file data storage • configuration policy need formalized into requirements for applicable services

Table of contents

Introduction

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

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

1 Scope

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

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

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

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

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

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

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

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

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

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

simplest possible level that also remains sensible and coherent. At least this set is included in every implementation of a given service.

The standardized PUS services fulfil the following criteria:

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

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

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

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

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

2
Normative references

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

ECSS-S-ST-00-01	ECSS system – Glossary of terms
ECSS-E-ST-70	Space engineering – Ground systems and operations
ECSS-E-ST-70-01	Space engineering – Spacecraft on-board control procedures
ECSS-E-ST-70-11	Space engineering – Space segment operability
ECSS-E-ST-70-31	Space engineering – Ground systems and operations – Monitoring and control data definition
CCSDS 133.0-B-2, June 2020	Space Packet Protocol, Blue Book
CCSDS 301.0-B-4, November 2010	Time Code Formats, Blue Book
CCSDS 727.0-B-5, July 2020	File Delivery Protocol, Blue Book
CCSDS 720.1-G-4, May 2021	File Delivery Protocol, Green Book

3
Terms, definitions and abbreviated terms

3.1 Terms from other standards

a. For the purpose of this Standard, the terms and definitions from ECSS-S-ST-00-01 apply, in particular for the following terms:

1. space system
2. space segment
3. spacecraft
4. ground segment

3.2 Terms specific to the present standard

3.2.1 acceptance notification

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

3.2.2 acceptance verification report

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

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

3.2.3 application process

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

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

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

3.2.4 capability

functionality of a service or a subservice

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

3.2.5 data report

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

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

3.2.6 event report

report related to an occurrence of an event

NOTE Event reports are generated by the subservice providers.

3.2.7 execution notification

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

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

3.2.8 execution verification report

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

NOTE The execution reporting subservice for a request is hosted by the application process that hosts the

subservice responsible for executing that request. Each execution verification report is reporting either a successful or a failed execution stage (start, progress or completion) of a request.

3.2.9 instruction

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

3.2.10 message

request or report

3.2.11 notification

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

3.2.12 object path

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

3.2.13 on-board file system

system used to control data organised in files

3.2.14 on-board memory

logical memory space

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

3.2.15 on-board parameter

lowest level of elementary data item on-board

NOTE A parameter has a unique interpretation.

3.2.16 report

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

NOTE This Standard identifies three types of reports:

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

3.2.17 repository path

logical path to where a file or a directory is located

NOTE A repository path can either represent a physical path such as a directory path within a file system or a logical path such as a mounted device (e.g. "/mm1"pointing to a mass memory device), a

directory within a mounted device (e.g. "/mm1/dir1").

3.2.18 request

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

3.2.19 routing notification

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

3.2.20 routing verification report

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

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

3.2.21 service

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

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

3.2.22 subservice

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

3.2.23 subservice entity

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

3.2.24 subservice provider

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

3.2.25 subservice user

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

3.2.26 transaction

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

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

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

3.2.27 verification report

routing, acceptance or execution verification report

3.3 Abbreviated terms

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

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

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

3.4 Nomenclature

The following nomenclature applies throughout this document:

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

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

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

c. The words “may” and “need not” are used in this Standard to express positive and negative permissions, respectively. All the positive permissions are expressed with the word “may”. All the negative permissions are expressed with the words “need not”.

d. The word “can” is used in this Standard to express capabilities or possibilities, and therefore, if not accompanied by one of the previous words, it implies descriptive text.

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

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

4
Context and background

4.1 Introduction

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

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

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

As depicted in Figure 4.1-, it is usually understood that the on-board application processes host the subservice providers and the ground application processes the subservice users but this standard does not constrain those relationships. For example, a ground equipment can host some subservice providers so that the equipment can be remotely controlled by a mission control centre, a payload can host some subservice users for controlling solid-state mass memories (e.g. using file management subservices).

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

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

- an on-board computer can host one or more application processes or an application process can be distributed across two or more on-board computers.

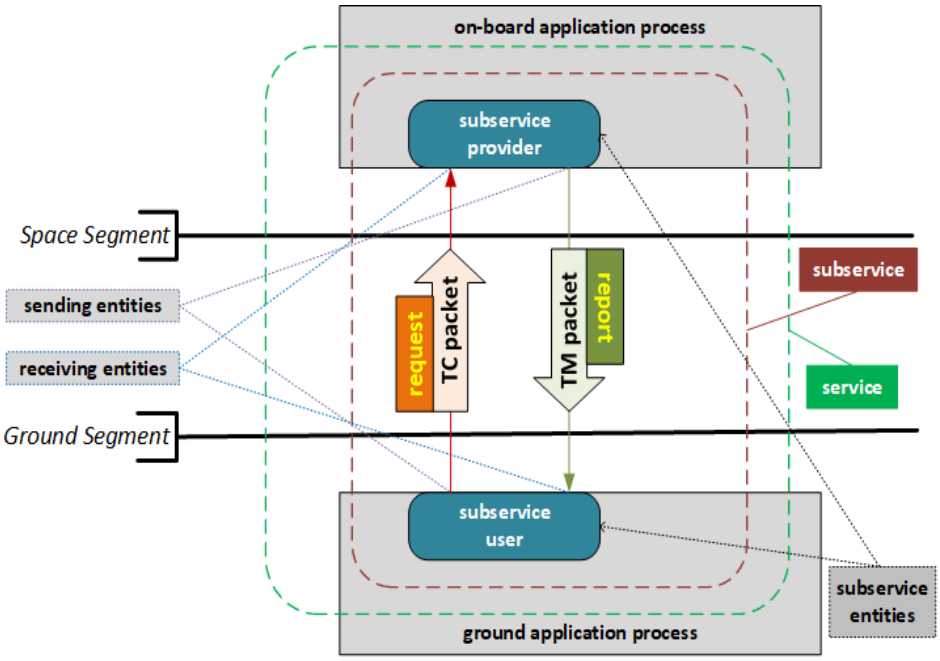

Figure 4.1-1 The space to ground PUS service system context

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

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

Three distinct categories of report are distinguished:

f. the *verification reports*, which report on the routing, acceptance, start, progress and completion of the request execution;

g. the *data reports*, which are generated:

   1. on request, as one or multiple responses to the instructions of a request to elicit some specific service data,

   2. autonomously as one or multiple reports activated by a request or, routinely, i.e. as part of a continuous reporting functionality;

h. the *event reports*, which carry information related to the occurrences of the events detected by a service.

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

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

When messages (requests and reports) are exchanged between ground and space, they are encapsulated into CCSDS space packets, refer to clause 7.3.14.

Figure 4.1-2 provides an example of how PUS services can be deployed on-ground and on-board a spacecraft and how commanding with this Standard is understood.

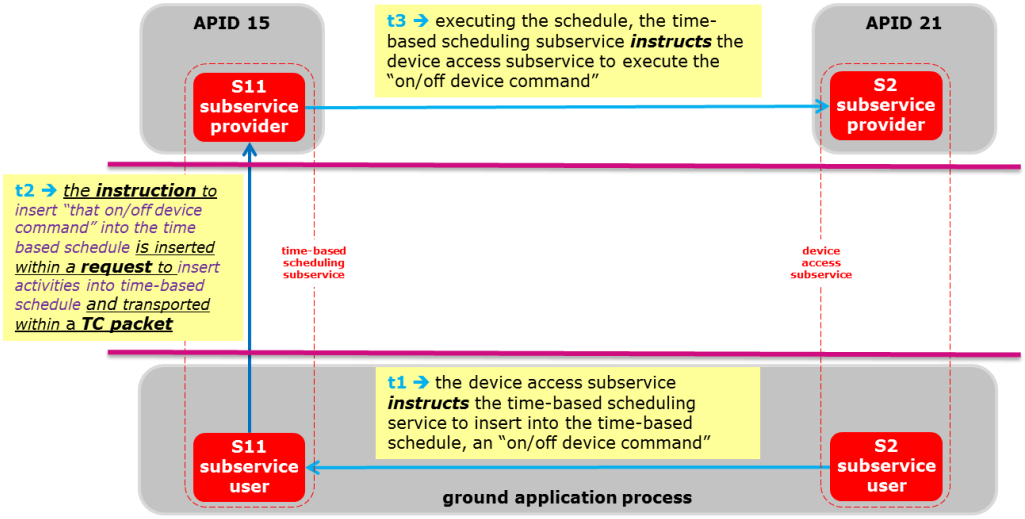

Figure 4.1-2 A PUS utilization example

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

4.2 Modelling the PUS

4.2.1 General

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

The multi-layer model, depicted in Figure 4.2- consists of:

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

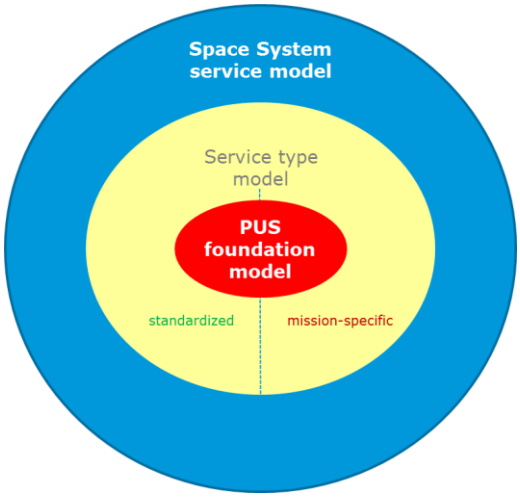

Figure 4.2-1 The PUS model

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

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

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

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

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

4.2.2 The PUS foundation model

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

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

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

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

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

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

The PUS foundation model is specified in clause 5.

4.2.3 The service type model

4.2.3.1 Introduction

The PUS service type model includes:

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

4.2.3.2 Standard service types

This Standard contains the specification of a set of standard PUS service types. The choice of which service types are used by a new mission depends on the mission requirements. All service types are optional and a given service type can be implemented at any of several distinct levels and its parameters and functions can be tailored.

The standard service types are listed in Table 4.2-. They include:

- service types that provide basic functions such as collecting parameter statistics.

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

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

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

Table 4.2-1: The standardized service types

service type	
name	ID
request verification	1
device access	2
housekeeping	3
parameter statistics reporting	4
event reporting	5
memory management	6
(reserved)	7
(reserved)	8
time management	9
(reserved)	10
time-based scheduling	11
on-board monitoring	12
large packet transfer	13
real-time forwarding control	14
on-board storage and retrieval	15
(reserved)	16
Test	17
on-board control procedure	18
event-action	19
parameter management	20
request sequencing	21
position-based scheduling	22
file management	23
file transfer	24
file data storage	25
parameter extraction	26
critical packet log management	27

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

4.2.3.3 Mission-specific service types

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

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

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

4.2.4 The space system service model

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

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

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

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

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

5
The PUS foundation model

5.1 Introduction

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

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

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

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

The generic service type abstraction level specifies:

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

The generic service deployment abstraction level specifies:

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

5.2 Convention

This Standard uses two types of identification mechanisms:

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

Names and identifiers are always unique in a given context.

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

5.3 The generic service type abstraction level

5.3.1 General

a. Each service type shall be uniquely identified by exactly one service type name.

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

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

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

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

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

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

NOTE This Standard promotes the use of the standardized service types and associated specifications to interact with the functionalities provided on-board. The mission-specific service types extensions are used to specify the monitoring and control of the on-board functionalities that are truly mission specific or not accounted for by the standardized service types. The mission-specific extensions cannot be used as alternatives to the standardized service types specifications.

5.3.2 Subservice type

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

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

b. Each subservice type shall be defined by exactly one service type.

c. Each subservice type shall be uniquely identified by exactly one subservice type name.

d. For each subservice type, whether the realization of that subservice type is implicitly required for each realization of the service type or required by tailoring shall be declared when specifying that subservice type.

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

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

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

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

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

NOTE These observables are on-board parameters that are provided by the related subservice, refer for example to the observables of the parameter monitoring subservice in clause 6.12.3.12.

5.3.3 Service type deployment

a. The application process hosting policy used by each service type shall be declared when specifying that service type.

NOTE The application process hosting policy specifies the constraints that need to be fulfilled when deploying the subservice providers of the related services. It specifies, for example, whether all subservice providers of a given service have to be hosted by the same application process or whether they can be hosted by different application processes.

5.3.4 Message type

5.3.4.1 General

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

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

NOTE These identifiers are used in the telemetry packet secondary header (refer to clause 7.4.3.1) and in the telecommand packet secondary header (refer to clause 7.4.4.1) to identify the type of messages transported by these packets but also in specific requests and reports, e.g. in the requests to add report types to the application process forwarding control table (refer to clause 6.14.3.5.1).

c. Each message type identifier shall be composed of:

1. the service type identifier of the service type that contains that message type;
2. a message subtype identifier that uniquely identifies that message type within that service type.

d. Each message subtype identifier shall be an unsigned integer greater than or equal to 1, and less than or equal to 255.

e. Each standard message type identifier shall have a message subtype identifier less than or equal to 127.

NOTE The standard message type identifiers are the identifiers specified in this Standard.

f. Each mission specific message type that belongs to a standard service type shall have a message subtype identifier greater than or equal to 128.

g. Each message type shall either be:

1. a request type, or
2. a report type.

NOTE 1 For item 1, refer to clause 5.3.4.2.

NOTE 2 For item 2, refer to clause 5.3.4.3.

5.3.4.2 Request type

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

NOTE 1 An example of a request type that defines exactly one instruction type is the "modify parameter monitoring definitions" request type specified in clause 6.12.3.8.4. The single related instruction type is the "modify a parameter monitoring definition" instruction type specified in requirement 6.12.3.8.4c.

NOTE 2 An example of a request type that defines more than one instruction type is the "report parameter monitoring definitions" request type specified in clause 6.12.3.9. The related instruction types are specified in requirement 6.12.3.9b, i.e.:

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

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

b. Each instruction type shall be defined for exactly one request type.

c. Each instruction type shall be uniquely identified by exactly one instruction type name.

d. For each request type and for each instruction type of that request type, whether that request type provides a single instruction slot or multiple instruction slots for that instruction type shall be declared when specifying that request type.

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

An example of an instruction type that offers the possibility to have multiple instructions inside a single request is the "report a parameter monitoring definition" instruction type specified in requirement 6.12.3.9b for which the request to "report parameter monitoring definitions" defined in clause 6.12.3.9 provide the capability to have multiple instructions inside a single request.

An example of an instruction type for which it does not make sense to allow multiple instructions in a request is the "report all parameter monitoring definition" instruction type also specified in requirement 6.12.3.9b.

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

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

f. For each instruction type, the instruction arguments used by that instruction type, their definition and their ordering within the instruction type shall be declared when specifying that instruction type.

g. For each request type that provides multiple instruction slots, if that request type constrains the scope of the instructions that can be issued within a request of that type, the argument or set of arguments of the related instruction types that define that scope shall be grouped together in the definition of the request type.

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

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

NOTE A request type argument can be an instruction type argument (or set of instruction type arguments) as specified in requirement 5.3.4.2g, or a directive argument (or set of directive arguments) specifying, for example,

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

5.3.4.3 Report type

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

   NOTE 1 For item 1, an example of a data report type is the housekeeping parameter report type specified in clause 6.3.3.4.

   NOTE 2 For item 2:
   - the verification report types are those specified in clause 6.1, i.e. the request verification service type.
   - the verification reports are used in the request related transactions, refer to clause 5.3.6.2.

   NOTE 3 For item 3, the event report types are those specified in clause 6.5.4, see also clause 5.3.6.4.

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

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

c. Each notification type shall be defined for exactly one report type.

d. Each notification type shall be uniquely identified by exactly one notification type name.

e. For each report type and for each notification type of that report type, whether that report type provides a single notification slot or multiple notification slots for that notification type shall be declared when specifying that report type.

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

   An example of a notification type that offers the possibility to have multiple notifications inside a single report but for which it is explicitly required

to have only one notification per report is the housekeeping parameter report structure report specified in clause 6.3.3.7.

5.3.5 Capability type

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

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

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

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

NOTE The applicability constraint of each standardized capability type is specified in clause 6 (see also 0). For example:

- a "minimum" applicability constraint means that each related subservice provides that capability (see for example Table C-1);
- a "by declaration" applicability constraint means that for each related subservice, whether that capability is provided by that subservice is a decision to take when specifying that subservice (See for example requirement 6.3.3.5.1a);
- an "implied by another capability type" applicability constraint means that if a subservice provides that other capability then that subservice also provides that implied

capability (see for example requirement 6.3.3.5.2a);

- a "by declaration and only if another capability type is provided" applicability constraint means that the decision to include that capability depends on the decision taken for that subservice to provide that other capability (see for example requirement 6.2.5.3a and the associated note).

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

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

5.3.6 Transaction type

5.3.6.1 General

a. Each transaction type shall be defined by exactly one capability type.

b. Each transaction type shall either be:
   1. a request related transaction type,
   2. an autonomous data reporting transaction type, or
   3. an event reporting transaction type.

   NOTE 1 For item 1, refer to clause 5.3.6.2.

   NOTE 2 For item 2, refer to clause 5.3.6.3.

   NOTE 3 For item 3, refer to clause 5.3.6.4.

5.3.6.2 Request related transaction type

5.3.6.2.1 General

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

   NOTE The verification report types introduced in clause 5.3.4.3 are involved in the request related transaction types as a consequence of the execution verification profile specified in clause 5.3.6.2.3.

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

5.3.6.2.2 Response type

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

NOTE 1 An example of a request type that is linked to a data report type is the "report parameter monitoring definitions" request type. The linked data report type, playing the role of the response type, is the "parameter monitoring definition report", refer to requirement 6.12.3.9a.

NOTE 2 As stated in requirement 5.3.4.3b, each data report type defines exactly one notification type. The link that exists between a request type and a report type implies that each instruction type defined by that request type is linked to the notification type defined by that report type.

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

NOTE 1 An example of an instruction type whose realization can cause the generation of multiple notifications is the "report all parameter monitoring definitions" instruction type, refer to requirement 6.12.3.9h.

NOTE 2 An example of an instruction type whose realization causes the generation of a single notification is the "report a parameter monitoring definition" instruction type, refer to requirement 6.12.3.9g.

5.3.6.2.3 Execution verification profile

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

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

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

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

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

specific argument of the instruction to "add a parameter report definition to a parameter functional reporting definition", refer to requirement 1.1.1.1.1a.

c. For each instruction type, the conditions to verify during the execution of each instruction of that type shall be declared when specifying that instruction type.

d. For each instruction type, the post-conditions to verify at the end of the execution of each instruction of that type shall be declared when specifying that instruction type.

e. For each request type, the post-conditions to verify at the end of the execution of each request of that type shall be declared when specifying that request type.

f. For each request type, the execution verification profile used to report the start, progress and completion of execution of each request of that type shall be declared when specifying that request type.

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

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

together with any number of associated parameters whose values are reported to support the processing of that failed execution notification.

g. Each failed execution notification shall indicate at which point of the execution verification profile the request has failed execution.

NOTE This implies for example the identification of the instruction that failed execution. If the code value of the failure notice is not sufficient to unambiguously identify the failure point, that unambiguous identification is realized with the support of the associated parameters.

h. Each progress of execution notification shall provide the means to uniquely identify the request execution step that is notified.

NOTE This identification is used by the subservice user that has initiated the execution of that request.

i. For each instruction type, the functionality that the subservice performs when executing an instruction of that type shall be declared when specifying that instruction type.

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

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

5.3.6.3 Autonomous data reporting transaction type

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

NOTE 1 Examples of autonomous data report types are:

- the housekeeping parameter report type (refer to clause 6.3.3.4),
- the check transition report type (refer to clause 6.12.3.7).

NOTE 2 It is noted that some data reports can be generated autonomously but also in response to specific requests. This is for example the case of the housekeeping parameter reports that can be generated periodically according to a collection interval (refer to requirement 6.3.3.3c), but are also generated as the response of a request to generate a one shot report for housekeeping parameter report structures (refer to clause 6.3.3.8).

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

5.3.6.4 Event reporting transaction type

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

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

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

The message subtype identifier gives the severity level of the event report types, refer to clause 6.5.4. For example, all event reports for low severity events have the same message type. i.e. the same combination of service type identifier and message subtype identifier. There is no means, at event report type level, to identify the event that is associated to the related event reports. For that event association, this Standard defines the concept of event definitions. Each event definition is associated to a single event and a single event report type. Each event definition is uniquely identified by the combination of the application process that generates the corresponding event reports and an event definition identifier that is unique within the context of that application process (refer to clause 6.5.3).

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

5.3.7 Tailoring the generic service type abstraction level

a. Tailoring the generic service type abstraction level shall consist of:

1. adding mission-specific service types;
2. adding mission-specific subservice types;
3. adding mission-specific capability types;
4. adding mission-specific message types.

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

5.4 The generic service deployment abstraction level

5.4.1 Introduction

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

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

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

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

5.4.2 Application process

5.4.2.1 General

a. Each application process shall either be:
   1. an on-board application process, or
   2. a ground application process.

b. Each application process that hosts at least one subservice provider shall be identified by an application process identifier that is unique across the system that hosts that application process.

   NOTE The systems introduced in this requirement are either ground segment related or space segment related. The application process identifiers uniquely identify the application processes within a system. This Standard does not further elaborate on this system concept and how the application processes are identified across the full space system.

c. Each application process identifier shall be an unsigned integer that is less than or equal to 65535.

   NOTE The application process identifiers are used to identify the ground application processes and the application processes on-board a spacecraft.

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

   NOTE 1 The CCSDS space packet protocol reserves 11 bits for the identification of the on-board application processes. These 11 bits are used to identify the on-board destination of the requests and the on-board source of the reports.

NOTE 2 The APID 2047 is reserved for idle packets, specified in CCSDS 133.0-B-2. The APID 0 is reserved for spacecraft time packets. Other APID values are reserved, refer to the Space Assigned Numbers Authority Registry – SANA, in the bibliography clause, at the end of this document.

e. Each ground application process identifier shall be an unsigned integer that is greater or equal to 2048 and less that or equal to 65535.

f. No application process identifier used to identify a ground application process shall be used to identify an on-board application process.

NOTE 1 This Standard makes use of the application process identifiers to identify the application process that sources a message (a request or a report) and the application process targeted by that message. These messages can be exchanged between a ground application process and an on-board application process, or between two on-board application processes or two ground application processes. The segregation between ground and space application process identifiers ensures the unique identification of the involved (ground and on-board) application processes.

NOTE 2 The same application process identifier can be used by different space segment related systems. This enables, for example, the possibility to reuse the same on-board application processes for different spacecraft of the same constellation.

g. Each on-board application process shall have access to the spacecraft reference time.

NOTE For the spacecraft reference time, refer to clause 5.4.3.2 and clause 6.9.3.

h. If the spacecraft supports the capability to provide the spacecraft time reference status, each on-board application process shall have access to the spacecraft time reference status.

NOTE For the spacecraft time reference status, refer to requirement 5.4.3.2b.

i. Each application process user identifier shall be an unsigned integer that is greater than or equal to 0, and less than or equal to 65535.

j. For each report that it generates, each on-board application process shall time tag that report using the spacecraft reference time.

k. For each application process, whether that application process time tags the reports before collecting the values of the constituting parameters or after shall be declared when specifying that application process.

NOTE When a report contains parameter values acquired at different times (e.g. housekeeping reports with multiple samples of the same parameter), the

acquisition time of each set of parameter values can be deduced from the time tag of the report.

l. For each application process, whether that application process provides the capability to report the status of the spacecraft time reference used when time tagging reports shall be declared when specifying that application process.

m. For each application process, whether that application process provides the capability to count the type of generated messages per destination and report the corresponding message type counter shall be declared when specifying that application process.

n. Each application process that provides the capability to count the type of generated messages per destination and report the corresponding message type counter shall maintain, per destination, a counter for each message type that it generates.

5.4.3 Interfaced system objects

5.4.3.1 Introduction

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

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

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

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

The system objects introduced in this Standard are:

- the on-board reference clock, refer to clause 5.4.3.2;
- the on-board parameters, refer to clause 5.4.3.3;
- the on-board memories, refer to clause 5.4.3.4;
- the virtual channels, refer to clause 5.4.3.5;
- the on-off devices, refer to clause 6.2.4;
- the registries, refer to clause 6.2.5;
- the CPDUs, refer to clause 6.2.6 and clause 9;
- the physical and the logical devices, refer to clause 6.2.7.1.1 and clause 6.2.7.2.1;
- the housekeeping parameter report structures, refer to clause 6.3.3.2;
- the parameter functional reporting definitions, refer to clause 6.3.5.2;

- the event definitions, refer to clause 6.5.3;
- the functions, refer to clause **Error! Reference source not found.**;
- the time-based sub-schedules, refer to clause 6.11.5.1;
- the time-based scheduling groups, refer to clause 6.11.6.1;
- the parameter monitoring definitions, refer to clause 6.12.3.3;
- the functional monitoring definitions, refer to clause 6.12.4.2;
- the packet stores, refer to clause 6.15.3.1;
- the on-board control procedures, refer to clause 6.18.4.1;
- the request sequences, refer to clause 6.21.4;
- the position-based sub-schedules, refer to clause 6.22.7.1;
- the position-based scheduling groups, refer to clause 6.22.8.1;
- the on-board file systems, refer to clause 5.4.5.
- the CFDP entity, refer to clause 6.24.4.2

5.4.3.2 On-board reference clock

a. The spacecraft shall host an on-board reference clock that provides the spacecraft time reference.

NOTE This Standard does not specify how the reference clock is implemented on-board. It can, for example, be a free running counter or a clock synchronized to a GPS receiver.

b. Whether the spacecraft supports the capability to provide the spacecraft time reference status shall be declared when specifying the spacecraft architecture.

c. If the spacecraft supports the capability to provide the spacecraft time reference status, the list of spacecraft reference statuses that can be reported by the spacecraft shall be declared when specifying the spacecraft architecture.

5.4.3.3 On-board parameter

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

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

NOTE 2 A service may need to acquire a reading of an on-board parameter for the purposes of its routine activity (for example, to monitor its value, to use its value to determine the validity of another on-board parameter, to use its value in a calculation etc.).

NOTE 3 The "baseline" set of on-board parameters is defined during the spacecraft design process. However, the flexibility can also exist to define new parameters in

orbit or to change the definition of an existing on-board parameter or to set the value of an on-board parameter (refer to clause 6.20). This capability is of course restricted to software parameters held in on-board memory and the on-board software design can additionally have built-in protections to ensure against the overwriting of essential on-board parameters.

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

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

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

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

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

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

5.4.3.4 On-board memory

5.4.3.4.1 General

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

NOTE 1 The on-board memory concept introduced in this Standard is for logical memories, i.e. any logical memory space, potentially managed by different on-board processors. The mapping with physical memories is out of the scope of this Standard.

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

alignment constraint that specifies the minimum number of bytes used by the services to address the corresponding physical memory.

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

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

b. At any time, each on-board memory identifier shall uniquely identify exactly one on-board memory that is unique across the entire spacecraft.

c. For each on-board memory, the following characteristics of that memory shall be declared when specifying that memory:

1. the memory access alignment constraint;
2. the memory size, in bytes;
3. the allowed operations;
4. the addressing scheme.

NOTE For item 4, refer to clause 5.4.3.4.2.

d. When declaring the characteristics of an on-board memory, the allowed operations shall be one of the following:

1. "read only";
2. "read and write";
3. "write only".

e. For each on-board memory, whether scrubbing that memory is supported shall be declared when specifying that memory.

f. For each on-board memory, whether write protecting that memory is supported shall be declared when specifying that memory.

5.4.3.4.2 Addressing scheme

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

b. Absolute addressing implies that the memory addresses and related offsets shall be expressed in bytes.

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

NOTE Base plus offset addressing means that the memory addresses are byte offsets from a base reference. A base reference gives (explicitly or implicitly) the address within the memory which is used as the

byte-zero reference for the offset. The base reference can itself be an absolute address or a symbolic address e.g. the name of a table, a parameter set or a file whose absolute address is implicitly known on-board.

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

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

5.4.3.5 Virtual channel

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

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

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

5.4.4 Checksum algorithm

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

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

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

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

5.4.5 On-board file system

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

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

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

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

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

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

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

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

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

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

g. For each on-board file system, the concurrent access policy that it is provided shall be declared when specifying that file system.

h. Each on-board file system that provides concurrent access to files shall ensure the integrity of the files managed.

NOTE The concurrent access policy defines, for example, the maximum number of files that the file system can access at a given time, whether the file system provides concurrent multiple accesses to a specific file.

5.4.6 Service

a. Each service shall be of exactly one service type.

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

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

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

NOTE An example of a subservice type whose realization is required by tailoring is the functional monitoring subservice type of the on-board monitoring service type, refer to requirement 6.12.2.1.2a.

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

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

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

NOTE The service topology includes:

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

5.4.7 Subservice

5.4.7.1 General

a. Each subservice shall be of exactly one subservice type.

b. Each subservice shall belong to exactly one service.

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

5.4.7.2 Subservice entity

5.4.7.2.1 General

a. Each subservice entity shall belong to exactly one subservice.

b. Each subservice entity shall be hosted by exactly one application process.

c. Each subservice entity shall be either a subservice user or a subservice provider.

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

5.4.7.2.2 Subservice provider

a. Each subservice shall provide exactly one subservice provider.

NOTE A subservice provider is an operational element of a subservice that is in charge of execution of the subservice requests and generation of the subservice reports. The subservice providers are usually hosted by the on-board application processes.

5.4.7.2.3 Subservice user

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

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

5.4.8 Capability

a. Each subservice shall provide at least one subservice capability.

b. For each subservice and for each capability type defined by the corresponding subservice type, the inclusion of the related capability in that subservice shall comply with the applicability constraints of that capability type.

NOTE For the applicability constraints of a capability type, refer to requirement 5.3.5b.

5.4.9 Failed progress of execution

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

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

5.4.10 Transaction

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

NOTE A transaction is either:

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

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

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

A request related transaction involves:

- a request,
- depending on the acknowledgement specified for that request (refer to clause 5.4.11.2.2) and the execution verifications of that request (refer to clause 5.4.11.2.3), zero or more verification reports,
- depending on the successful execution of the instructions contained within that request, if that request type is linked to a response type, one or more responses (refer to clause 5.3.6.2).

An autonomous data reporting transaction involves an autonomous data report.

An event reporting transaction involves an event report.

5.4.11 Message

5.4.11.1 General

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

NOTE The message type is specified in clause 5.3.4. A message is either a request or a report.

5.4.11.2 Request

5.4.11.2.1 General

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

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

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

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

NOTE 3 Once a request is issued by an application process, changing the content of that request is not allowed.

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

c. Each request shall be uniquely identified by a request identifier that is the combination of:

1. the application process identifier that hosts the subservice user that generates the request;
2. the application process identifier that hosts the subservice provider that executes the request;
3. the sequence count or request name that is produced by the sourcing application process.

NOTE 1 This Standard assumes that the request identifier is unique for the mission duration but does not further elaborate on how this uniqueness is achieved. In reality, it can happen that the same identifier is used for several requests, e.g. during tests or when the sequence count counter wraps around, implying the need to include timing information to ensure the uniqueness of request identification for the overall mission duration.

NOTE 2 This requirement makes the assumption that the application process identifiers are sufficient to uniquely identify each application process across the overall space segment. If the same application process identifier can be used by different ground segment or space segment related systems, the use of the system identifiers is required to uniquely identifies the application processes.

NOTE 3 When a request is transported within a CCSDS telecommand packet, refer to clause 7.3.14:

- the application process identifier of the destination identifier is set in the application process identifier field of the packet identification field of the packet primary header field;
- the sequence count or request name is set in the packet sequence count or packet name field of the packet sequence control field of the packet primary header field;
- the source identifier is set in the source identifier field of the packet secondary header field.

d. Each request shall be of exactly one request type.

e. Each request whose request type provides a single instruction slot shall contain exactly one instruction that is of an instruction type defined for that request type.

f. Each request whose request type provides multiple instruction slots shall contain an ordered list of one or more instructions, each one being of an instruction type defined for that request type.

NOTE For example, the request to "enable event-action definitions" can include either a single instruction to "enable all event-action definitions" or one or more instructions to "enable an event-action definition", refer to requirement 6.19.7.1b.

5.4.11.2.2 Acknowledgement

a. Each request shall contain:

1. a flag indicating whether the reporting of the successful acceptance of that request by the destination application process is requested;
2. a flag indicating whether the reporting of the successful start of execution of that request by the destination application process is requested;
3. a flag indicating whether the reporting of the successful progresses of execution of that request by the destination application process is requested;
4. a flag indicating whether the reporting of the successful completion of execution of that request by the destination application process is requested.

NOTE 1 Related to item 1:

- each successful acceptance is only reported if that flag indicates such reporting need, refer to requirement 6.1.4.2d;
- each failed acceptance is reported by the destination application process, refer to requirement 6.1.4.3f.

NOTE 2 For item 2:

- each successful start of execution is only reported if the item 2 flag indicates the reporting need, refer also to requirements 5.4.11.2.3a.2 and 6.1.5.1.1b;
- each failed start of execution is notified by the subservice provider in charge of executing that request and reported by the destination application process that hosts that subservice provider, refer to requirements 5.4.11.2.3a.1 and 6.1.5.1.2b.

NOTE 3 For item 3:

- each successful progress of execution is only reported if the item 3 flag indicates the reporting need, refer also to requirements 5.4.11.2.3a.3(c) and 6.1.5.2.1b;
- each failed progress of execution is notified by the subservice provider in charge of executing that request, refer to requirement 5.4.11.2.3a.3(b);
- depending on the subservice provider's request type related failed progress of execution notifications reporting policy (refer to requirement 5.4.9a), the failed progress of execution notifications are reported by the destination application process that hosts that subservice provider within failed progress of execution verification reports (refer to

requirement 6.1.5.2.2b) or as part of the completion of execution verification report for the related request (refer to requirement 6.1.5.3.2b).

NOTE 4 For item 4:

- each successful acceptance is only reported if the item 4 flag indicates the reporting need, refer also to requirements 5.4.11.2.3a.4(c) and 6.1.5.3.1b.;
- each failed completion of execution is notified by the subservice provider in charge of executing that request and reported by the destination application process that hosts that subservice provider, refer to requirements 6.1.5.3.2b and 6.1.5.3.2b.

NOTE 5 the reporting need of successful routing is not considered

5.4.11.2.3 Request execution verification

a. For each request that it receives, the subservice provider in charge of the execution of that request shall, in sequence:
   1. if the pre-conditions for the execution of that request are not fulfilled:
      (a) notify the execution reporting subservice of its parent application process of the failed start of execution;
      (b) stop processing that request;
   2. if the pre-conditions for the execution of that request are fulfilled, notify the execution reporting subservice of its parent application process of the successful start of execution;
   3. for each step, if any:
      (a) verify the execution conditions of that step, if any;
      (b) if the execution conditions of that step are not fulfilled, notify the execution reporting subservice of its parent application process of the failed progress of execution of that step;
      (c) if the step's execution conditions are fulfilled, notify the execution reporting subservice of its parent application process of the successful progress of execution of that step;
   4. at the end of the execution of that request:
      (a) verify the post-conditions of execution, if any;
      (b) if any step execution has failed or if the post-conditions of execution are not fulfilled, notify the execution reporting subservice of its parent application process of the failed completion of execution and stop processing that request;

(c) if the post-conditions of execution are fulfilled, notify the execution reporting subservice of its parent application process of the successful completion of execution;

NOTE 1 A successful completion of execution notification means only that the subservice provider has checked all post-conditions defined in the execution verification profile of that request. It does not necessarily mean that the request execution is successful. That meaning depends on the execution verification profile.

NOTE 2 A failed completion of execution notification is generated for requests that are aborted during their execution.

5.4.11.3 Report

5.4.11.3.1 General

a. Each report shall be generated by exactly one subservice provider.

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

NOTE 2 Once a report is issued by the application process, changing the content of that report is not allowed.

b. Each report shall be addressed to exactly one subservice user.

NOTE The subservice user addressed by this requirement is the final destination. This Standard does not address e.g.:

- the possibility for a report to be forwarded via different paths to its final destination,
- in case e.g. of event reports, the possibility to dispatch the report on-board,
- the possibility for having more than one ground application processing the report.

c. Each report shall be uniquely identified by a report identifier that is the combination of:

1. the application process identifier that hosts the subservice user that generates the report;
2. the application process identifier that hosts the subservice provider that processes the report;
3. the source sequence count that is produced by the application process that sources the report.

NOTE 1 This Standard assumes that the report identifier is unique for the mission duration but does not further elaborate on how this uniqueness is achieved. In reality, it can happen that the same identifier is used for several requests, e.g. during tests or when the sequence count counter wraps

around, implying, for example, the need to include timing information to ensure the uniqueness of report identification for the overall mission duration.

NOTE 2 This requirement makes the assumption that the application process identifiers are sufficient to uniquely identify each application process across the overall space segment. If the same application process identifier can be used by different ground segment or space segment related systems, the use of the system identifiers is required to uniquely identifies the application processes.

NOTE 3 When a report is transported within a CCSDS telemetry packet, refer to clause 7.3.14:

- the source identifier is set in the application process identifier field of the packet identification field of the packet primary header field;
- the sequence count is set in the packet sequence count or packet name field of the packet sequence control field of the packet primary header field;
- the destination identifier is set in the destination identifier field of the packet secondary header field.

d. Each report shall be of exactly one report type.

e. Each report whose report type provides a single notification slot shall contain exactly one notification that is of a notification type defined for that report type.

f. Each report whose report type provides multiple notification slots shall contain an ordered list of one or more notifications, where:

1. all notifications in the list are of the same notification type, and
2. that notification type is one of those defined for that report type.

5.4.11.3.2 Response

a. The destination of any response shall be the source of the corresponding request.

b. If a request implies the generation of a response that exceeds the length that can be carried in a telemetry packet of the maximum packet size, the response shall be split into multiple self contained PUS telemetry packets.

c. The criteria used by a subservice for splitting their responses into multiple self-contained PUS telemetry packets shall be declared when specifying that subservice.

d. Related user data split in multiple telemetry packets shall be logically grouped by making use of the sequence flags in bits 16 and 17 of the CCSDS Packet Primary Header as follows:

(a) 01 for the first packet of a group;

(b) 00 for a continuing packet of a group;

(c) 10 for a last packet of a group.

e. For any packet not belonging to a group, the sequence flags shall be set to 11.

NOTE 1 The cases where user data cannot fit into a single telemetry packet are expected to be limited, e.g. memory dumps or detailed content reports.

NOTE 2 For most telemetry packets, sequence flags are expected to be set to 11.

5.4.11.3.3 Data report

a. For each data report that can be generated in an autonomous data reporting transaction, the destination of the data report in that case shall be declared when specifying the related subservice.

5.4.12 Building the space system architecture

a. Deploying the service topology of an overall space system should consist of:

1. specifying new implementations of PUS services by instantiating the service types and related components;
2. assessing the adequacy of reusing existing service implementations:
   (a) ensuring their compliance to the PUS standard services;
   (b) verifying their compliance to the overall system constraints.

6
Service type system requirements

6.1 ST[01] request verification

6.1.1 Scope

6.1.1.1 General

The request verification service type concerns:

- each application process that is involved in the routing of requests to the application processes responsible for their execution, and
- for each request, the application process responsible for its execution, i.e. the application process that hosts the service that executes the request.

The request verification service type provides the capability for:

- checking that a request received on-board has not been corrupted during the ground to space uplink;
- checking the availability of the application process that is the destination for that request;
- checking the availability of the service that executes that request;
- reporting the success or failure of these checks;
- generating the execution request verification reports on behalf of the service that executes that request.

The request verification service type defines three standardized subservice types, i.e.:

- the routing and reporting subservice type,
- the acceptance and reporting subservice type,
- the execution reporting subservice type.

6.1.1.2 Routing and reporting subservice

The routing and reporting subservice type provides the capability to check that the conditions required to pursue the routing of a request are fulfilled. This includes checking the integrity of the request during its routing to the application process that is responsible for executing it.

This subservice type provides the means to report, to the ground, the failure of request routing.

This Standard assumes that the subservices of type "routing and reporting" (one or more depending on the on-board architecture) are part of the function that performs the on-board routing and as such, each one is hosted by an application process entity that routes requests on-board to their final destination. The request routing logic and related architecture is not further elaborated in this Standard.

6.1.1.3 Acceptance and reporting subservice

Each subservice of type "acceptance and reporting" is hosted by an application process that hosts subservice providers responsible for executing requests.

The acceptance and reporting subservice type provides the capability to check the acceptance of a request prior to its distribution to the service addressed by that request. This subservice type provides the means to report the successful or failed acceptance of each received request.

6.1.1.4 Execution reporting subservice

Each subservice of type "execution reporting" is similarly hosted by an application process that hosts subservice providers responsible for executing requests. It receives the request execution notifications issued by the subservice providers and provides the means to generate the corresponding execution verification reports on behalf of those subservice providers.

Each request execution notification indicates a request execution stage, which can be a start of execution, a progress of execution or a completion of execution. The notification also indicates whether that execution stage succeeded or failed and, in case of failure, the reason for such failure.

6.1.2 Service layout

6.1.2.1 Subservice

a. Each request verification service shall contain at least one of the following:
   1. one or more routing and reporting subservices,
   2. one or more acceptance and reporting subservices,
   3. one or more execution reporting subservices.

   NOTE 1 This Standard does not impose that a single service is used for all verification reports of a request. For example, the routing verification reports generated for a request can be issued by different request verification subservices of different request verification services (e.g. one associated to the platform and one associated to a payload).

   NOTE 2 The routing and reporting subservice deployment results from the spacecraft architecture. The on-board routing of a request can involve several routing and reporting subservices, each one performing specific routing verification checks.

   NOTE 3 The acceptance verification reports can only be issued by the acceptance and reporting subservice

hosted by the application process that executes the request.

NOTE 4 The execution verification reports can only be issued by the execution reporting subservice that is hosted by the application process that executes the request.

6.1.2.2 Application process

6.1.2.2.1 Destination of verification reports

a. For each verification report that it generates, the application process shall address that report to the application process that hosts the subservice user that has generated the corresponding request.

NOTE The destination of the report corresponds to the source identifier of the corresponding request, refer to requirement 5.4.11.2.1.c.

6.1.2.2.2 Application process that routes requests

a. Each application process that is involved in routing requests shall host exactly one routing and reporting subservice.

NOTE This Standard does not preclude that the requests that are addressed to the application process that hosts that routing and reporting subservice are also checked by that subservice.

6.1.2.2.3 Application process that executes requests

a. Each application process that hosts one or more subservices that execute requests shall host:

1. exactly one acceptance and reporting subservice;
2. at most one execution reporting subservice.

NOTE The decision to implement the execution reporting subservice is not an application process decision but a decision that is derived from the operational needs of the services that execute the requests received by the application process.

6.1.3 Routing and reporting subservice

6.1.3.1 Accessibility

6.1.3.1.1 Application process

a. The list of application processes that the routing and reporting subservice addresses shall be declared when specifying the spacecraft architecture.

6.1.3.2 Routing verification of a request

a. The routing and reporting subservice shall provide the capability to perform routing verification for the requests that it receives.

b. The list of routing verification checks that the routing and reporting subservice performs shall be declared when specifying that subservice.

NOTE 1 Depending on the spacecraft architecture, the routing of a request can involve several routing and reporting subservices. The routing verification checks can be distributed in accordance.

NOTE 2 The routing and reporting subservice can, for example check:

- that the request has not been corrupted;
- the existence of the destination;
- the readiness of this destination to receive the request, e.g. the device which embeds that destination is on;
- the ability to continue the routing of the request.

c. For each request that it receives, the routing and reporting subservice shall:

1. perform the routing verification checks on that request;
2. determine, based on the output of those checks, whether the routing verification of that request has succeeded or failed.

6.1.3.3 Reporting failed routing

a. The routing and reporting subservice shall provide the capability to report the failed routing of requests.

NOTE The corresponding verification reports are of message type "TM[1,10] failed routing verification report".

b. Each failed routing verification report shall contain exactly one failed routing notification.

c. Each failed routing notification shall contain:

1. the identifier of the request that failed the routing verification;
2. the failure notice made of:
   (a) a failure code;
   (b) auxiliary data, if any, used to explain the reason for the failed routing.

NOTE For item 2, see requirements 6.1.3.3d and 6.1.3.3e.

d. The list of failure codes defined for failed routing notifications shall be declared when specifying the routing and reporting subservice.

NOTE The failed routing notification failure codes are common to all requests that are routed by the subservice.

e. For each failure code defined for failed routing notifications, the associated auxiliary data shall be declared when specifying the routing and reporting subservice.

f. For each request that fails its routing verification, the routing and reporting subservice shall:

1. generate a single failed routing notification and associated report for that request;
2. discard that request.

6.1.3.4 Subservice observables

a. The following observables shall be defined for the routing and reporting subservice

1. the number of requests that routing verification succeeded;
2. the number of requests that routing verification failed.

6.1.4 Acceptance and reporting subservice

6.1.4.1 Acceptance verification of a request

a. The acceptance and reporting subservice shall provide the capability to perform acceptance verification for a request that it receives.

b. The list of acceptance verification checks that the acceptance and reporting subservice performs during the acceptance verification of a request shall be declared when specifying that subservice.

NOTE The acceptance and reporting subservice can, for example, check:

- that the request has not been corrupted;
- the availability of the service.

c. For each request that it receives, the acceptance and reporting subservice shall:

1. perform the acceptance verification checks on that request;
2. determine, based on the output of those checks, whether the acceptance verification of that request has succeeded or failed.

6.1.4.2 Reporting successful acceptance

a. The acceptance and reporting subservice shall provide the capability to report the successful acceptance verification of requests.

NOTE The corresponding verification reports are of message type "TM[1,1] successful acceptance verification report".

b. Each successful acceptance verification report shall contain exactly one successful acceptance notification.

c. Each successful acceptance notification shall contain:

1. the identifier of the request that successfully passed the acceptance verification.

d. For each request that successfully passes its acceptance verification, the acceptance and reporting subservice shall:

1. if the successful acceptance reporting is requested, generate a single successful acceptance notification and associated report for that request.

NOTE For the successful acceptance reporting, refer to requirement 5.4.11.2.2a.1.

6.1.4.3 Reporting failed acceptance

a. The acceptance and reporting subservice shall provide the capability to report the failed acceptance of requests.

NOTE The corresponding verification reports are of message type "TM[1,2] failed acceptance verification report".

b. Each failed acceptance verification report shall contain exactly one failed acceptance notification.

c. Each failed acceptance notification shall contain:

1. the identifier of the request that failed the acceptance verification;
2. the failure notice made of:
   (a) a failure code;
   (b) auxiliary data, if any, used to explain the reason for the failed acceptance.

NOTE For item 2, see requirements 6.1.4.3d and 6.1.4.3e.

d. The list of failure codes defined for failed acceptance notifications shall be declared when specifying the acceptance and reporting subservice.

NOTE The failure codes used by the subservice to notify failed acceptance are not request dependent.

e. For each failure code defined for failed acceptance notifications, the associated auxiliary data shall be declared when specifying the acceptance and reporting subservice.

f. For each request that fails its acceptance verification, the acceptance and reporting subservice shall:

1. generate a single failed acceptance notification and associated report for that request;
2. discard that request.

6.1.4.4 Subservice observables

a. The following observables shall be defined for the acceptance and reporting subservice:

1. the number of requests that acceptance verification succeeded;
2. the number of requests that acceptance verification failed.

6.1.5 Execution reporting subservice

6.1.5.1 Reporting the start of execution of a request

6.1.5.1.1 Reporting successful start of request execution

a. The execution reporting subservice shall provide the capability to generate the successful start of request execution verification reports.

NOTE The corresponding verification reports are of message type "TM[1,3] successful start of request execution verification report".

b. For each successful start of request execution notification that it receives, the execution reporting subservice shall:

1. if the successful start of request execution reporting is requested, generate a single successful start of request execution verification report containing that notification.

NOTE 1 For the successful start of request execution notification, refer to requirement 5.3.5.2.3g.

NOTE 2 For the requested successful start of request execution reporting, refer to requirement5.4.11.2.2a.2.

6.1.5.1.2 Reporting failed start of request execution

a. The execution reporting subservice shall provide the capability to generate the failed start of request execution verification reports.

NOTE The corresponding verification reports are of message type "TM[1,4] failed start of request execution verification report".

b. For each failed start of request execution notification that it receives, the execution reporting subservice shall:

1. generate a single failed start of request execution verification report containing that notification.

NOTE For the failed start of request execution notification, refer to requirement 5.3.5.2.3g.

6.1.5.2 Reporting the progress of execution of a request

6.1.5.2.1 Reporting successful progress of request execution

a. The execution reporting subservice shall provide the capability to generate the successful progress of request execution verification reports.

NOTE The corresponding verification reports are of message type "TM[1,5] successful progress of request execution verification report".

b. For each successful progress of request execution notification that it receives, the execution reporting subservice shall:

1. If the successful progress of request execution reporting is requested, generate a single successful progress of request execution verification report containing that notification.

NOTE 1 For the successful progress of request execution notification, refer to requirement 5.3.5.2.3g.

NOTE 2 For the requested successful progress of request execution reporting, refer to requirement 5.4.11.2.2a.3.

6.1.5.2.2 Reporting failed progress of request execution

a. The execution reporting subservice shall provide the capability to generate the failed progress of request execution verification reports.

NOTE The corresponding verification reports are of message type "TM[1,6] failed progress of request execution verification report".

b. For each failed progress of request execution notification that it receives, the execution reporting subservice shall:

1. if the application process that hosts the execution reporting subservice is configured for the corresponding request type to report the failed progress of request execution notifications in failed progress of execution verification reports, generate a single failed progress of request execution verification report containing that notification.

NOTE 1 For the failed progress of request execution notification, refer to requirement 5.3.5.2.3g.

NOTE 2 For item 1 failed progress of request execution notifications reporting policy, refer to requirement 5.4.9a. See also requirement 6.1.5.3.2c for the alternative handling of the failed progress of execution notifications.

6.1.5.3 Reporting the completion of execution of a request

6.1.5.3.1 Reporting successful completion of request execution

a. The execution reporting subservice shall provide the capability to generate the successful completion of request execution verification reports.

NOTE The corresponding verification reports are of message type "TM[1,7] successful completion of request execution verification report".

b. For each successful completion of request execution notification that it receives, the execution reporting subservice shall:

1. if the successful completion of request execution reporting is requested, generate a single successful completion of request execution verification report containing that notification.

NOTE 1 For the successful start of request execution notification, refer to requirement 5.3.5.2.3g.

NOTE 2 For the requested successful completion of request execution reporting, refer to requirement 5.4.11.2.2a.4.

6.1.5.3.2 Reporting failed completion of request execution

a. The execution reporting subservice shall provide the capability to generate the failed completion of request execution verification reports.

NOTE The corresponding verification reports are of message type "TM[1,8] failed completion of request execution verification report".

b. For each failed completion of request execution notification that it receives, the execution reporting subservice shall:

1. generate a single failed completion of request execution verification report containing that notification.

NOTE For the failed completion of request execution notification, refer to requirement 5.3.5.2.3g.

c. For each failed completion of request execution notification that is accompanied of failed progress of request execution notifications to be reported as part of the completion of request execution verification report, the execution reporting subservice shall include those failed progress of request execution notifications in the failed completion of request execution notification.

NOTE For the failed progress of request execution notifications reporting policy refer to requirement 5.4.9a.

6.1.5.4 Subservice observables

a. The following observables shall be defined for the parameter functional reporting configuration subservice:

1. Number of requests having passed start of execution verification
2. Number of requests having failed start of execution verification
3. Number of requests having passed completion of execution verification
4. Number of requests having failed completion of execution verification

6.2 ST[02] device access

6.2.1 Scope

6.2.1.1 General

The device access service type provides the capability of distributing commands to and acquiring data from the on-board devices. The corresponding services rely on the low-level device communication mechanisms; hence, they do not require any device-specific application level protocol.

The device access service type defines a single standardized subservice type, i.e. the device access subservice type.

6.2.1.2 Device access subservice

An on-board device can be any on-board entity that can be configured by means of commands or that is able to generate data.

The device access subservice type provides capabilities to interact with:

- on-board devices such as actuators, sensors, transponders and equipment that have no direct support for PUS services;
- equipment during the assembly, integration and test phases or in-flight trouble-shooting, e.g. to validate the basic communication capabilities.

On-board device commands are inserted within requests. On-board device observables are reported within reports.

On-board device commands are mainly intended for bypassing the nominal functions implemented by the on-board software. To support this, a minimum of device command verifications are performed on-board by the device access service.

The device access service type supports addressing devices physically or logically. Physically accessing a device implies knowledge of the transmission link and of the communication protocol. Logically accessing a device can be done with a command identifier and its parameters. The on-board software maps this logical information onto the physical link and protocol. A typical example is the low-level commanding of a Mil-Std-1553B bus remote terminal:

- to command it as a ‘physical device’, the command word is specified, containing the address, the transmission direction, the sub-address and the data word count.
- On the other hand, to command the same remote terminal as a ‘logical device’, the logical device identifier, the logical command and its associated parameters are specified. It is the task of the service to map such a command onto the right communication protocol and physical link.

The device access subservice type provides capabilities for the following:

- On/off device commands;
- Register load commands and register contents acquisition;
- CPDU commands distributed by software;

- Physical device low-level commands for configuration and actuation;
- Physical device low-level commands for data acquisition;
- Logical device low-level commands for configuration and actuation;
- Logical device low-level commands for data acquisition.

6.2.2 Service layout

6.2.2.1 Subservice

6.2.2.1.1 Device access subservice

a. Each device access service shall contain at least one device access subservice.

6.2.2.2 Application process

a. Each application process shall host at most one device access subservice provider.

6.2.3 Capability

a. The device access subservice shall provide at least one of:
   1. the capability for distributing on/off device commands specified in clause 6.2.4;
   2. the capability for distributing register commands specified in clause 6.2.5;
   3. the capability for distributing software CPDU commands specified in clause 6.2.6;
   4. the capability for physical and logical devices commanding and data acquisition specified in clause 6.2.7.

6.2.4 On/off device

6.2.4.1 General

a. The list of on-off devices that are accessed by the device access subservice shall be declared when specifying that subservice.

b. For each on/off device, the hardware addresses that the device access subservice uses to command that device shall be declared when specifying that subservice.

NOTE The addresses can, for example, include the addresses to switch a device on or off, to cold or warm reset, to open or close valves or to command a switch.

6.2.4.2 Distribute on/off device commands

a. The device access subservice capability to distribute on/off device commands shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[2,1] distribute on/off device commands".

NOTE 2 For that declaration, refer to requirement 6.2.3a.

b. Each request to distribute on/off device commands shall contain an ordered list of one or more instructions to distribute an on/off device command.

NOTE The delay to apply between two consecutive instructions is dependent on the spacecraft architecture.

c. Each instruction to distribute an on/off device command shall contain:

1. the device address.

NOTE For item 1, refer to requirement 6.2.4.1b.

d. The device access subservice shall reject any request to distribute on/off device commands if:

1. that request contains an instruction that refers to an unknown device address.

e. For each request to distribute on/off device commands that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to distribute on/off device commands that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to distribute an on/off device command, the device access subservice shall:

1. distribute the related on/off command to the related device address.

6.2.5 Register

6.2.5.1 General

a. The list of registers that are accessed by the device access subservice shall be declared when specifying that subservice.

b. For each register, the hardware address that the device access subservice uses to access that register shall be declared when specifying that subservice.

NOTE The set of registers that are accessible for loading can differ from the set of registers that are accessible for dumping.

c. For each register, the set of register fields used to configure that register shall be declared when specifying that register.

d. For each register, the checks that the device access subservice performs when loading that register shall be declared when specifying that subservice.

NOTE 1 The checks when loading a register are called "register consistency checks".

NOTE 2 The declaration of the register consistency checks can also be made e.g. per type of registers.

6.2.5.2 Distribute register load commands

a. The device access subservice capability to distribute register load commands shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[2,2] distribute register load commands".

NOTE 2 For that declaration, refer to requirement 6.2.3a.

b. Each request to distribute register load commands shall contain an ordered list of one or more instructions to distribute a register load command.

c. Each instruction to distribute a register load command shall contain:

1. the register address;
2. the data for the register fields.

NOTE 1 For item 1, refer to requirement 6.2.4.1b.

NOTE 2 For item 2, refer to requirement 6.2.5.1c.

d. The device access subservice shall reject any request to distribute register load commands if any of the following conditions occurs:

1. that request contains an instruction that refers to an unknown register address;
2. that request contains an instruction that fails its register consistency checks.

e. For each request to distribute register load commands that is rejected, the device access subservice shall generate a failed start of execution notification.

NOTE A partial load can result in an unknown or inconsistent device status.

f. For each request to distribute register load commands that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to distribute a register load command, the device access subservice shall:

1. distribute the command to the register.

6.2.5.3 Distribute register dump commands

a. The device access subservice capability to distribute register dump commands shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[2,5] distribute register dump commands". The

responses are data reports of message type "TM[2,6] register dump report".

NOTE 2 That capability requires the capability for that subservice to distribute register load commands (refer to clause 6.2.5.2).

b. Each request to distribute register dump commands shall contain an ordered list of one or more instructions to distribute a register dump command.

c. Each instruction to distribute a register dump command shall contain:

1. the register address.

NOTE For item 1, refer to requirement 6.2.5.1b.

d. The device access subservice shall reject any request to distribute register dump commands if:

1. that request contains an instruction that refers to an unknown register address.

e. For each request to distribute egister dump commands that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to distribute register dump commands that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to distribute a register dump command, the device access subservice shall:

1. distribute that register dump command;
2. generate the corresponding register dump notification that includes:
   (a) the register address,
   (b) the register data made of the value of each register field.

h. For each valid request to distribute register dump commands, the device access subservice shall generate a single register dump report that contains all related register dump notifications.

6.2.6 CPDU

6.2.6.1 General

a. The list of CPDUs managed by the device access subservice shall be declared when specifying that subservice.

NOTE The CPDUs addressed by the device access subservice are those specified in clause9.

6.2.6.2 Distribute CPDU commands

a. The device access subservice capability to distribute CPDU commands shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[2,4] distribute CPDU commands".

NOTE 2 For that declaration, refer to requirement 6.2.3a.

b. Each request to distribute CPDU commands shall contain an ordered list of one or more instructions to distribute a CPDU command.

NOTE The delay to apply between two consecutive instructions is dependent on the spacecraft architecture.

c. Each instruction to distribute a CPDU command shall contain:

1. the identifier of that CPDU;
2. an ordered list of one or more command pulse instructions.

NOTE 1 For item 1, refer to requirements 6.2.6.1a and 9.2.2b.

NOTE 2 For item 2, refer to requirement 9.2.3b.

d. The device access subservice shall reject any request to distribute CPDU commands if:

1. that request contains an instruction that refers to an unknown CPDU.

e. For each request to distribute CPDU commands that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to distribute CPDU commands that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to distribute a CPDU command, the device access subservice shall:

1. reconstruct the CPDU request in the format expected by that CPDU;
2. distribute that CPDU request.

NOTE This Standard does not prescribe any delay constraint related to the generation of two consecutive CPDU requests.

6.2.7 Physical and logical device access

6.2.7.1 Physical device commanding and data acquisition

6.2.7.1.1 Physical devices

a. For each device that can be physically addressed, the device identifier and the communication links that the device access subservice uses to address that device shall be declared when specifying that subservice.

b. For each physical device and for each associated communication link, the protocols to use over that communication link for transmitting commands or receiving data shall be declared when specifying that physical device and that communication link.

NOTE For example, a physical device may be reached via two communication links, e.g. a Mil-Std-1553B bus

and a SpaceWire link. For each of the associated communication links, one or more protocols can be defined, e.g. one for commanding, one for receiving data.

6.2.7.1.2 Distribute physical device commands

a. The device access subservice capability to distribute physical device commands shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[2,7] distribute physical device commands".

NOTE 2 Each command to a physical device is either for device configuration or for device actuation.

NOTE 3 For that declaration, refer to requirement 6.2.3a.

b. Each request to distribute physical device commands shall contain an ordered list of one or more instructions to distribute a physical device command.

NOTE 1 The instructions referred to the same physical device are dispatched to the device in the order specified in the request and without implementing any delay apart from that which is intrinsic in the transmission protocol. In principle, these instructions are dispatched at the maximum rate allowed by the transmission protocol.

NOTE 2 No relationship can be assumed for the ordering of dispatch among instructions specifying different physical devices referred to in the same request.

c. Each instruction to distribute a physical device command shall contain:

1. the physical device identifier;
2. the protocol-specific data;
3. the command data.

NOTE 1 For example, if the physical device is a Mil-Std-1553B bus remote terminal:

- the physical device identifier can represent the bus remote terminal address. In this case, the physical device identifier implicitly indicates the bus to use;
- the protocol-specific data can represent the transaction direction, the sub-address (or mode code indicator), the data word count (or mode code);
- the command data can represent the data words of the bus message, i.e. a maximum of 32 "16-bits-words".

NOTE 2 For item 1, refer to requirement 6.2.7.1.1a.

NOTE 3 For items 2 and 3, the protocol specific data and the command data are specific to the device identified

by the physical device identifier and driven by requirement 6.2.7.1.1b for that device.

d. The device access subservice shall reject any request to distribute physical device commands if any of the following conditions occurs:
   1. that request contains an instruction that refers to an unknown physical device;
   2. that request contains an instruction that contains invalid protocol-specific data.

e. For each request to distribute physical device commands that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to distribute physical device commands that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to distribute a physical device command, the device access subservice shall:
   1. transmit the command data to the physical device by using the protocol-specific data and the applicable protocol;
   2. check the result of the transmission;
   3. if the command transmission check is not successful, generate a failed execution notification that includes:
      (a) the instruction index within the request;
      (b) the transmission return code;
      (c) if available, the auxiliary data associated to that transmission return code that details the failure reason.

h. For each request to distribute physical device commands that results in at least one unsuccessful command transmission check, the device access subservice shall generate a single failed completion of execution verification report that contains the first failed progress of execution notification generated for that request.

6.2.7.1.3 Acquire data from physical devices

a. The device access subservice shall provide the capability to acquire data from physical devices if the capability to distribute physical device commands is provided by that subservice.

   NOTE 1 The corresponding requests are of message type "TC[2,8] acquire data from physical devices". The responses are data reports of message type "TM[2,9] physical device data report".

   NOTE 2 For the capability to distribute physical device commands, refer to clause 6.2.7.1.2.

b. Each request to acquire data from physical devices shall contain an ordered list of one or more instructions to acquire data from a physical device.

c. Each instruction to acquire data from a physical device shall contain:

1. the transaction identifier;
2. the physical device identifier;
3. the protocol-specific data that is used to identify the data to report.

NOTE 1 For item 1, in the physical device data report, the transaction identifier is used to identify the request and the instruction.

NOTE 2 For item 2, refer to requirement 6.2.7.1.1a.

NOTE 3 For item 3, the protocol specific data field is specific to the device identified by the physical device identifier and driven by requirement 6.2.7.1.1b for that device.

d. The device access subservice shall reject any request to acquire data from physical devices if any of the following conditions occurs:
   1. that request contains an instruction that refers to an unknown physical device;
   2. that request contains an instruction that contains invalid protocol-specific data.

e. For each request to acquire data from physical devices that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to acquire data from physical devices that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to acquire data from a physical device, the device access subservice shall:
   1. transmit the acquisition command to the physical device by using the protocol-specific data and the applicable protocol;
   2. check the data acquisition return code that reports on the result of the transmission;
   3. if the data acquisition is successful, generate a single physical device data notification that includes:
      (a) the transaction identifier;
      (b) the data acquisition return code;
      (c) the auxiliary data associated to that data acquisition return code, if any;
      (d) the data block corresponding to the acquired data.
   4. if the data acquisition fails, generate a failed execution notification that includes:
      (a) the transaction identifier;
      (b) the transaction execution status, which consists of the data acquisition, the return code and associated auxiliary data.

      NOTE A physical device data report contains a single physical device data notification.

h. For each physical device and for each communication link, the list of data acquisition return codes and associated auxiliary data shall be declared when specifying that physical device and that communication link.

NOTE Auxiliary data can be associated to each data acquisition return code in the list, to provide detail reporting on the reason for that return code.

i. For each request to acquire data from physical devices that results in at least one data acquisition failure, the device access subservice shall generate a single failed completion of execution verification report that includes the first failed progress of execution notification generated for that request.

6.2.7.2 Logical device commanding and data acquisition

6.2.7.2.1 Logical devices

a. For each device that can be logically addressed, the logical device identifier, the set of supported commands and associated arguments that the device access subservice uses to address that device and the set of parameter identifiers used for data acquisition shall be declared when specifying that subservice.

6.2.7.2.2 Distribute logical device commands

a. The device access subservice capability to distribute logical device commands shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[2,10] distribute logical device commands".

NOTE 2 Each command to a logical device is either for device configuration or for device actuation.

NOTE 3 That capability requires the capability for that subservice to distribute physical device commands (refer to clause 6.2.7.1.2).

b. Each request to distribute logical device commands shall contain an ordered list of one or more instructions to distribute a logical device command.

NOTE 1 The instructions referred to the same logical device are dispatched to the device in the order specified in the request and without implementing any delay apart from that which is intrinsic in the transmission protocol. In principle, these instructions are dispatched at the maximum rate allowed by the transmission protocol.

NOTE 2 No relationship can be assumed among instructions specifying different logical devices referred to in the same request.

c. Each instruction to distribute a logical device command shall contain:

1. the logical device identifier;
2. the command identifier;

3. the command arguments.

NOTE 1 The instructions in a request to distribute logical device commands do not contain any reference to the physical link or to the transmission protocol of a device. Logically commanding a device allows for example to use the same request for interfacing a device during the development of the on-board software and during in-flight operations, i.e. the same user request protocol but different means to physically address the device.

NOTE 2 For item 1, refer to requirement 6.2.7.2.1a.

NOTE 3 For items 2 and 3, the command ID and the command arguments are specific to the logical device, refer to requirement 6.2.7.2.1a.

d. The device access subservice shall reject any request to distribute logical device commands if any of the following conditions occurs:
   1. that request contains an instruction that refers to an unknown logical device;
   2. that request contains an instruction that refers to an unknown command.

e. For each request to distribute logical device commands that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to distribute logical device commands that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to distribute a logical device command, the device access subservice shall:
   1. map the logical device identifier onto the physical device identifier, the communication link and the communication protocol;
   2. map the command identifier onto the protocol-specific data;
   3. use the command arguments to format the command data for transmission;
   4. transmit the command data to the physical device by using the protocol-specific data and the applicable protocol;
   5. check the result of the transmission;
   6. if the command transmission check is not successful, generate, for that instruction, a failed execution notification that includes:
      (a) the instruction index within the request;
      (b) the transmission return code;
      (c) the auxiliary data associated to that transmission return code that details the failure reason, if any.

h. For each request to distribute logical device commands that results in at least one unsuccessful command transmission check, the device access subservice shall generate a single failed completion of execution

verification report that includes the first failed progress of execution notification generated for that request.

6.2.7.2.3 Acquire data from logical devices

a. The device access subservice shall provide the capability to acquire data from logical devices if the capability to distribute logical device commands is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[2,11] acquire data from logical devices". The responses are data reports of message type "TM[2,12] logical device data report".

NOTE 2 For the capability to distribute logical device commands, refer to clause 6.2.7.2.2.

b. Each request to acquire data from logical devices shall contain an ordered list of one or more instructions to acquire data from a logical device.

c. Each instruction to acquire data from a logical device shall contain:

1. the transaction identifier;
2. the logical device identifier;
3. the parameter identifier of the data to report.

NOTE 1 In the logical device data report, the transaction identifier is used to identify the request and the instruction.

NOTE 2 The instructions in a request to acquire data from logical devices do not contain any reference to the physical link or to the transmission protocol of a device.

NOTE 3 For items 2 and 3, refer to requirement 6.2.7.2.1a.

d. The device access subservice shall reject any request to acquire data from logical devices if any of the following conditions occurs:

1. that request contains an instruction that refers to an unknown logical device;
2. that request contains an instruction that refers to an unknown parameter.

e. For each request to acquire data from logical devices that is rejected, the device access subservice shall generate a failed start of execution notification.

f. For each request to acquire data from logical devices that contains only valid instructions, the device access subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to acquire data from a logical device, the device access subservice shall:

1. map the logical device identifier onto the physical device identifier, the communication link and the communication protocol;
2. map the parameter identifier onto the protocol-specific data;

3. transmit the acquisition command to the physical device by using the protocol-specific data and the applicable protocol;
4. check the data acquisition return code that reports on the result of the transmission;
5. if the data acquisition is successful, generate a single logical device data notification that includes:
   (a) the transaction identifier;
   (b) the data acquisition return code;
   (c) if available, the auxiliary data associated to that data acquisition return code;
   (d) the acquired parameter value.
6. if the data acquisition is successful, generate a logical device data report that includes that logical device data notification;
7. if the data acquisition fails, generate, for that instruction, a failed execution notification that includes:
   (a) the transaction identifier;
   (b) the transaction execution status, which consists of the data acquisition return code and, if any, the associated auxiliary data.

   NOTE Each logical device data report contains exactly one logical device data notification.

h. For each logical device, the list of data acquisition return codes and associated auxiliary data shall be declared when specifying that logical device.

   NOTE Auxiliary data can be associated to each data acquisition return code in the list, to provide detail reporting on the reason for that return code.

i. For each request to acquire data from logical devices that results in at least one data acquisition failure, the device access subservice shall generate a single failed completion of execution verification report that includes the first failed progress of execution notification generated for that request.

6.2.8 Subservice observables

This Standard does not define any observables for the device access subservice.

6.3 ST[03] housekeeping

6.3.1 Scope

6.3.1.1 General

The housekeeping service type provides means to control and adapt the spacecraft reporting plan according to the mission phases.

The housekeeping service type provides the visibility of any on-board parameters assembled in housekeeping parameter report structures of different categories as required for the mission. The parameter report structures used by the housekeeping service can be predefined on-board or created when needed.

The housekeeping service type defines two standardized subservice types, i.e.:

- the housekeeping reporting subservice type,
- the parameter functional reporting configuration subservice type.

6.3.1.2 Housekeeping reporting subservice

The housekeeping reporting subservice type provides reporting functions respectively dedicated to nominal and contingency operations.

The housekeeping parameter reports, can be generated periodically or on request.

The periodic generation of each type of parameter report can be enabled or disabled. For example, the periodic generation of the reports for a housekeeping parameter report type can be disabled to reduce the on-board processing load. Another housekeeping parameter report type can be enabled when a particular anomaly occurs and be disabled at other times.

A collection interval is attached to each type of parameter report. The collection interval represents the time interval at which the parameters are collected to generate the corresponding reports.

A sampling interval is associated to each on-board parameter. The sampling interval is used by the application process responsible for acquiring or calculating the values of the corresponding parameter.

Each parameter report is defined as a combination of simply commutated parameters and/or super commutated parameters.

A simply commutated parameter definition implies that only one sampled value of that parameter is present within each related parameter report corresponding to one value of the parameter collected during the collection interval.

A super commutated parameter definition implies that more than one sampled values of that parameter is present, each sample value corresponding to a value of the parameter collected during the collection interval at a sub-period equal to the collection interval divided by the number of super commutated sampled values.

Within a parameter report definition, each related parameter appears only once, either as a simply commutated parameter or as a super commutated parameter.

Each housekeeping parameter report has an associated housekeeping category, grouping reports of the same operational significance. This standard defines six categories which use can be tailored depending on mission needs. To facilitate ground system detection and routing the housekeeping category is indicated by the message type of the generated report.

6.3.1.3 Parameter functional reporting configuration subservice

The parameter functional reporting configuration subservice type provides the capability to control the generation of the parameter reports generated by the housekeeping reporting subservice e.g. to ease the management of housekeeping configuration on mode transitions.

The parameter functional reporting configuration subservice operates on sets of housekeeping parameter reports, e.g. enabling or disabling the generation of such sets. Functional configurations can be applied exclusively, in which case the periodic generation of each report type of the service is disabled before applying the functional configurations.

6.3.2 Service layout

6.3.2.1 Subservice

6.3.2.1.1 Housekeeping reporting subservice

a. Each housekeeping service shall contain at least one housekeeping reporting subservice.

6.3.2.1.2 Parameter functional reporting configuration subservice

a. Each housekeeping service shall contain at most one parameter functional reporting configuration subservice.

6.3.2.2 Application process

6.3.2.2.1 Housekeeping reporting subservice

a. Each application process shall host at most one housekeeping reporting subservice provider.

6.3.2.2.2 Parameter functional reporting configuration subservice

a. Each application process shall host at most one parameter functional reporting configuration subservice provider.

6.3.3 Housekeeping reporting subservice

6.3.3.1 Parameter accessibility

a. The housekeeping reporting subservice shall be able to collect and report the sampled values of each on-board parameter that is accessible to the application process that hosts that subservice.

6.3.3.2 Housekeeping reporting configuration

a. The housekeeping reporting subservice configuration policy to apply when starting and restarting the housekeeping reporting subservice shall be declared when specifying the housekeeping reporting subservice.

NOTE in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

6.3.3.3 Housekeeping parameter report structure

a. The on-board resources allocated to the housekeeping reporting subservice to host the housekeeping parameter report structures shall be declared when specifying that subservice.

NOTE The allocated resources constrain the number of housekeeping parameter report structures and their content, in number of parameters.

b. The on-board resources allocated to the contemporaneous evaluation of housekeeping parameter report structures used by the housekeeping reporting subservice shall be declared when specifying that subservice.

NOTE The number of housekeeping parameter report structures that can be contemporaneously evaluated by the subservice depends on these resources and the overall number of sampled values required for each corresponding report.

c. Each housekeeping parameter report structure shall consist of:

1. a housekeeping parameter report structure identifier;
2. the collection interval used to generate the corresponding reports;
3. an ordered list of zero or more simply commutated parameters;
4. an ordered list of zero or more super commutated parameter sets, each set consisting of:
   (a) the number of sampled values to report for each parameter of that set, and
   (b) the ordered list of one or more parameters contained within that set;
5. if the housekeeping reporting subservice provides the capability for managing the periodic generation of housekeeping parameter reports, a status indicating whether the periodic generation action of the corresponding housekeeping parameter reports is enabled or disabled.

NOTE 1 The collection interval is expressed as units of the minimum sampling interval, refer to requirement 5.4.3.3c.

NOTE 2 For item 4(a), the number of sampled values to report for each parameter of the set is named "super commutated sample repetition number".

NOTE 3 For item 5:

- for the capability for managing the periodic generation of housekeeping parameter reports, refer to clause 6.3.3.5;
- this status is named "housekeeping parameter report periodic generation action status". If the housekeeping subservice does not provide the capability for managing the periodic generation of housekeeping parameter reports, the periodic generation of housekeeping parameter reports is always enabled.

d. Each housekeeping report structure shall be uniquely identified by the combination of the application process that hosts the housekeeping reporting subservice provider and its housekeeping parameter report structure identifier.

e. Each housekeeping report structure shall have an associated housekeeping report category which is statically deduced from the housekeeping structure identifier.

NOTE The housekeeping report category determines the message type used by that report.

f. The association between housekeeping categories and housekeeping parameter report structure identifier ranges shall be declared when specifying the housekeeping reporting subservice.

NOTE An example of category usage tailoring is to assign category 1 to nominal housekeeping reports, category 2 to diagnostic reports, category 3 to essential reports used in low bit rate, category 4 for troubleshooting reports only stored on board, and leaving category 5 and 6 as spare.

6.3.3.4 Housekeeping parameter report

a. The housekeeping reporting subservice shall provide the capability for generating housekeeping parameter reports.

NOTE The corresponding reports are data reports of message type:

- "TM[3,25] category 1 housekeeping parameter report";
- "TM[3,26] category 2 housekeeping parameter report";
- "TM[3,50] category 3 housekeeping parameter report";
- "TM[3,51] category 4 housekeeping parameter report";
- "TM[3,52] category 5 housekeeping parameter report";
- "TM[3,53] category 6 housekeeping parameter report";

b. Each housekeeping parameter report shall contain exactly one housekeeping parameter notification.

c. Each housekeeping parameter notification shall contain:

1. the housekeeping parameter report structure identifier;
2. in the specified order for simply commutated parameters, a single sampled value for each simply commutated parameter;
3. in the specified order for super commutated parameter sets, for each super commutated parameter set:
   (a) the "super commutated sample repetition number" sets of sampled values.

NOTE 1 For the housekeeping parameter report structure, refer to clause 6.3.3.2.

NOTE 2 For item 3(a), each set of sampled values is composed of a single sampled value for each parameter of the super commutated parameter set. The sampled values are ordered according to the ordering of the parameters within the corresponding super commutated parameter set. For example, for the super commutated parameter set that contains 2 parameters A and B, if the required number of sampled values is 2, each report will contain "value 1 of A", "value 1 of B", "value 2 of A", "value 2 of B" in that order.

d. For each housekeeping parameter report structure for which periodic generation is enabled, the housekeeping reporting subservice shall generate a corresponding housekeeping parameter report periodically, according to the collection interval specified for that definition.

NOTE For the collection interval, refer to requirement 6.3.3.3c.

e. For each housekeeping parameter report structure for which periodic generation is enabled, the housekeeping reporting subservice shall collect one sampled value for each simply commutated parameter during the collection interval specified for the corresponding housekeeping parameter report structure.

f. For each housekeeping parameter report structure for which periodic generation is enabled, the housekeeping reporting subservice shall collect all sampled values for each super commutated parameter during the collection interval specified for the corresponding housekeeping parameter report structure, in accordance with a sub-period equal to the collection interval divided by the corresponding "super commutated sample repetition number".

NOTE If the sub-period is shorter than the period at which the parameter value is updated by the on-board software, some sampled values will be identical.

6.3.3.5 Managing the periodic generation of housekeeping parameter reports

6.3.3.5.1 Enable the periodic generation of housekeeping parameter reports

a. The housekeeping reporting subservice capability to enable the periodic generation of housekeeping parameter reports shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,5] enable the periodic generation of housekeeping parameter reports".

NOTE 2 Enabling and disabling the periodic generation of housekeeping parameter reports is required if the housekeeping service includes a parameter functional reporting configuration subservice, refer to clause 6.3.5.

NOTE 3 For the capability to disable the periodic generation of housekeeping parameter reports, refer to clause 6.3.3.5.2.

b. Each request to enable the periodic generation of housekeeping parameter reports shall contain an ordered list of one or more instructions to enable the periodic generation of a housekeeping parameter report.

c. Each instruction to enable the periodic generation of a housekeeping parameter report shall contain:

1. the housekeeping parameter report structure identifier to enable.

d. The housekeeping reporting subservice shall reject any request to enable the periodic generation of housekeeping parameter reports if:

1. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown.

e. For each request to enable the periodic generation of housekeeping parameter reports that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each request to enable the periodic generation of housekeeping parameter reports that contains only valid instructions, the housekeeping reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable the periodic generation of a housekeeping parameter report, the housekeeping reporting subservice shall:

1. set the periodic generation action status of that housekeeping parameter report structure to "enabled".

6.3.3.5.2 Disable the periodic generation of housekeeping parameter reports

a. The housekeeping reporting subservice shall provide the capability to disable the periodic generation of housekeeping parameter reports if the

capability to enable the periodic generation of housekeeping parameter reports is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,6] disable the periodic generation of housekeeping parameter reports".

NOTE 2 For the capability to enable the periodic generation of housekeeping parameter report, refer to clause 6.3.3.5.1.

b. Each request to disable the periodic generation of housekeeping parameter reports shall contain exactly one of:

1. An ordered list of one or more instructions to disable the periodic generation of a housekeeping parameter report;
2. a single instruction to disable the periodic generation of all housekeeping parameter reports.

NOTE The instructions to disable the periodic generation of all housekeeping parameter reports contain no argument.

c. Each instruction to disable the periodic generation of a housekeeping parameter report shall contain:

1. the housekeeping parameter report structure identifier to disable.

d. The housekeeping reporting subservice shall reject any request to disable the periodic generation of housekeeping parameter reports if:

1. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown.

e. For each request to disable the periodic generation of housekeeping parameter reports that it is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each request to disable the periodic generation of housekeeping parameter reports that contains only valid instructions, the housekeeping reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable the periodic generation of a housekeeping parameter report, the housekeeping reporting subservice shall:

1. set the periodic generation action status of that housekeeping parameter report structure to "disabled".

6.3.3.6 Creating and deleting housekeeping parameter report structures

6.3.3.6.1 Create a housekeeping parameter report structure

a. The housekeeping reporting subservice capability to create a housekeeping parameter report structure shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,1] create a housekeeping parameter report structure".

NOTE 2 For the capability to delete housekeeping parameter report structures, refer to clause 6.3.3.6.2.

b. Each request to create a housekeeping parameter report structure shall contain exactly one instruction to create a housekeeping parameter report structure.

c. Each instruction to create a housekeeping parameter report structure shall contain:

1. the housekeeping parameter report structure identifier to create;
2. the collection interval;
3. the list of simply commutated parameters in the required order;
4. the list of super commutated parameter sets in the required order.

NOTE 1 The ordering of the simply and super commutated parameter sets corresponds to the order of the corresponding sampled values in the housekeeping parameter reports.

NOTE 2 See clause 6.3.3.2.

d. The housekeeping reporting subservice shall reject any request to create a housekeeping parameter report structure if any of the following conditions occurs:

1. that request contains an instruction that refers to a housekeeping parameter report structure that is already in use;
2. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown;
3. the same parameter is identified more than once in that request;
4. the resources allocated to the hosting of housekeeping parameter report structures are exceeded.

e. For each request to create a housekeeping parameter report structure that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each valid instruction to create a housekeeping parameter report structure, the housekeeping reporting subservice shall:

1. create that definition;
2. set its periodic generation action status to "disabled".

6.3.3.6.2 Delete housekeeping parameter report structures

a. The housekeeping reporting subservice shall provide the capability to delete housekeeping parameter report structures if the capability to create a housekeeping report definition is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,3] delete housekeeping parameter report structures".

NOTE 2 This Standard assumes that all housekeeping parameter report structures (predefined or created by request) can be deleted.

NOTE 3 For the capability to create a housekeeping parameter report structure, refer to clause 6.3.3.6.1.

b. Each request to delete housekeeping parameter report structures shall contain an ordered list of one or more instructions to delete a housekeeping parameter report structure.

c. Each instruction to delete a housekeeping parameter report structure shall contain:

1. the housekeeping parameter report structure identifier to delete.

d. The housekeeping reporting subservice shall reject any request to delete housekeeping parameter report structures if any of the following conditions occurs:

1. that request contains an instruction that refers to a housekeeping parameter report structure that is not in use;

2. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown;

3. that request contains an instruction that refers to a housekeeping parameter report structure whose periodic generation action status is "enabled".

e. For each request to delete housekeeping parameter report structures that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each request to delete housekeeping parameter report structures that contains only valid instructions, the housekeeping reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a housekeeping parameter report structure, the housekeeping reporting subservice shall:

1. delete the housekeeping parameter report structure referred to by that instruction.

6.3.3.7 Report housekeeping parameter report structures

a. The housekeeping reporting subservice capability to report housekeeping parameter report structures shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,9] report housekeeping parameter report structures". The responses, one for each instruction, are data reports of message type "TM[3,10] housekeeping parameter report structure report".

NOTE 2 That capability requires the capability for that subservice to create a housekeeping parameter report type (refer to clause 6.3.3.6.1).

NOTE 3 All housekeeping parameter report structures are available for reporting, i.e. including those that are predefined on-board.

b. Each request to report housekeeping parameter report structures shall contain an ordered list of one or more instructions to report a housekeeping parameter report structure.

c. Each instruction to report a housekeeping parameter report structure shall contain:

1. the housekeeping parameter report structure identifier to report.

d. The housekeeping reporting subservice shall reject any request to report housekeeping parameter report structures if:

1. That request contains an instruction that refers to a housekeeping parameter report structure that is unknown.

e. For each request to report housekeeping parameter report structures that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each request to report housekeeping parameter report structures that contains only valid instructions, the housekeeping reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report a housekeeping parameter report structure, the housekeeping reporting subservice shall generate a single housekeeping parameter report structure report that contains exactly one housekeeping parameter report structure notification that includes:

1. the housekeeping parameter report structure identifier;
2. If the housekeeping reporting subservice provides the capability for managing the periodic generation of housekeeping parameter reports, the periodic generation action status;
3. the collection interval;
4. the ordered list of simply commutated parameters;
5. the ordered list of super commutated parameter sets.

NOTE For item 2 capability for managing the periodic generation of housekeeping parameter reports, refer to clause 6.3.3.5.

6.3.3.8 Generate a one shot report for housekeeping parameter report structures

a. The housekeeping reporting subservice capability to generate a one shot report for housekeeping parameter report structures shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[3,27] generate a one shot report for housekeeping parameter report structures". The responses, one for each instruction, are data reports of the following message types :

- "TM[3,25] category 1 housekeeping parameter report";
- "TM[3,26] category 2 housekeeping parameter report";
- "TM[3,50] category 3 housekeeping parameter report";
- "TM[3,51] category 4 housekeeping parameter report";
- "TM[3,52] category 5 housekeeping parameter report";
- "TM[3,53] category 6 housekeeping parameter report";

b. Each request to generate a one shot report for housekeeping parameter report structures shall contain an ordered list of one or more instructions to generate a one shot report for a housekeeping parameter report structure.

c. Each instruction to generate a one shot report for a housekeeping parameter report structure shall contain:

1. the housekeeping parameter report structure identifier of the report to generate.

d. The housekeeping reporting subservice shall reject any request to generate a one shot report for housekeeping parameter report structures if:

1. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown.

e. For each request to generate a one shot report for housekeeping parameter report structures that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each request to generate a one shot report for housekeeping parameter report structures that contains only valid instructions, the housekeeping reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to generate a one shot report for a housekeeping parameter report structure, the housekeeping reporting subservice shall generate a single housekeeping parameter report.

NOTE 1 The housekeeping parameter report is defined in clause 6.3.3.4.

NOTE 2 This Standard does not prescribe the behaviour of the housekeeping reporting subservice when the housekeeping parameter report includes super commutated parameters. The content of the super commutated part of the housekeeping parameter reports is implementation dependent.

6.3.3.9 Append parameters to a housekeeping parameter report structure

a. The housekeeping reporting subservice capability to append parameters to a housekeeping parameter report structure shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,29] append parameters to a housekeeping parameter report structure".

NOTE 2 That capability requires the capability for that subservice to create a housekeeping parameter report type (refer to clause 6.3.3.6.1).

NOTE 3 This Standard assumes that all housekeeping parameter report structures (predefined or created by request) can be modified by that request.

b. Each request to append parameters to a housekeeping parameter report structure shall contain exactly one instruction to append parameters to a housekeeping parameter report structure.

c. Each instruction to append parameters to a housekeeping parameter report structure shall contain:

1. the housekeeping parameter report structure identifier to modify;
2. if the housekeeping parameter report structure only includes simply commutated parameters, at least one of:
   (a) the ordered list of simply commutated parameters to add;
   (b) the ordered list of super commutated parameter sets to add;
3. if the housekeeping parameter report structure includes super commutated parameters:
   (a) the ordered list of super commutated parameter sets to add.

d. The housekeeping reporting subservice shall reject any request to append parameters to a housekeeping parameter report structure if any of the following conditions occurs:

1. the periodic generation action status of the housekeeping parameter report is "enabled";
2. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown;
3. that request contains an instruction that refers to a parameter that is unknown;
4. that request contains an instruction that refers to simply commutated parameters to add to a definition that contains super commutated parameters;
5. that request contains an instruction that refers to a parameter that is already present in the definition;
6. the resources allocated to the hosting of housekeeping parameter report structures are exceeded.

e. For each request to append parameters to a housekeeping parameter report structure that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each valid instruction to append parameters to a housekeeping parameter report structure, the housekeeping reporting subservice shall:

1. add, at the end of the housekeeping parameter report structure, the list of simply commutated parameters, if any, followed by the list of super commutated parameter sets, if any.

6.3.3.10 Modify the collection interval of housekeeping parameter report structures

a. The housekeeping reporting subservice capability to modify the collection interval of housekeeping parameter report structures shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,31] modify the collection interval of housekeeping parameter report structures".

NOTE 2 This Standard assumes that all housekeeping parameter report structures (predefined or created by request) can be modified by that request.

b. Each request to modify the collection interval of housekeeping parameter report structures shall contain an ordered list of one or more instructions to modify the collection interval of a housekeeping parameter report structure.

c. Each instruction to modify the collection interval of a housekeeping parameter report structure shall contain:

1. the housekeeping parameter report structure identifier to modify;
2. the new collection interval.

NOTE The collection interval is expressed as units of the minimum sampling interval, refer to requirement 5.4.3.3c.

d. The housekeeping reporting subservice shall reject any request to modify the collection interval of housekeeping parameter report structures if:

1. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown.

e. For each request to modify the collection interval of housekeeping parameter report structures that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each request to modify the collection interval of housekeeping parameter report structures that contains only valid instructions, the housekeeping reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to modify the collection interval of a housekeeping parameter report structure, the housekeeping reporting subservice shall:

1. set the collection interval of that housekeeping parameter report structure to the new collection interval specified in that instruction.

6.3.3.11 Report the periodic generation properties of housekeeping parameter report structures

a. The housekeeping reporting subservice capability to report the periodic generation properties of housekeeping parameter report structures shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[3,33] report the periodic generation properties of housekeeping parameter report structures". The responses are data reports of message type "TM[3,34] housekeeping parameter report periodic generation properties report".

b. Each request to report the periodic generation properties of housekeeping parameter report structures shall contain exactly one of:

1. an ordered list of one or more instructions to report the periodic generation properties of a housekeeping parameter report structure;
2. a single instruction to report the periodic generation properties of all housekeeping parameter report structures.

NOTE The instructions to report the periodic generation properties of all housekeeping parameter report structures contain no argument

c. Each instruction to report the periodic generation properties of a housekeeping parameter report structure shall contain:

1. the housekeeping parameter report structure identifier to report.

d. The housekeeping reporting subservice shall reject any request to report the periodic generation properties of housekeeping parameter report structures if:

1. that request contains an instruction that refers to a housekeeping parameter report structure that is unknown.

e. For each request to report the periodic generation properties of housekeeping parameter report structures that is rejected, the housekeeping reporting subservice shall generate a failed start of execution notification.

f. For each request to report the periodic generation properties of housekeeping parameter report structures that contains only valid instructions, the housekeeping reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report the periodic generation properties of a housekeeping parameter report structure, the housekeeping reporting subservice shall generate a single housekeeping parameter report periodic generation properties notification that includes:

1. the housekeeping parameter report structure identifier;
2. the related periodic generation action status;
3. the related collection interval.

h. For each valid request to report the periodic generation properties of housekeeping parameter report structures, the housekeeping reporting subservice shall generate a single housekeeping parameter report periodic

generation properties report that contains all related housekeeping parameter report periodic generation properties notifications.

6.3.3.12 Subservice observables

This Standard does not define any observables for the housekeeping reporting subservice.

6.3.4 Diagnostic reporting subservice (deleted)

NOTE The diagnostic reporting subservice type was previously defined by the housekeeping service type, but the introduction of housekeeping report categories into the housekeeping reporting subservice type now covers and generalizes its functional requirements. A specific subservice type for diagnostic reporting is thus no longer needed.

6.3.5 Parameter functional reporting configuration subservice

6.3.5.1 Accessibility

a. The parameter functional reporting configuration subservice shall be able to control the generation of each housekeeping parameter report generated by the housekeeping reporting subservices of all housekeeping service instances implemented on-board.

NOTE potentially a service encompases multiple subservice providers. This requirement refers to subservices included in the service, refer to section 5.3.

6.3.5.2 Parameter functional reporting definition

a. Each parameter functional reporting definition shall consist of:
   1. the identifier of that parameter functional reporting definition;
   2. a list of one or more parameter reporting entries.

b. Each parameter reporting entry of a parameter functional reporting definition shall consist of:
   1. the identification of a parameter report definition consisting of:
      (a) if the housekeeping service is distributed on several on-board application processes, the application process identifier of that parameter report definition;
      (b) the identifier of the parameter report definition;

2. the periodic generation action status to apply to the parameter report definition when that parameter functional reporting is applied;
3. the collection interval to apply to the parameter report definition when that parameter functional reporting is applied.

NOTE 1 The housekeeping reporting subservices that can be addressed by the parameter functional reporting configuration subservice are those subservices of the housekeeping service that includes that parameter functional reporting configuration subservice.

NOTE 2 For item 1(b), refer to requirement 6.3.3.3c.1.

6.3.5.3 Apply parameter functional reporting configurations

a. The parameter functional reporting configuration subservice shall provide the capability to apply parameter functional reporting configurations.

NOTE The corresponding requests are of message type "TC[3,37] apply parameter functional reporting configurations".

b. Each request to apply parameter functional reporting configurations shall contain:
   1. the configuration execution flag indicating whether the execution of that request is exclusive or non-exclusive;
   2. an ordered list of one or more instructions to apply a parameter functional reporting configuration.

NOTE An exclusive request execution implies that the periodic generation of known structure parameter report definitions are disabled prior to application of the parameter functional reporting configuration.

c. Each instruction to apply a parameter functional reporting configuration shall contain:
   1. the parameter functional reporting definition identifier.

d. The parameter functional reporting configuration subservice shall reject any request to apply parameter functional reporting configurations if any of the following conditions occurs:
   1. that request refers to an invalid configuration execution flag;
   2. that request contains an instruction that refers to a parameter functional reporting definition that is unknown;

e. For each request to apply parameter functional reporting configurations that is rejected, the parameter functional reporting configuration subservice shall generate a failed start of execution notification.

f. For each request to apply parameter functional reporting configurations that contains at least one valid instruction, the parameter functional reporting configuration subservice shall:

1. if the configuration execution flag of that request is exclusive, set the periodic generation action status of each enabled parameter report of the housekeeping service to "disabled".

   NOTE This implies that all enabled housekeeping parameter reports of the housekeeping reporting subservices hosted by the parent housekeeping service are disabled. This disabling is executed before applying the configurations in the parameter functional reporting definitions identified in the instructions of the request.

g. For each valid request to apply parameter functional reporting configurations that contains only valid instructions, the parameter functional reporting configuration subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to apply a parameter functional reporting configuration, the parameter functional reporting configuration subservice shall:

   1. for each parameter report definition referenced by the parameter functional reporting definition identified in that instruction, instruct the corresponding housekeeping reporting subservice:

      (a) if the parameter report definition exists, to modify the collection interval of that parameter report definition, and

      (b) according to the periodic generation enabling or disabling action specified for that parameter report definition, to enable or to disable the periodic generation of the related parameter reports.

6.3.5.4 Managing parameter functional reporting definitions

6.3.5.4.1 Create a parameter functional reporting definition

a. The parameter functional reporting configuration subservice capability to create a parameter functional reporting definition shall be declared when specifying that subservice.

   NOTE 1 The corresponding requests are of message type "TC[3,38] create a parameter functional reporting definition".

   NOTE 2 For the capability to delete parameter functional reporting definitions, refer to clause 6.3.5.4.2.

b. Each request to create a parameter functional reporting definition shall contain exactly one instruction to create a parameter functional reporting definition.

c. Each instruction to create a parameter functional reporting definition shall contain:

   1. the identifier of the parameter functional reporting definition to create;

   2. a list of one or more parameter reporting entries consisting of:

(a) if the housekeeping service is distributed on several on-board application processes, the application process identifier of that parameter report definition;

(b) the identifier of the parameter report definition;

(c) the periodic generation action status;

(d) the collection interval.

NOTE For item 2(a), refer to requirement 6.3.5.2b.

d. The parameter functional reporting configuration subservice shall reject any request to create a parameter functional reporting definition if:

1. that request contains an instruction that refers to an unknown application process;

2. that request contains an instruction that refers to an unknown parameter report definition;

3. that request contains an instruction that refers to a parameter functional reporting definition that already exists;

4. that request contains more than one instruction for the same parameter report definition.

NOTE for item 2, it is expected this is applicable only to the structure parameter report definitions that the parameter functional reporting configuration subservice has knowledge of

e. For each request to create a parameter functional reporting definition that is rejected, the parameter functional reporting configuration subservice shall generate a failed start of execution notification.

f. For each valid instruction to create a parameter functional reporting definition, the parameter functional reporting configuration subservice shall:

1. create a new parameter functional reporting definition.

6.3.5.4.2 Delete parameter functional reporting definitions

a. The parameter functional reporting configuration subservice shall provide the capability to delete parameter functional reporting definitions if the capability to create a parameter functional reporting definition is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,39] delete parameter functional reporting definitions".

NOTE 2 For the capability to create a parameter functional reporting definition, refer to clause 6.3.5.4.1.

b. Each request to delete parameter functional reporting definitions shall contain an ordered list of one or more instructions to delete a parameter functional reporting definition.

c. Each instruction to delete a parameter functional reporting definition shall contain:

1. the identifier of the parameter functional reporting definition to delete.

d. The parameter functional reporting configuration subservice shall reject any request to delete parameter functional reporting definitions if:

1. that request contains an instruction that refers to a parameter functional reporting definition that is unknown.

e. For each request to delete parameter functional reporting definitions that is rejected, the parameter functional reporting configuration subservice shall generate a failed start of execution notification.

f. For each request to delete parameter functional reporting definitions that contains only valid instructions, the parameter functional reporting configuration subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a parameter functional reporting definition, the parameter functional reporting configuration subservice shall:

1. delete that definition.

6.3.5.5 Report parameter functional reporting definitions

a. The parameter functional reporting configuration subservice capability to report parameter functional reporting definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,40] report parameter functional reporting definitions". The responses, one for each instruction, are data reports of message type "TM[3,41] parameter functional reporting definition report".

NOTE 2 That capability requires the capability for that subservice to create a parameter functional reporting definition (refer to clause 6.3.5.4.1).

b. Each request to report parameter functional reporting definitions shall contain an ordered list of one or more instructions to report a parameter functional reporting definition.

c. Each instruction to report a parameter functional reporting definition shall contain:

1. the identifier of the parameter functional reporting definition to report.

d. The parameter functional reporting configuration subservice shall reject any request to report parameter functional reporting definitions if:

1. that request contains an instruction that refers to a parameter functional reporting definition that is unknown.

e. For each request to report parameter functional reporting definitions that is rejected, the parameter functional reporting configuration subservice shall generate a failed start of execution notification.

f. For each request to report parameter functional reporting definitions that contains only valid instructions, the parameter functional reporting

configuration subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report a parameter functional reporting definition, the parameter functional reporting configuration subservice shall generate a single parameter functional reporting definition report that contains:

1. the identifier of the parameter functional reporting definition;
2. for each related parameter reporting entry, exactly one parameter functional reporting definition notification, that includes:
   (a) if the housekeeping service is distributed on several on-board application processes, the application process identifier;
   (b) the identifier of the parameter report definition;
   (c) the periodic generation action status;
   (d) the collection interval.

6.3.5.6 Modifying the parameter functional reporting definitions

6.3.5.6.1 Add parameter report definitions to a parameter functional reporting definition

a. The parameter functional reporting configuration subservice capability to add parameter report definitions to a parameter functional reporting definition shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,42] add parameter report definitions to a parameter functional reporting definition".

NOTE 2 That capability requires the capability for that subservice to create a parameter functional reporting definition (refer to clause 6.3.5.4.1).

NOTE 3 For the capability to remove parameter report definitions from a parameter functional reporting definition, refer to clause 6.3.5.6.2.

b. Each request to add parameter report definitions to a parameter functional reporting definition shall contain:

1. the identifier of the parameter functional reporting definition;
2. an ordered list of one or more instructions to add a parameter report definition to a parameter functional reporting definition.

c. Each instruction to add a parameter report definition to a parameter functional reporting definition shall contain:

1. the parameter report entry to add that consists of:
   (a) if the housekeeping service is distributed on several on-board application processes, the application process identifier;
   (b) the identifier of the parameter report definition;
   (c) the periodic generation action status;
   (d) the collection interval.

NOTE For item 1(a), refer to requirement 6.3.5.2b.

d. The parameter functional reporting configuration subservice shall reject any request to add parameter report definitions to a parameter functional reporting definition if any of the following conditions occurs:

1. that request refers to a parameter functional reporting definition that is unknown;
2. that request contains more than one instruction for the same parameter report definition;
3. that request contains an instruction that refers to a parameter report definition that is already in that parameter functional reporting definition.

e. For each request to add parameter report definitions to a parameter functional reporting definition that is rejected, the parameter functional reporting configuration subservice shall generate a failed start of execution notification.

f. For each request to add parameter report definitions to a parameter functional reporting definition that contains only valid instructions, the parameter functional reporting configuration subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to add a parameter report definition to a parameter functional reporting definition, the parameter functional reporting configuration subservice shall:

1. add the related definition.

6.3.5.6.2 Remove parameter report definitions from a parameter functional reporting definition

a. The parameter functional reporting configuration subservice shall provide the capability to remove parameter report definitions from a parameter functional reporting definition if the capability to add parameter report definitions to a parameter functional reporting definition is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[3,43] remove parameter report definitions from a parameter functional reporting definition".

NOTE 2 For the capability to add parameter report definitions to a parameter functional reporting definition, refer to clause 6.3.5.6.1.

b. Each request to remove parameter report definitions from a parameter functional reporting definition shall contain:

1. the parameter functional reporting definition identifier;
2. an ordered list of one or more instructions to remove a parameter report definition from a parameter functional reporting definition.

c. Each instruction to remove a parameter report definition from a parameter functional reporting definition shall contain:

1. the identification of the parameter reporting definition to remove, consisting of:
   (a) if the housekeeping service is distributed on several on-board application processes, the application process identifier;

(b) the identifier of the parameter report definition.

NOTE For item 1(a), refer to requirement 6.3.5.2b.

d. The parameter functional reporting configuration subservice shall reject any request to remove parameter report definitions from a parameter functional reporting definition if any of the following conditions occurs:

1. that request refers to a parameter functional reporting definition that is unknown;
2. that request contains an instruction that refers to a parameter report definition that is not in that parameter functional reporting definition.

e. For each request to remove parameter report definitions from a parameter functional reporting definition that is rejected, the parameter functional reporting configuration subservice shall generate a failed start of execution notification.

f. For each request to remove parameter report definitions from a parameter functional reporting definition that contains only valid instructions, the parameter functional reporting configuration subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to remove a parameter report definition from a parameter functional reporting definition, the parameter functional reporting configuration subservice shall:

1. remove that definition.

6.3.5.6.3 Modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition

a. The parameter functional reporting configuration subservice capability to modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[3,44] modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition".

b. Each request to modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition shall contain:

1. the identifier of the parameter functional reporting definition to modify;
2. an ordered list of one or more instructions to modify the periodic generation properties of a parameter report definition of a parameter functional reporting definition.

c. Each instruction to modify the periodic generation properties of a parameter report definition of a parameter functional reporting definition shall contain:

1. if the housekeeping service is distributed on several on-board application processes, the application process identifier of that parameter report definition;
2. the identifier of the parameter report definition;
3. the periodic generation action status;
4. the collection interval.

NOTE For item 1(a), refer to requirement 6.3.5.2b.

d. The parameter functional reporting configuration subservice shall reject any request to modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition if any of the following conditions occurs:
   1. that request refers to a parameter functional reporting definition that is unknown;
   2. that request contains an instruction that refers to a parameter report definition that is not in that parameter functional reporting definition.

e. For each request to modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition that is rejected, the parameter functional reporting configuration subservice shall generate a failed start of execution notification.

f. For each instruction to modify the periodic generation properties of a parameter report definition of a parameter functional reporting definition that it rejects, the parameter functional reporting configuration subservice shall generate the failed start of execution notification for that instruction.

g. For each request to modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition that contains only valid instructions, the parameter functional reporting configuration subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to modify the periodic generation properties of a parameter report definition of a parameter functional reporting definition, the parameter functional reporting configuration subservice shall:
   1. modify the related parameter report entry by:
      (a) changing the periodic generation action status to the supplied value;
      (b) changing the collection interval to the supplied value.

6.3.5.7 Subservice observables

a. The following observables shall be defined for the parameter functional reporting configuration subservice:
   1. the identifier of the most recent parameter functional reporting definition that has been applied in exclusive mode, if any;
   2. the identifier of the most recent parameter functional reporting definition that has been applied in non-exclusive mode, if any.

6.4 ST[04] parameter statistics reporting

6.4.1 Scope

6.4.1.1 General

The parameter statistics reporting service type provides the capability to evaluate statistics on-board for a list of on-board parameters. The maximum, minimum, mean and standard deviation values of each of these on-board parameters during a time interval is reported to the ground system.

This can, for example, be used to reduce the quantity of engineering data that is systematically reported to the ground system.

This service type is especially appropriate for missions with limited ground coverage (e.g. low Earth orbiter), where a statistics report can be used to provide a summary of the behaviour of parameters during the previous period of "no contact".

The parameter statistics reporting service type defines a single standardized subservice type, i.e. the parameter statistics reporting subservice type.

6.4.1.2 Parameter statistics reporting subservice

The parameter statistics reporting subservice type includes optional capability to modify the list of evaluated parameters and their associated time intervals.

6.4.2 Service layout

6.4.2.1 Subservice

6.4.2.1.1 Parameter statistics reporting subservice

a. Each parameter statistics reporting service shall contain at least one parameter statistics reporting subservice.

6.4.2.2 Application process

a. Each application process shall host at most one parameter statistics reporting subservice provider.

6.4.3 Parameter statistics definition

6.4.3.1 General

a. The maximum number of parameter statistics definitions that the parameter statistics reporting subservice can contemporaneously evaluate at any time shall be declared when specifying that subservice.

b. Each parameter statistics definition shall contain:
   1. the identification of the parameter for which statistics are evaluated;
   2. the related sampling interval.

c. For each parameter, at most one parameter statistics definition shall be used at any time by the parameter statistics reporting subservice for that parameter.

6.4.3.2 Statistic types

a. The parameter statistics reporting subservice shall support the evaluation of the following statistic types:
   1. the maximum value evaluation statistic type;
   2. the minimum value evaluation statistic type;
   3. the mean value evaluation statistic type.

b. Whether the parameter statistics reporting subservice supports the standard deviation evaluation statistic type shall be declared when specifying that subservice.

c. For each parameter for which statistics are evaluated, the parameter statistics reporting subservice shall evaluate, at any time, all supported types of statistics.

6.4.3.3 Sampling interval

a. For each parameter that statistics are evaluated, the default sampling interval that the parameter statistics reporting subservice uses for that parameter shall be declared when specifying that subservice.

NOTE Refer to requirement 5.4.3.2.b for the on-board parameter minimum sampling interval.

6.4.4 Reset the parameter statistics

a. The parameter statistics reporting subservice shall provide the capability to reset the parameter statistics evaluation on request.

NOTE 1 The corresponding requests are of message type "TC[4,3] reset the parameter statistics".

NOTE 2 In this case, the resetting of the parameter statistics is independent of the generation of a parameter statistics report.

b. Each request to reset the parameter statistics shall contain exactly one instruction to reset the parameter statistics.

NOTE The instructions to reset the parameter statistics contain no argument.

c. For each valid instruction to reset the parameter statistics, the parameter statistics reporting subservice shall:
   1. stop the evaluation of parameter statistics;
   2. clear any results accumulated;
   3. restart the evaluation process.

NOTE The resetting of the parameter statistics can also result from the request to report the parameter statistics (refer to clause 6.4.5.1).

6.4.5 On-request parameter statistics reporting

6.4.5.1 Capability

a. The parameter statistics reporting subservice shall provide exactly one of the following capabilities:
   1. the capability to explicitly state in each request to report the parameter statistics, whether or not to reset the parameter statistics after the generation of the parameter statistics report;
   2. the capability to automatically reset the parameter statistics after responding to each request to report the parameter statistics.

b. Whether the parameter statistics reporting subservice provides the capability to explicitly state in each request to report the parameter statistics, whether or not to reset the parameter statistics after the generation of the parameter statistics report shall be declared when specifying that subservice.

c. Whether the parameter statistics reporting subservice provides the capability to automatically reset the parameter statistics after responding to each request to report the parameter statistics shall be declared when specifying that subservice.

6.4.5.2 Report the parameter statistics

a. The parameter statistics reporting subservice shall provide the capability for on-request reporting of the results of the parameter statistics evaluation.

   NOTE 1 The corresponding requests are of message type "TC[4,1] report the parameter statistics". The responses are data reports of message type "TM[4,2] parameter statistics report" (refer to clause 6.4.5.3).

   NOTE 2 Parameter statistics reports are also generated by the periodic parameter statistics reporting specified in clause 6.4.6.

b. Each request to report the parameter statistics shall contain:
   1. if the subservice provides the capability to explicitly state in each request to report the parameter statistics, whether or not to reset the parameter statistics after the generation of the parameter statistics report, the resetting indication.
   2. exactly one instruction to report the parameter statistics.

      NOTE 1 For the capability in item 1, refer to requirement 6.4.5.1b.

      NOTE 2 The instructions to report the parameter statistics contain no argument.

c. For each valid instruction to report the parameter statistics, the parameter statistics reporting subservice shall generate a single parameter statistics report.

   NOTE For the parameter statistics report, refer to clause 6.4.5.3.

d. For each valid request to report the parameter statistics, after executing the instruction to report the parameter statistics, the parameter statistics reporting subservice shall reset the parameter statistics if:

1. that request explicitly indicates that reset, or
2. that subservice is configured to automatically reset the evaluation of the parameter statistics after responding to each request to report the parameter statistics.

NOTE 1 For item 1, refer to requirement 6.4.5.1b.

NOTE 2 For item 2, refer to requirement 6.4.5.1c.

6.4.5.3 Parameter statistics report

a. The parameter statistics reporting subservice shall provide the capability to generate parameter statistics reports.

NOTE 1 The corresponding reports are data reports of message type "TM[4,2] parameter statistics report".

NOTE 2 Parameter statistics reports are generated in response to the requests to report parameter statistics specified in clause 6.4.5.2. They are also generated by the periodic parameter statistics reporting specified in clause 6.4.6.

b. When generating a parameter statistics report the parameter statistics reporting subservice shall generate a single parameter statistic notification for each parameter for which the subservice has sampled at least one value since the statistics were last reset.

c. Each parameter statistic notification shall contain:

1. the identifier of the sampled parameter;
2. the number of samples used to produce the statistics;
3. the maximum value that has been sampled during the time interval and the time at which this maximum sampled value was first attained;
4. the minimum value that has been sampled during the time interval and the time at which this minimum sampled value was first attained;
5. the mean of the sampled values during the time interval;
6. if the parameter statistics reporting subservice supports the evaluation of the standard deviation, the standard deviation of the sampled values during the time interval.

NOTE For the item 6 evaluation of the standard deviation support, refer to requirement 6.4.3.2b.

d. Each parameter statistics report shall contain:

1. the start time and the end time of the time interval over which the evaluation of the parameter statistics was performed;
2. all related parameter statistic notifications.

6.4.6 Periodic parameter statistics reporting

6.4.6.1 General

a. Whether the parameter statistics reporting subservice supports for the periodic reporting of the results of the parameter statistics evaluation shall be declared when specifying that subservice.

b. The periodic reporting interval that corresponds to the time interval after which the parameter statistics reporting subservice reports and resets the statistics shall either:

   1. be implicitly known by that subservice, or
   2. be specified in each request to enable the periodic parameter reporting.

c. If the parameter statistics subservice implicitly knows the periodic reporting interval, that interval shall be declared when specifying that subservice.

d. Whether the parameter statistics subservice provides the capability to explicitly state in each request to enable the periodic parameter reporting the periodic reporting interval shall be declared when specifying that subservice.

e. The parameter statistics reporting subservice shall maintain a status indicating whether the periodic parameter statistics reporting is enabled or disabled.

   NOTE This status is named "periodic parameter statistics reporting status".

6.4.6.2 Enable the periodic parameter statistics reporting

a. The parameter statistics reporting subservice shall provide the capability to enable the periodic parameter statistics reporting if that subservice supports reporting periodically the results of the parameter statistics evaluation.

   NOTE 1 The corresponding requests are of message type "TC[4,4] enable the periodic parameter reporting".

   NOTE 2 For the support to report periodically the results of the parameter statistics evaluation, refer to requirement 6.4.6.1a.

   NOTE 3 For the capability to disable the periodic parameter statistics reporting, refer to clause 6.4.6.3.

b. Each request to enable the periodic parameter statistics reporting shall contain exactly one instruction to enable the periodic parameter statistics reporting.

c. Each instruction to enable the periodic parameter statistics reporting shall contain:

   1. if the subservice is configured for the capability in requirement 6.4.6.1d, the periodic reporting interval.

d. The parameter statistics reporting subservice shall reject any request to enable the periodic parameter statistics reporting if:

1. that request contains an instruction that specifies a reporting interval that is smaller than the sampling interval of any parameter for which statistics are evaluated.

e. For each request to enable the periodic parameter statistics reporting that is rejected, the parameter statistics reporting subservice shall generate a failed start of execution notification.

f. For each valid instruction to enable the periodic parameter statistics reporting, the parameter statistics reporting subservice shall:

1. set the periodic parameter statistics reporting status to "enabled";
2. if the instruction specifies a reporting interval, set the periodic reporting interval to the specified interval.

g. During the entire enabled periodic reporting duration, the parameter statistics reporting subservice shall generate exactly one parameter statistics report at the end of each reporting interval period.

NOTE For the parameter statistics report, refer to clause 6.4.5.3.

h. The parameter statistics reporting subservice shall systematically reset the parameter statistics evaluation whenever a periodic parameter statistics report is generated.

6.4.6.3 Disable the periodic parameter statistics reporting

a. The parameter statistics reporting subservice shall provide the capability to disable the periodic parameter statistics reporting if that subservice supports reporting periodically the results of the parameter statistics evaluation.

NOTE 1 The corresponding requests are of message type "TC[4,5] disable the periodic parameter statistics reporting".

NOTE 2 For the support to report periodically the results of the parameter statistics evaluation, refer to requirement 6.4.6.1a.

NOTE 3 For the capability to enable the periodic parameter statistics reporting, refer to clause 6.4.6.2.

b. Each request to disable the periodic parameter statistics reporting shall contain exactly one instruction to disable the periodic parameter statistics reporting.

NOTE The instructions to disable the periodic parameter statistics reporting contain no argument.

c. For each valid instruction to disable the periodic parameter statistics reporting, the parameter statistics reporting subservice shall:

1. set the periodic parameter statistics reporting status to "disabled".

6.4.7 Maintaining the list of evaluated parameters

6.4.7.1 Add or update parameter statistics definitions

a. The parameter statistics reporting subservice capability to add or update parameter statistics definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[4,6] add or update parameter statistics definitions".

NOTE 2 For the capability to delete parameter statistics definitions, refer to clause 6.4.7.2.

b. Whether the setting of the sampling interval in the instructions to add or update a parameter statistics definition is supported shall be declared when specifying the parameter statistics reporting subservice.

NOTE Parameters can be sampled at quite different frequencies, depending on the particular characteristics of the parameter. For example, a rapidly varying parameter such as gyro output may be sampled at a high frequency whilst a slowly varying analogue parameter such as a temperature may be sampled at a very low frequency.

c. Each request to add or update parameter statistics definitions shall contain an ordered list of one or more instructions to add or update a parameter statistics definition.

d. Each instruction to add or update a parameter statistics definition shall contain:

1. the parameter identifier;
2. if sampling intervals are supported as specified in requirement 6.4.7.1b, the sampling interval.

e. The parameter statistics reporting subservice shall reject any request to add or update parameter statistics definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a parameter that is unknown;
2. that request contains an instruction that contains a sampling interval is greater than the reporting interval;
3. that request contains an instruction that implies adding a parameter statistics definition but the maximum number of definitions that the subservice supports is already reached.

NOTE For item 3, refer to requirement 6.4.3.1a.

f. For each request to add or update parameter statistics definitions that it is rejected, the parameter statistics reporting subservice shall generate a failed start of execution notification for that instruction.

g. For each request to add or update parameter statistics definitions that contains only valid instructions, the parameter statistics reporting

subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to add or update a parameter statistics definition, the parameter statistics reporting subservice shall:

1. if no parameter statistics definition exists for that parameter:
   (a) add the parameter statistics definition to the list of evaluated parameters;
   (b) start the evaluation of the statistics for that parameter;
2. if a parameter statistics definition exists for that parameter:
   (a) update the sampling interval of that parameter statistics definition;
   (b) restart the evaluation of the statistics for that parameter.

NOTE 1 The evaluation of the statistics starts immediately, i.e. independently of the reporting interval. Within the next report (and only that report), a parameter whose parameter statistics definition was added during the previous reporting interval is reported over a shorter interval than parameters that were already in the list.

NOTE 2 If a request contains two instructions to add or update a parameter statistics definition for the same parameter, the second instruction overrides the effect of the first instruction,

6.4.7.2 Delete parameter statistics definitions

a. The parameter statistics reporting subservice shall provide the capability to delete parameter statistics definitions if the capability to add or update parameter statistics definitions is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[4,7] delete parameter statistics definitions".

NOTE 2 For the capability to add or updates parameter statistics definitions, refer to clause 6.4.7.1.

b. Each request to delete parameter statistics definitions shall contain exactly one of:

1. an ordered list of one or more instructions to delete a parameter statistics definition;
2. a single instruction to delete all parameter statistics definitions.

NOTE The instructions to delete all parameter statistics definitions contain no argument.

c. Each instruction to delete a parameter statistics definition shall contain:

1. the parameter identifier.

d. The parameter statistics reporting subservice shall reject any request to delete parameter statistics definitions if:

1. that request contains an instruction that refers to a parameter that is not in the list of evaluated parameters.

e. For each request to delete parameter statistics definitions that is rejected, the parameter statistics reporting subservice shall generate a failed start of execution notification.

f. For each request to delete parameter statistics definitions that contains only valid instructions, the parameter statistics reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a parameter statistics definition, the parameter statistics reporting subservice shall:

1. remove that parameter statistics definition from the list of evaluated parameters.

h. For each valid instruction to delete all parameter statistics definitions, the parameter statistics reporting subservice shall:

1. remove all parameter statistics definitions from the list of evaluated parameters, if any.

i. For each valid instruction to delete parameter statistics definitions, if the list of evaluated parameters is empty after the execution of the instructions, the parameter statistics reporting subservice shall:

1. set the periodic parameter statistics reporting status to "disabled" if the list of evaluated parameters is empty after execution of all instructions.

6.4.7.3 Report the parameter statistics definitions

a. The parameter statistics reporting subservice capability to report the parameter statistics definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[4,8] report the parameter statistics definitions". The responses are data reports of message type "TM[4,9] parameter statistics definition report".

NOTE 2 That capability requires the capability for that subservice to add or updates parameter statistics definitions, refer to clause 6.4.7.1.

b. Each request to report the parameter statistics definitions shall contain exactly one instruction to report the parameter statistics definitions.

NOTE The instructions to report the parameter statistics definitions contain no argument.

c. For each valid instruction to report the parameter statistics definitions, the parameter statistics reporting subservice shall generate a single parameter statistics definition notification that includes:

1. if the parameter statistics reporting subservice permits changing the periodic reporting interval, the current periodic reporting interval;
2. for each parameter statistics definition in the list of evaluated parameters:
   (a) the parameter identifier;
   (b) if sampling intervals are supported as specified in requirement 6.4.7.1b, the sampling interval.

NOTE For item 1 permission to change the periodic reporting interval, refer to requirement 6.4.6.1d.

d. For each valid request to report the parameter statistics definitions, the parameter statistics reporting subservice shall generate a single parameter statistics definition report that includes the related parameter statistics definition notification.

6.4.8 Subservice observables

a. The following observables shall be defined for the parameter statistics reporting subservice:

1. the number of parameter statistics definitions currently defined on-board.

6.5 ST[05] event reporting

6.5.1 Scope

6.5.1.1 General

The event reporting service type provides the capability to report information of operational significance that is not explicitly provided within the provider-initiated reports of another service.

The service covers the requirements for reporting of the occurrences of events such as:

- on-board failures and anomalies, including anomalies detected by a failure detection, isolation and recovery (FDIR) function;
- initiation, progress and completion of activities initiated either from ground or autonomously on-board;
- hardware device built-in test results;
- normal payload events.

The event reporting service type defines a single standardized subservice type, i.e. the event reporting subservice type.

6.5.1.2 Event reporting subservice

Each event that occurrences can be caught by the event reporting subservice and reported is associated to an event report type. Each event report type specifies the severity level of the event to report (informative, low severity, medium severity or high severity). To facilitate ground system detection and routing the severity level is indicated by the message type of the generated report.

Each event is also associated to an event definition. An event definition is identified by an event definition identifier that is unique within the application process that generates the corresponding event reports. Auxiliary data can be associated to each event definition to report the context and the cause of the event occurrence.

The event reporting subservice type includes optional capabilities to:

- selectively enable and disable the generation of its event reports;
- generate an event report on request.

6.5.2 Service layout

6.5.2.1 Subservice

6.5.2.1.1 Event reporting subservice

a. Each event reporting service shall contain at least one event reporting subservice.

6.5.2.2 Application process

a. Each application process shall host at most one event reporting subservice provider.

6.5.3 Event definitions

a. The list of events that can be detected by the event reporting subservice shall be declared when specifying that subservice.

b. The event list shall include at a minimum all asynchronous error conditions that can be raised by any subservice not otherwise reported by the request verfication service.

NOTE For example, a telecommand not dispatched on time by the time-based scheduling subservice, or the loss of data on file swapping by the file data storage subservice.

c. For each event that can be detected by the event reporting subservice, the event definition used to report on the occurrences of that event, the related event severity level, the event definition identifier and, if any, auxiliary data shall be declared when specifying that subservice.

NOTE The event severity levels are:

- informative;
- low severity;
- medium severity;
- high severity.

d. Each event definition shall be uniquely identified by the combination of the identifier of the application process that hosts the event reporting subservice provider that is in charge to report on the occurrences of the associated event and an event definition identifier.

NOTE The term "event definition system identifier" is used in this standard to represent that combination of application process identifier and event definition identifier.

6.5.4 Event reporting

a. The event reporting subservice shall provide the capability to generate event reports.

NOTE The corresponding event reports are of message type:

- "TM[5,1] informative event report";
- "TM[5,2] low severity anomaly report";
- "TM[5,3] medium severity anomaly report";
- "TM[5,4] high severity anomaly report".

b. The destination of the event reports generated by the event reporting subservice shall be declared when specifying that subservice.

NOTE All the event reports generated by an event reporting subservice have the same destination.

c. If the event reporting subservice supports the capability for controlling the generation of event reports specified in clause 6.5.5, that subservice shall generate an event notification whenever it detects the occurrence of an event associated to an event definition for which event report generation is enabled.

d. If the event reporting subservice does not support the capability for controlling the generation of event reports specified in clause 6.5.5, that subservice shall generate an event notification whenever it detects the occurrence of an event.

e. Each event notification shall contain:

1. the event definition identifier of the associated event definition;
2. the auxiliary data associated to that event definition, if any.

NOTE For item 2, refer to requirement 6.5.3c.

f. For each event notification that it generates, the event reporting subservice shall generate an event report of the related event severity level, which contains that notification.

NOTE The message subtype identifier of the event report message type indicates the event severity level, refer to requirement 6.5.4a.

6.5.5 Controlling the generation of event reports

6.5.5.1 Event report generation status

a. For each event that can be detected by the event reporting subservice, a status indicating whether the event report generation for that event is enabled or disabled shall be maintained

NOTE This is called "event report generation status"

b. The event reporting subservice configuration policy to apply when starting and restarting the event reporting subservice shall be declared when specifying the subservice.

NOTE 1 this includes, for each event that can be detected by the event reporting subservice, the initial enabled or disabled event report generation status

NOTE 2 in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

6.5.5.2 Enable the report generation of event definitions

a. The event reporting subservice capability to enable the report generation of event definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[5,5] enable the report generation of event definitions".

NOTE 2 The event reports generated on-board are for use by the ground but can also be used by on-board functions such as those implemented within event action services or OBCP engines.

NOTE 3 For the capability to disable the report generation of event definitions, refer to clause 6.5.5.3.

b. Each request to enable the report generation of event definitions shall contain an ordered list of one or more instructions to enable the report generation of an event definition.

c. Each instruction to enable the report generation of an event definition shall contain:

1. the event definition identifier of the event definition to enable.

NOTE For the event definition identifier, refer to requirement 6.5.3c.

d. The event reporting subservice shall reject any request to enable the report generation of event definitions if:

1. that request contains an instruction that refers to an unknown event definition.

e. For each request to enable the report generation of event definitions that is rejected, the event reporting subservice shall generate a failed start of execution notification.

f. For each request to enable the report generation of event definitions that contains only valid instructions, the event reporting subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable the report generation of an event definition, the event reporting subservice shall:

1. set the event report generation status of the event definition to "enabled".

6.5.5.3 Disable the report generation of event definitions

a. The event reporting subservice shall provide the capability to disable the report generation of event definitions if the capability to enable the report generation of event definitions is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[5,6] disable the report generation of event definitions".

NOTE 2 For example, event reporting can be disabled to reduce the on-board processing load.

NOTE 3 Disabling the report generation of an event definition implies that the event reporting subservice does not inform the ground about the raising occurrences of the related event. The on-board services that are configured to react to the corresponding event reports are also not triggered. Disabling the report generation of an event definition does not mean that the raising

occurrences of the related event cannot be directly (meaning without the needs for event reports) caught by e.g. the on-board software.

NOTE 4 For the capability to enable the report generation of event definitions, refer to clause b.

b. Each request to disable the report generation of event definitions shall contain an ordered list of one or more instructions to disable the report generation of an event definition.

c. Each instruction to disable the report generation of an event definition shall contain:

1. the event definition identifier of the event definition to disable.

d. The event reporting subservice shall reject any request to disable the report generation of event definitions if:

1. that instruction refers to an unknown event definition.

e. For each request to disable the report generation of event definitions that is rejected, the event reporting subservice shall generate a failed start of execution notification.

f. For each request to disable the report generation of event definitions that contains only valid instructions, the event reporting subservicce shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable the report generation of an event definition, the event reporting subservice shall:

1. set the event report generation status of the event definition to "disabled".

6.5.5.4 Report the list of disabled event definitions

a. The event reporting subservice capability to report the list of disabled event definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[5,7] report the list of disabled event definitions". The responses are data reports of message type "TM[5,8] disabled event definitions list report".

NOTE 2 That capability requires the capability for that subservice to enable the report generation of event definitions (refer to clause b).

b. Each request to report the list of disabled event definitions shall contain exactly one instruction to report the list of disabled event definitions.

NOTE The instructions to report the list of disabled event definitions contain no argument.

c. For each valid instruction to report the list of disabled event definitions, the event reporting subservice shall:

1. generate, for each event definition whose event report generation status is "disabled", a single disabled event definition notification that includes:

(a) the related event definition identifier.

d. For each valid request to report the list of disabled event definitions, the event reporting subservice shall generate a single disabled event definitions list report that includes all related disabled event definition notifications.

6.5.5.5 Generate event report

a. The event reporting subservice capability to generate an event report on request shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[5,9] generate event report".

b. Each request to generate an event report shall contain exactly one instruction to generate an event report.

c. Each instruction to request the generation of an event report shall contain:

1. event APID
2. event severity
3. event definition identifier
4. event data, if any

d. The event reporting subservice shall reject any instruction to generate an event report if any of the following conditions occurs:

1. that instruction refers to an invalid event APID
2. that instruction refers to an invalid event severity
3. that instruction refers to an invalid event definition identifier

e. For each request to generate an event report that is rejected, the event reporting subservice shall generate a failed start of execution notification.

f. For each valid instruction to request the generation of an event report, the event reporting subservice shall:

1. generate the requested event report telemetry packet

6.5.6 Subservice observables

a. The following observables shall be defined for the event reporting subservice:

1. per severity level:
   (a) the accumulated number of event occurrences,
   (b) the number of event definitions whose event report generation status is "disabled",
   (c) the accumulated number of generated event reports,
   (d) the event definition identifier of the last generated event report,
   (e) the generation time of the last event report.

6.6 ST[06] memory management

6.6.1 Scope

6.6.1.1 General

The term "memory" (see also clause5.4.3.3) is used to logically refer to any physical or virtual memory area which exists on-board the spacecraft, e.g. RAM, mass memory unit.

The memory management service provides the capability for loading, dumping and checking the contents of these memories without precluding that the same memory is used by more than one memory management service or that overlapping memories are used on-board.

The memory management service type defines four standardized subservice types, i.e.:

- the raw data memory management subservice type;
- the structured data memory management subservice type;
- the common memory management subservice type;
- the memory configuration subservice type.

6.6.1.2 Raw data memory management subservice

The raw data memory management subservice type provides capabilities to manage memories that contain raw data.

Raw data means that:

- the type of memory content is not implicitly known, and
- addressing data within that memory implies pointing to a memory offset from the start of that memory, i.e. a start address.

6.6.1.3 Structured data memory management subservice

The structured data memory management subservice type provides capabilities to manage memories that contain structured data.

Structured data means that:

- the type of memory content is implicitly known, and
- addressing an object of that memory implies using a base that refers to the starting area of that object, i.e. the object location and an offset from that object location.

The type of memory content can itself be:

- generic, e.g. the memory contains files or
- specific, e.g. the memory contains on-board control procedures.

6.6.1.4 Common memory management subservice

The common memory management subservice type provides capabilities to manage functions that are common to the raw data memory management and the structured data memory management subservice types. In this Standard, this is the case for the "abort all memory dumps" capability that interacts with both data memory management related subservices to abort all dump requests that are under execution.

6.6.1.5 Memory configuration subservice

The memory configuration subservice type provides capabilities for managing the memory as a whole, i.e. independently of its content and a specific addressing scheme. For example, the subservice type includes the capability for enabling and disabling the scrubbing of memories and for write protecting memories.

6.6.2 Service layout

6.6.2.1 Subservice

6.6.2.1.1 General

a. Each memory management service shall contain at least one of:
   1. the raw data memory management subservice;
   2. the structured data memory management subservice.

6.6.2.1.2 Raw data memory management subservice

a. Each memory management service shall contain at most one raw data memory management subservice.

6.6.2.1.3 Structured data memory management subservice

a. Each memory management service shall contain at most one structured data memory management subservice.

6.6.2.1.4 Common memory management subservice

a. Each memory management service shall contain at most one common memory management subservice.

6.6.2.1.5 Memory configuration subservice

a. Each memory management service shall contain at most one memory configuration subservice.

6.6.2.2 Application process

a. All subservice providers of the memory management service shall be hosted by a single application process.

b. Each application process shall host subservices providers belonging to at most one management service.

6.6.3 Raw data memory management subservice

6.6.3.1 Checksumming

a. Whether the raw data memory management subservice provides checksumming shall be declared when specifying that subservice.

NOTE For the checksum algorithm, refer to clause 6.6.3.

6.6.3.2 Memory accessibility

a. The list of memories managed by the raw data memory management subservice shall be declared when specifying that subservice.

NOTE Refer also to clause5.4.3.3.

b. Each memory managed by the raw data memory management subservice shall use the absolute addressing scheme.

c. If the raw data memory management subservice manages more than one memory, the subservice shall use memory identifiers.

d. For each writeable memory that it manages, whether the raw data memory management subservice has write access to that memory shall be declared when specifying that subservice.

6.6.3.3 Load raw memory

6.6.3.3.1 Load raw memory data areas

a. The raw data memory management subservice shall provide the capability to load raw memory data areas.

NOTE The corresponding requests are of message type "TC[6,2] load raw memory data areas".

b. Each request to load raw memory data areas shall contain:

1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;
2. an ordered list of one or more instructions to load a raw memory data area.

NOTE 1 For item 1, refer to requirement 6.6.3.2a. If the raw data memory management subservice manages only one memory, the instructions apply to that memory.

NOTE 2 All the instructions in the request apply to the same memory.

c. The execution verification profile of each request to load raw memory data areas shall include the reporting of the completion of execution.

NOTE For the execution verification profile, refer to requirement 5.3.7.4.

d. Each instruction to load a raw memory data area shall contain:

1. the start address of where to load the data, expressed as a byte pointer aligned on the memory access alignment constraint;

2. the data to load;
3. if the raw data memory management subservice provides checksumming, the checksum value for the verification of the data after it has been loaded to the memory.

   NOTE For item 3, refer to requirement 6.6.3.1a.

e. The raw data memory management subservice shall reject any request to load raw memory data areas if any of the following conditions occurs:
   1. that request refers to an invalid memory identifier;
   2. the subservice does not have write access to the memory referred to in that request;
   3. that request refers to a memory that is write protected;
   4. that request contains an instruction that refers to:
      (a) a start address that exceeds the maximum memory size;
      (b) a start address which is not aligned with respect to the memory access alignment constraint;
      (c) a load length which is not a multiple of the memory access alignment constraint;
   5. loading the data contained in one of the related instructions exceeds the maximum memory size.

f. For each request to load raw memory data areas that is rejected, the raw data memory management subservice shall generate a failed start of execution notification.

g. For each request to load raw memory data areas that contains only valid instructions, the raw data memory management subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to load a raw memory data area, the raw data memory management subservice shall:
   1. write the data to memory.

i. If an error occurs during the writing to memory of the data related to an instruction to load a raw memory data area, the raw data memory management subservice shall:
   1. immediately abort the execution of the related request;
   2. generate a failed execution notification for that instruction.

      NOTE For example, an error can occur when the memory becomes write-protected by hardware during the course of the load operation.

j. If the subservice provides checksumming, then once the data related to an instruction to load a raw memory data area has been written to the memory, the raw data memory management subservice shall:
   1. calculate the checksum of the loaded data;
   2. compare it to the checksum value in that instruction;
   3. if that checksum comparison fails:
      (a) immediately abort the execution of the related request;
      (b) generate a failed execution notification for that instruction.

k. For each request to load raw memory data areas that is aborted, the raw data memory management subservice shall generate a failed completion of execution verification report that contains the failed execution notification.

6.6.3.3.2 Load a raw memory atomic data area in a non-interruptible transaction

a. The raw data memory management subservice capability to load a raw memory atomic data area in a non-interruptible transaction shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[6,11] load a raw memory atomic data area in a non-interruptible transaction".

b. Each request to load a raw memory atomic data area in a non-interruptible transaction shall contain exactly one instruction to load a raw memory atomic data area in a non-interruptible transaction.

c. The execution verification profile of each request to load a raw memory atomic data area in a non-interruptible transaction shall include the reporting of the completion of execution.

NOTE For the execution verification profile, refer to requirement 5.3.7.4.

d. Each instruction to load a raw memory atomic data area in a non-interruptible transaction shall contain:

1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;
2. the address of where to load the data, expressed as a byte pointer aligned on the memory access alignment constraint;
3. the bit mask, with length equal to the memory access alignment constraint;
4. the data to load, with length equal to the memory access alignment constraint.

NOTE 1 For item 1, refer to requirement 6.6.3.2a.

NOTE 2 For items 3 and 4:

- The bit mask is applied to the addressed memory area to identify the bits of that memory area impacted by the load request. The data to load is then applied to those bits.
- The value in the bit mask and the value in the data to load are each less than or equal to the maximum value that can be expressed using the access alignment constraint of that memory.

e. The raw data memory management subservice shall reject any request to load a raw memory atomic data area in a non-interruptible transaction if any of the following conditions occurs:

1. the subservice does not have both read and write access to the memory referred to in that request;

2. that request contains an instruction that refers to a memory identifier that is unknown;
3. that request contains an instruction that refers to a memory that is write protected;
4. that request contains an instruction that refers to a start address that exceeds the maximum memory size;
5. that request contains an instruction that refers to a start address which is not aligned with the memory access alignment constraint;
6. the deduced size of the bit mask and the data to load does not match the overall size of the request.

f. For each request to load a raw memory atomic data area in a non-interruptible transaction that is rejected, the raw data memory management subservice shall generate a failed start of execution notification.

g. For each valid instruction to load a raw memory atomic data area in a non-interruptible transaction, the raw data memory management subservice shall:
1. extract the current value of the memory area addressed by the instruction;
2. compute the new value of the atomic data by updating the bits that are selected by the mask to the value specified in the data to load;
3. set the memory area to that new value.

NOTE For item 1, the memory area addressed by the instruction is the memory area that is at the start address and has a size equal to the access alignment constraint of that memory.

h. If an error occurs during the writing to memory of the data related to an instruction load a raw memory atomic data area in a non-interruptible transaction, the raw data memory management subservice shall:
1. generate a failed execution notification for that instruction.

i. For each request to load a raw memory atomic data area in a non-interruptible transaction that is aborted, the raw data memory management subservice shall generate a failed completion of execution verification report that contains the failed execution notification.

6.6.3.4 Dump raw memory data

a. The raw data memory management subservice shall provide the capability to dump raw memory data.

NOTE The corresponding requests are of message type "TC[6,5] dump raw memory data". The responses are data reports of message type "TM[6,6] dumped raw memory data report".

b. Each request to dump raw memory data shall contain:
1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;

2. an ordered list of one or more instructions to dump a raw memory data.

NOTE For item 1, refer to requirement 6.6.3.2a. If the raw data memory management subservice manages only one memory, the instructions apply to that memory.

c. Each instruction to dump a raw memory data shall contain:

1. the start address of the memory area to dump, expressed as a byte pointer aligned on the memory access alignment constraint;
2. the octet length of the memory area to dump.

d. The raw data memory management subservice shall reject any request to dump a raw memory data if any of the following conditions occurs:

1. that request refers to an invalid memory identifier;
2. the subservice does not have read access to the memory referred to in that request;
3. that instruction refers to a start address that exceeds the maximum memory size;
4. that instruction refers to a start address which is not aligned with the memory access alignment constraint;
5. that instruction refers to a length that combined with the start address exceeds the maximum memory size;
6. that instruction refers to a length that is not a multiple of the memory access alignment constraint.

e. For each request to dump a raw memory data that it is rejected, the raw data memory management subservice shall generate the failed start of execution notification.

f. For each request to dump raw memory data that contains only valid instructions, the raw data memory management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to dump a raw memory data, the raw data memory management subservice shall:

1. extract the memory data specified by that instruction from the memory;
2. if the subservice provides checksumming, calculate the checksum of the extracted memory data;
3. generate a single dumped raw memory data notification that includes:
   (a) the start address of the memory area, expressed as a byte pointer aligned on the memory access alignment constraint;
   (b) the dumped data;
   (c) if the subservice provides checksumming, the calculated checksum of that dumped area.

NOTE For item 3(c), refer to requirement 6.6.3.1a.

h. For each valid request to dump raw memory data, the raw data memory management subservice shall generate a single dumped raw memory data report that contains:

1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;
2. all related dumped raw memory data notifications.

NOTE For item 1, refer to requirement 6.6.3.2a.

6.6.3.5 Check raw memory data

a. The raw data memory management subservice capability to check raw memory data shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[6,9] check raw memory data". The responses are data reports of message type "TM[6,10] checked raw memory data report".

NOTE 2 Checking memory and reporting the calculated checksum for ground checksum comparison, avoids downlinking on-board memory areas that are suspected to be faulty.

b. Each request to check raw memory data shall contain:

1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;
2. an ordered list of one or more instructions to check a raw memory data.

NOTE For item 1, refer to requirement 6.6.3.2a. If the raw data memory management subservice manages only one memory, the instructions apply to that memory.

c. Each instruction to check a raw memory data shall contain:

1. the start address of the memory area to check, expressed as a byte pointer aligned on the memory access alignment constraint;
2. the octet length of the memory area to check

d. The raw data memory management subservice shall reject any request to check raw memory data if any of the following conditions occurs:

1. that request refers to a memory identifier that is unknown;
2. the subservice does not have read access to the memory referred to in that request.
3. that request contains an instruction that refers to a start address that exceeds the maximum memory size;
4. that request contains an instruction that refers to a start address which is not aligned with the memory access alignment constraint;
5. that request contains an instruction that refers to a length which is not a multiple of the memory access alignment constraint;
6. that instruction refers to a length that combined with the start address exceeds the maximum memory size.

e. For each request to check raw memory data that is rejected, the raw data memory management subservice shall generate a failed start of execution notification.

f. For each request to check raw memory data that contains only valid instructions, the raw data memory management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to check a raw memory data, the raw data memory management subservice shall:

1. calculate the checksum of the memory area specified by that instruction;
2. generate a single checked raw memory data notification that includes:
   (a) the start address of the memory area, expressed as a byte pointer aligned on the memory access alignment constraint;
   (b) the octet length of the checked memory area;
   (c) the calculated checksum of that memory area.

h. For each valid request to check raw memory data, the raw data memory management subservice shall generate a single checked raw memory data report that contains:

1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;
2. all related checked raw memory data notifications.

NOTE For item 1, refer to requirement 6.6.3.2a.

6.6.3.6 Load raw memory data areas by reference

a. The raw data memory management subservice capability to load raw memory data areas by reference shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[6,19] load raw memory data areas by reference".

b. Each request to load raw memory data areas by reference shall contain:

1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;
2. the file path of the file containing the data to load;
3. an ordered list of one or more instructions to load a raw memory data area by reference.

NOTE 1 For item 1, refer to requirement 6.6.3.2a. If the raw data memory management subservice manages only one memory, the instructions apply to that memory.

NOTE 2 All the instructions in the request apply to the same memory and to the same file.

c. The execution verification profile of each request to load raw memory data areas by reference shall include the reporting of the completion of execution.

NOTE For the execution verification profile, refer to requirement 5.3.6.2.3f.

d. Each instruction to load a raw memory data area by reference shall contain:

1. the start address of where to load the data, expressed as a byte pointer aligned on the memory access alignment constraint;
2. the offset in bytes of the data in the source file;
3. the length in bytes of the data to load;
4. if the raw data memory management subservice provides checksumming, the checksum value for the verification of the data after it has been loaded to the memory.

NOTE For item 4, refer to requirement 6.6.3.1a.

e. The raw data memory management subservice shall reject any request to load raw memory data areas by reference if any of the following conditions occurs:

1. that request refers to an invalid memory identifier;
2. the subservice does not have write access to the memory referred to in that request;
3. that request refers to a memory that is write protected;
4. that request refers to a file that does not exist;
5. that request refers to a file that is not recognized as a file containing memory data;
6. that request contains an instruction that refers to:
   (a) a start address that exceeds the maximum memory size;
   (b) a start address which is not aligned with respect to the memory access alignment constraint;
   (c) a load length which is not a multiple of the memory access alignment constraint;
   (d) an offset that exceeds the source file size;
7. loading the data contained in one of the related instructions exceeds the maximum memory size.

f. For each request to load raw memory data areas by reference that is rejected, the raw data memory management subservice shall generate a failed start of execution notification.

g. For each request to load raw memory data areas by reference that contains only valid instructions, the raw data memory management subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to load a raw memory data area by reference, the raw data memory management subservice shall:

1. read the data from the source file;
2. write the data to memory.

i. If an error occurs during the writing to memory of the data related to an instruction to load a raw memory data area by reference, the raw data memory management subservice shall:

1. immediately abort the execution of the related request;
2. generate a failed execution notification for that instruction.

NOTE For example, an error can occur when the memory becomes write-protected by hardware during the course of the load operation.

j. If the subservice provides checksumming, then once the data related to an instruction to load a raw memory data area by reference has been written to the memory, the raw data memory management subservice shall:

1. calculate the checksum of the loaded data;
2. compare it to the checksum value in that instruction;
3. if that checksum comparison fails:
   (a) immediately abort the execution of the related request;
   (b) generate a failed execution notification for that instruction.

k. For each request to load raw memory data areas by reference that is aborted, the raw data memory management subservice shall generate a failed completion of execution verification report that contains the failed execution notification.

6.6.3.7 Dump raw memory data areas to file

a. The raw data memory management subservice capability to dump raw memory data areas to file shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[6,20] dump raw memory data areas to file".

b. Each request to dump raw memory data areas to file shall contain:

1. if the raw data memory management subservice manages more than one memory, the identifier of the memory;
2. the object path of the destination file;
3. an ordered list of one or more instructions to dump a raw memory data area to file.

NOTE 1 For item 1, refer to requirement 6.6.3.2a. If the raw data memory management subservice manages only one memory, the instructions apply to that memory.

c. Each instruction to dump a raw memory data area to file shall contain:

1. the start address of the memory area to dump, expressed as a byte pointer aligned on the memory access alignment constraint;
2. the octet length of the memory area to dump.

d. The raw data memory management subservice shall reject any request to dump raw memory data areas to file if any of the following conditions occurs:

1. that request refers to an invalid memory identifier;

2. the subservice does not have read access to the memory referred to in that request;
3. the destination file already exists;
4. that instruction refers to a start address that exceeds the maximum memory size;
5. that instruction refers to a start address which is not aligned with the memory access alignment constraint;
6. that instruction refers to a length that combined with the start address exceeds the maximum memory size;
7. that instruction refers to a length that is not a multiple of the memory access alignment constraint.

e. For request to dump raw memory data areas to file that is rejected, the raw data memory management subservice shall generate a failed start of execution notification.

f. For each valid request to dump raw memory data areas to file, the raw data memory management subservice shall:
   1. create the file according to the provided file path.

g. For each request to dump raw memory data areas to file that contains only valid instructions, the raw data memory management subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to dump a raw memory data area to file, the raw data memory management subservice shall:
   1. extract the memory data specified by that instruction from the memory;
   2. append the memory data to the destination file.

      NOTE This standard does not specify the formatting of data within the file. For example, data can be written as a raw byte stream, or include headers identifying the origin of the dumped data.

6.6.3.8 Subservice observables

This Standard does not define any observables for the raw data memory management subservice.

6.6.4 Structured data memory management subservice

6.6.4.1 Checksumming

a. Whether the structured data memory management subservice provides checksumming shall be declared when specifying that subservice.

6.6.4.2 Memory accessibility

a. The list of memories managed by the structured data memory management subservice shall be declared when specifying that subservice.

NOTE Refer also to clause 5.4.3.4.

b. Each memory managed by the structured data memory management subservice shall use the base plus offset addressing scheme.

c. For each writeable memory that it manages, whether the structured data memory management subservice has write access to that memory shall be declared when specifying that subservice.

6.6.4.3 Base plus offset

a. For each memory managed by the structured data memory management subservice, the definition of the base in its base plus offset addressing scheme shall be declared when specifying that memory.

NOTE For example:

- if the memory is used to store files, the base can be the unique identifier of the file used by the file management service (see clause 6.23), i.e. the combination of a repository path and a file name;
- if the memory is used by the OBCP service to store loaded OBCPs, the base can be the OBCP identifier.

b. For each memory managed by the structured data memory management subservice, whether that memory uses static base references or dynamic base references shall be declared when specifying that memory.

NOTE 1 The static base references are those declared upon the specification of the subservice, e.g. a list of static configuration tables.

NOTE 2 The dynamic base references are those dynamically created when using the memory, e.g. when uploading a file using the file management service.

c. If a memory managed by the structured data memory management subservice uses static base references, the list of base identifiers shall be declared when specifying that memory, including for each base:

1. its maximum size in a multiple of the memory access alignment constraint.

NOTE The maximum size of dynamic bases is derived from the size of the related memory object.

d. For each memory managed by the structured data memory management subservice that uses dynamic base references, the base identifier type used to access that memory shall be declared when specifying that memory.

NOTE 1 The base identifier type is either "base address" or "memory object identifier".

NOTE 2 The structure and format of the memory object identifiers depend on the memory content type and what the base refers to, refer to requirement 6.6.4.3a.

6.6.4.4 Load object memory data

a. The structured data memory management subservice shall provide the capability to load object memory data.

NOTE 1 The corresponding requests are of message type "TC[6,1] load object memory data".

NOTE 2 A request to load object memory data that contains more than one instruction is also known as scatter load.

b. Each request to load object memory data shall contain:

1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
2. the base identifier;
3. an ordered list of one or more instructions to load an object memory data.

NOTE 1 For item 1, refer to requirement 6.6.4.2a. If the structured data memory management subservice manages only one memory, the instructions apply to that memory.

NOTE 2 For item 2, refer to requirement 6.6.4.3a.

NOTE 3 All the instructions in the request apply to the memory object identified by the base identifier.

c. The execution verification profile of each request to load object memory data shall include the reporting of the completion of execution.

NOTE For the execution verification profile, refer to requirement 5.3.6.2.3f.

d. Each instruction to load an object memory data shall contain:

1. the byte offset within the memory object identified by the base identifier;
2. the data to load;
3. if the structured data memory management subservice provides checksumming, the checksum value for the verification of the data after it has been loaded to the memory.

NOTE For item 3, refer to requirement 6.6.4.1a.

e. The structured data memory management subservice shall reject any request to load object memory data if any of the following conditions occurs:

1. that request refers to a memory identifier that is unknown;
2. the subservice does not have write access to the memory referred to in that request;
3. that request refers to a memory that is write protected;
4. that request refers to a memory object that is write protected;
5. the base identifier in that request refers to a memory object that is unknown;

6. that request contains an instruction that refers to an offset that exceeds the maximum memory size of the memory object identified by the base identifier;
7. loading the data contained in any related instruction exceeds the maximum memory size of the memory object identified by the base identifier;
8. the size of the data contained within any of the related instruction is not a multiple of the memory access alignment constraint.

f. For each request to load object memory data that is rejected, the structured data memory management subservice shall generate a failed start of execution notification.

g. For each request to load object memory data that contains only valid instructions, the structured data memory management subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to load an object memory data, the structured data memory management subservice shall:
1. write the data to memory.

i. If an error occurs during the writing to memory of the data related to an instruction to load an object memory data, the structured data memory management subservice shall:
1. immediately abort the execution of the related request;
2. generate a failed execution notification for that instruction.

NOTE For example, an error can occur when the memory becomes write-protected by hardware during the course of the load operation.

j. If the subservice provides checksumming, then once the data related to an instruction to load an object memory data has been written to the memory, the structured data memory management subservice shall:
1. calculate the checksum of the loaded data;
2. compare it to the checksum value in that instruction;
3. if that checksum comparison fails:
   (a) immediately abort the execution of the related request;
   (b) generate a failed execution notification for that instruction.

k. For each request to load object memory data that is aborted, the structured data memory management subservice shall generate a failed completion of execution verification report that contains the failed execution notification.

6.6.4.5 Dump object memory data

a. The structured data memory management subservice shall provide the capability to dump object memory data.

NOTE The corresponding requests are of message type "TC[6,3] dump object memory data". The responses are data reports of message type "TM[6,4] dumped object memory data report".

b. Each request to dump object memory data shall contain:

1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
2. the base identifier;
3. an ordered list of one or more instructions to dump an object memory data.

NOTE 1 For item 1, refer to requirement 6.6.4.2a. If the structured data memory management subservice manages only one memory, the instructions apply to that memory.

NOTE 2 For item 2, refer to requirement 6.6.4.3a.

NOTE 3 All the instructions in the request apply to the memory object identified by the base identifier.

c. Each instruction to dump an object memory data shall contain:

1. the byte offset within the memory object identified by the base identifier;
2. the octet length of the memory area to dump.

d. The structured data memory management subservice shall reject any request to dump object memory data if any of the following conditions occurs:

1. that request refers to a memory identifier that is unknown;
2. the base identifier in that request refers to a memory object that is unknown;
3. the subservice does not have read access to the memory referred to in that request;
4. that instruction refers to an offset that combined with the length of the memory area to dump exceeds the maximum memory size of the memory object identified by the base identifier;
5. that instruction refers to an offset which is not aligned with the memory access alignment constraint;
6. that instruction refers to a length that is not a multiple of the memory access alignment constraint.

e. For each request to dump object memory data that is rejected, the structured data memory management subservice shall generate a failed start of execution notification.

f. For each request to dump object memory data that contains only valid instructions, the structured data memory management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to dump an object memory data, the structured data memory management subservice shall:

1. extract the memory data specified by that instruction from the memory;
2. if the subservice provides checksumming, calculate the checksum of the extracted memory data;

3. generate a single dumped object memory data notification that includes:
   (a) the byte offset within the memory object identified by the base identifier;
   (b) the dumped data;
   (c) if the subservice provides checksumming, the calculated checksum of that dumped area.

   NOTE For item 3(a), refer to requirement 6.6.4.1a.

h. For each valid request to dump object memory data, the structured data memory management subservice shall generate a single dumped object memory data report that contains:
   1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
   2. the base identifier;
   3. all related dumped object memory data notifications.

   NOTE 1 For item 1, refer to requirement 6.6.4.2a.

   NOTE 2 For item 2, refer to requirement 6.6.4.3a.

6.6.4.6 Check object memory data

a. The structured data memory management subservice capability to check object memory data shall be declared when specifying that subservice.

   NOTE 1 The corresponding requests are of message type "TC[6,7] check object memory data". The responses are data reports of message type "TM[6,8] checked object memory data report".

   NOTE 2 Checking memory and reporting the calculated checksum for ground checksum comparison avoids downlinking on-board memory areas that are suspected to be faulty.

b. Each request to check object memory data shall contain:
   1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
   2. the base identifier;
   3. an ordered list of one or more instructions to check an object memory data.

   NOTE 1 For item 1, refer to requirement 6.6.4.2a. If the structured data memory management subservice manages only one memory, the instructions apply to that memory.

   NOTE 2 For item 2, refer to requirement 6.6.4.3a.

   NOTE 3 All the instructions in the request apply to the memory object identified by the base identifier.

c. Each instruction to check an object memory data shall contain:
   1. the byte offset within the memory object identified by the base identifier;

2. the octet length of the memory area to check.

d. The structured data memory management subservice shall reject any request to check object memory data if any of the following conditions occurs:
   1. the request refers to a memory identifier that is unknown;
   2. the subservice does not have read access to the memory referred to in that request;
   3. the base identifier in that request refers to a memory object that is unknown.
   4. that instruction refers to an offset that combined with the length of the memory area to check exceeds the maximum memory size of the memory object identified by the base identifier;
   5. that instruction refers to an offset which is not aligned with the memory access alignment constraint;
   6. that instruction refers to a length that is not a multiple of the memory access alignment constraint.

e. For each request to check object memory data that is rejected, the structured data memory management subservice shall generate a failed start of execution notification.

f. For each request to check object memory data that contains only valid instructions, the structured data memory management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to check an object memory data, the structured data memory management subservice shall:
   1. calculate the checksum of the memory area specified by that instruction;
   2. generate a single checked object memory data notification that includes:
      (a) the byte offset within that base;
      (b) the octet length of the data that has been checked;
      (c) the calculated checksum of the checked data.

h. For each valid request to check object memory data, the structured data memory management subservice shall generate a single checked object memory data report that contains
   1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
   2. the base identifier;
   3. all related checked object memory data notifications.

   NOTE 1 For item 1, refer to requirement 6.6.4.2a.

   NOTE 2 For item 2, refer to requirement 6.6.4.3a.

6.6.4.7 Check an object memory object

a. The structured data memory management subservice capability to check an object memory object shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[6,17] check an object memory object". The responses are data reports of message type "TM[6,18] checked object memory object report".

NOTE 2 For example, if the memory is used to store files, the base can be the unique identifier of a file and this request can be used to obtain a checksum of the contents of the file.

b. Each request to check an object memory object shall contain exactly one instruction to check an object memory object.

c. Each instruction to check an object memory object shall contain:

1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
2. the base identifier of the memory object to checksum.

NOTE 1 For item 1, refer to requirement 6.6.4.2a.

NOTE 2 For item 2, refer to requirement 6.6.4.3a.

d. The structured data memory management subservice shall reject any request to check an object memory object if any of the following conditions occurs:

1. that request refers to a memory identifier that is unknown;
2. the base identifier in that request refers to a memory object that is unknown;
3. the subservice cannot determine the length of the memory object referred to by the base identifier.

e. For each request to check an object memory object that is rejected, the structured data memory management subservice shall generate a failed start of execution notification.

f. For each valid instruction to check an object memory object, the structured data memory management subservice shall:

1. calculate the checksum of the memory object specified by that instruction;
2. generate a single checked object memory object notification that includes:
   (a) the octet length of the data that has been checked;
   (b) the calculated checksum of the memory object.

g. For each valid request to check an object memory object, the structured data memory management subservice shall generate a single checked object memory object report that includes:

1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
2. the base identifier;
3. the related checked object memory object notification.

NOTE 1 For item 1, refer to requirement 6.6.4.2a.

NOTE 2 For item 2, refer to requirement 6.6.4.3a.

6.6.4.8 Load object memory data areas by reference

a. The structured data memory management subservice capability to load object memory data areas by reference shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[6,21] load object memory data areas by reference".

b. Each request to load object memory data areas by reference shall contain:

1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
2. the base identifier;
3. the file path of the file containing the data to load;
4. an ordered list of one or more instructions to load an object memory data area by reference.

NOTE 1 For item 1, refer to requirement 6.6.4.2a. If the structured data memory management subservice manages only one memory, the instructions apply to that memory.

NOTE 2 All the instructions in the request apply to the same memory and to the same file.

c. The execution verification profile of each request to load object memory data areas by reference shall include the reporting of the completion of execution.

NOTE For the execution verification profile, refer to requirement 5.3.6.2.3f.

d. Each instruction to load an object memory data area by reference shall contain:

1. the byte offset within the memory object identified by the base identifier;
2. the offset in bytes of the data in the source file;
3. the length in bytes of the data to load;
4. if the structured data memory management subservice provides checksumming, the checksum value for the verification of the data after it has been loaded to the memory.

NOTE For item 4, refer to requirement 6.6.4.1a.

e. The structured data memory management subservice shall reject any request to load object memory data areas by reference if any of the following conditions occurs:

1. that request refers to an invalid memory identifier;
2. the subservice does not have write access to the memory referred to in that request;
3. that request refers to a memory that is write protected;
4. that request refers to a memory object that is write protected;
5. the base identifier in that request refers to a memory object that is unknown;

6. that request refers to a file that does not exist;
7. that request refers to a file that is not recognized as a file containing memory data;
8. that request contains an instruction that refers to:
   (a) a byte offset that exceeds the maximum memory size of the memory object identified by the base identifier;
   (b) a byte offset which is not aligned with respect to the memory access alignment constraint;
   (c) a load length which is not a multiple of the memory access alignment constraint;
   (d) an offset that exceeds the source file size;
9. loading the data contained in one of the related instructions exceeds the maximum memory size.

f. For each request to load object memory data areas by reference that is rejected, the structured data memory management subservice shall generate a failed start of execution notification.

g. For each request to load object memory data areas by reference that contains only valid instructions, the structured data memory management subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to load an object memory data area by reference, the structured data memory management subservice shall:
1. read the data from the source file;
2. write the data to memory.

i. If an error occurs during the writing to memory of the data related to an instruction to load an object memory data area by reference, the structured data memory management subservice shall:
1. immediately abort the execution of the related request;
2. generate a failed execution notification for that instruction.

NOTE For example, an error can occur when the memory becomes write-protected by hardware during the course of the load operation.

j. If the subservice provides checksumming, then once the data related to an instruction to load an object memory data area by reference has been written to the memory, the structured data memory management subservice shall:
1. calculate the checksum of the loaded data;
2. compare it to the checksum value in that instruction;
3. if that checksum comparison fails:
   (a) immediately abort the execution of the related request;
   (b) generate a failed execution notification for that instruction.

k. For each request to load object memory data areas by reference that is aborted, the structured data memory management subservice shall generate a failed completion of execution verification report that contains the failed execution notification.

6.6.4.9 Dump object memory data areas to file

a. The structured data memory management subservice capability to dump object memory data areas to file shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[6,22] dump object memory data areas to file".

b. Each request to dump object memory data areas to file shall contain:

1. if the structured data memory management subservice manages more than one memory, the identifier of the memory;
2. the base identifier;
3. the object path of the destination file;
4. an ordered list of one or more instructions to dump an object memory data area to file.

NOTE For item 1, refer to requirement 6.6.4.2a. If the structured data memory management subservice manages only one memory, the instructions apply to that memory.

c. Each instruction to dump an object memory data area to file shall contain:

1. the byte offset within the memory object identified by the base identifier;
2. the octet length of the memory area to dump.

d. The structured data memory management subservice shall reject any request to dump object memory data areas to file if any of the following conditions occurs:

1. that request refers to an invalid memory identifier;
2. the base identifier in that request refers to a memory object that is unknown;
3. the subservice does not have read access to the memory referred to in that request;
4. the destination file already exists;
5. that instruction refers to an offset that, combined with the length of the memory area to dump, exceeds the maximum memory size of the memory object identified by the base identifier;
6. that instruction refers to an offset which is not aligned with the memory access alignment constraint;
7. that instruction refers to a length that is not a multiple of the memory access alignment constraint.

e. For each request to dump object memory data areas to file that is rejected, the structured data memory management subservice shall generate a failed start of execution notification.

f. For each valid request to dump object memory data areas to file, the structured data memory management subservice shall:

1. create the file according to the provided file path.

g. For each request to dump object memory data areas to file that contains only valid instructions, the structured data memory management

subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to dump an object memory data area to file, the structured data memory management subservice shall:

1. extract the memory data specified by that instruction from the memory;
2. append the memory data to the destination file.

NOTE This standard does not specify the formatting of data within the file. For example, data can be written as a raw byte stream, or include headers identifying the origin of the dumped data.

6.6.4.10 Subservice observables

This Standard does not define any observables for the structured data memory management subservice.

6.6.5 Common memory management subservice

6.6.5.1 Abort all memory dumps

a. The common memory management subservice shall provide the capability to abort all memory dumps.

NOTE 1 The corresponding requests are of message type "TC[6,12] abort all memory dumps".

NOTE 2 Abort all memory dumps implies aborting all dumps of the raw data memory management subservice (refer to clauses 6.6.3.4 and 6.6.3.7) and those of the structured data memory management subservice (refer to clauses 6.6.4.5 and 6.6.4.9).

b. Each request to abort all memory dumps shall contain exactly one instruction to abort all memory dumps.

NOTE The instructions to abort all memory dumps contain no argument.

c. For each valid instruction to abort all memory dumps, the common memory management subservice shall:

1. if the service includes a raw data memory management subservice, abort the execution of all dump requests that are under execution by that subservice;
2. if the service includes a structured data memory management subservice, abort the execution of all dump requests that are under execution by that subservice.

6.6.5.2 Subservice observables

a. The following observables shall be defined for the common memory management subservice:

1. a flag signalling that at least one dump is in-progress.

6.6.6 Memory configuration subservice

6.6.6.1 Scrubbing memory

6.6.6.1.1 Capability

a. Whether the memory configuration subservice supports scrubbing memories shall be declared when specifying that subservice.

6.6.6.1.2 Memory accessibility

a. The list of memories for which scrubbing can be initiated by the memory configuration subservice shall be declared when specifying that subservice.

6.6.6.1.3 Status

a. For each memory for which scrubbing can be initiated by the memory configuration subservice, that subservice shall maintain a status indicating whether scrubbing the memory is enabled or disabled.

NOTE 1 This status is named "memory scrubbing status".

NOTE 2 This Standard does not specify the mechanism providing the memory scrubbing functionality. This mechanism is memory dependent and can rely on hardware or software processes. The memory scrubbing status is used to trigger the scrubbing of the related memory.

b. For each memory for which scrubbing can be initiated by the memory management service, the initial value of the memory scrubbing status for that memory shall be declared when specifying the service.

6.6.6.1.4 Enable the scrubbing of a memory

a. The memory configuration subservice shall provide the capability to enable the scrubbing of a memory if that subservice supports scrubbing memories.

NOTE 1 The corresponding requests are of message type "TC[6,13] enable the scrubbing of a memory".

NOTE 2 For the support to scrub memories, refer to requirement 6.6.6.1.1a.

NOTE 3 For the capability to disable the scrubbing of a memory, refer to clause 6.6.6.1.5.

b. Each request to enable the scrubbing of a memory shall contain exactly one instruction to enable the scrubbing of a memory.

c. Each instruction to enable the scrubbing of a memory shall contain:

1. if the memory configuration subservice manages more than one memory, the identifier of the memory.

NOTE For item 1, refer to requirement 6.6.6.1.2a.

d. The memory configuration subservice shall reject any request to enable the scrubbing of a memory if:

1. that request contains an instruction that refers to a memory that cannot be scrubbed.

   NOTE For item 1, refer to requirement 6.6.6.1.2a.

e. For each request to enable the scrubbing of a memory that is rejected, the memory configuration subservice shall generate a failed start of execution notification.

f. For each valid instruction to enable the scrubbing of a memory, the memory configuration subservice shall:

   1. set the memory scrubbing status of that memory to "enabled".

6.6.6.1.5 Disable the scrubbing of a memory

a. The memory configuration subservice shall provide the capability to disable the scrubbing of a memory if that subservice supports scrubbing memories.

   NOTE 1 The corresponding requests are of message type "TC[6,14] disable the scrubbing of a memory".

   NOTE 2 For the support to scrub memories, refer to requirement 6.6.6.1.1a.

   NOTE 3 For the capability to enable the scrubbing of a memory, refer to clause 6.6.6.1.4.

b. Each request to disable the scrubbing of a memory shall contain exactly one instruction to disable the scrubbing of a memory.

c. Each instruction to disable the scrubbing of a memory shall contain:

   1. if the memory configuration subservice manages more than one memory, the identifier of the memory.

      NOTE For item 1, refer to requirement 6.6.6.1.2a.

d. The memory configuration subservice shall reject any request to disable the scrubbing of a memory if:

   1. that request contains an instruction that refers to a memory that cannot be scrubbed.

      NOTE For item 1, refer to requirement 6.6.6.1.2a.

e. For each request to disable the scrubbing of a memory that is rejected, the memory configuration subservice shall generate a failed start of execution notification.

f. For each valid instruction to disable the scrubbing of a memory, the memory configuration subservice shall:

   1. set the memory scrubbing status of that memory to "disabled".

6.6.6.2 Write protecting memory

6.6.6.2.1 Capability

a. Whether the memory configuration subservice supports write protecting memories shall be declared when specifying that subservice.

6.6.6.2.2 Memory accessibility

a. The list of memories for which write protection can be controlled by the memory configuration subservice shall be declared when specifying that subservice.

6.6.6.2.3 Status

a. For each memory for which the write protection can be controlled by the memory configuration subservice, that subservice shall maintain a status indicating whether the memory is write protected or write unprotected.

NOTE 1 This status is named "memory write protection status".

NOTE 2 The actual implementation of the write protection is memory dependent i.e. write protection by hardware or by software.

6.6.6.2.4 Enable the write protection of a memory

a. The memory configuration subservice shall provide the capability to enable the write protection of a memory if that subservice supports write protecting memories.

NOTE 1 The corresponding requests are of message type "TC[6,15] enable the write protection of a memory".

NOTE 2 For the support to write protecting memories, refer to requirement 6.6.6.2.1a.

NOTE 3 For the capability to disable the write protection of a memory, refer to clause 6.6.6.2.5.

b. Each request to enable the write protection of a memory shall contain exactly one instruction to enable the write protection of a memory.

c. Each instruction to enable the write protection of a memory shall contain:

1. if the memory configuration subservice manages more than one memory, the identifier of the memory.

NOTE For item 1, refer to requirement 6.6.6.2.2a.

d. The memory configuration subservice shall reject any request to enable the write protection of a memory if:

1. that request contains an instruction that refers to a memory that cannot be write protected.

NOTE For item 1, refer to requirement 6.6.6.2.2a.

e. For each request to enable the write protection of a memory that is rejected, the memory configuration subservice shall generate a failed start of execution notification.

f. For each valid instruction to enable the write protection of a memory, the memory configuration subservice shall:

1. set the memory write protection status of that memory to "write protected".

6.6.6.2.5 Disable the write protection of a memory

a. The memory configuration subservice shall provide the capability to disable the write protection of a memory if that subservice supports write protecting memories.

NOTE 1 The corresponding requests are of message type "TC[6,16] disable the write protection of a memory".

NOTE 2 For the support to write protecting memories, refer to requirement 6.6.6.2.1a.

NOTE 3 For the capability to enable the write protection of a memory, refer to clause 6.6.6.2.4.

b. Each request to disable the write protection of a memory shall contain exactly one instruction to disable the write protection of a memory.

c. Each instruction to disable the write protection of a memory shall contain:

1. if the memory configuration subservice manages more than one memory, the identifier of the memory.

NOTE For item 1, refer to requirement 6.6.6.2.2a.

d. The memory configuration subservice shall reject any request to disable the write protection of a memory if:

1. that request contains an instruction that refers to a memory that cannot be write protected.

NOTE For item 1, refer to requirement 6.6.6.2.2a.

e. For each request to disable the write protection of a memory that is rejected, the memory configuration subservice shall generate a failed start of execution notification.

f. For each valid instruction to disable the write protection of a memory, the memory configuration subservice shall:

1. set the memory write protection status of that memory to "write unprotected".

6.6.6.3 Subservice observables

a. The following observables shall be defined for the memory configuration subservice:

1. for each memory for which scrubbing can be controlled by this subservice, its enabled or disabled scrubbing status;
2. for each memory for which write protection can be controlled by this subservice, its write protected or write unprotected status.

6.7 ST[07] (reserved)

6.8 ST[08] (reserved)

NOTE Service type 8 was assigned to the function management service type for backward compatibility reasons. Missions tailoring this Standard are encouraged to develop mission specific service types or mission specific subservice types of standardized services to replace the previous function management service type.

6.9 ST[09] time management

6.9.1 Scope

6.9.1.1 General

The time management service type provides capabilities related to the generation of time reports, the control of their generation rate, the modification of the spacecraft reference time or the reference time of other application processes and the control of the time distribution to on-board users.

All spacecraft maintain a spacecraft time reference, which can be downlinked in time reports. The ground segment can perform a correlation between the reported spacecraft time and the UTC (coordinated universal time) used by the ground segment. This correlation enables the ground system to reconstitute accurately the on-board time of other information reported by the spacecraft, such as the time of occurrence of an event.

The time management service type defines four standardized subservice types, i.e.:

- the time reporting subservice type;
- the time reporting control subservice type;
- the time control subservice type;
- the time distribution subservice type.

6.9.1.2 Time reporting subservice

The time reporting subservice type includes the capability to generate time reports. The time contained in a time report uses a standardized time code format and the subservice type includes capabilities for two of these formats. However, a given time reporting subservice can support only one time code format.

The time code formats are:

- CUC (CCSDS unsegmented), which represents consecutive bits of a binary counter. The CUC format is suitable for applications such as spacecraft time measurement.
- CDS (CCSDS day segmented), which is typically used to report on-board time that is synchronized with a ground time reference, e.g. TAI, UTC.

Each of these time formats consists of two fields, the time code preamble field (P-field) and the time specification field (T-field).

6.9.1.3 Time reporting control subservice

The time reporting control subservice type includes the capability to control the rate of generation of the time reports. This subservice type is used when a mission has varying requirements for time correlation accuracy.

6.9.1.4 Time control subservice

The time control subservice type includes the capability to modify the spacecraft reference time or the reference time of other application processes implementing the time control subservice. This subservice type includes the capability to set by commanding a new reference time (absolute time) or a time offset to be applied to the current reference time (relative time).

6.9.1.5 Time distribution subservice

The time distribution subservice type includes the capability to control the time distribution to on-board users. This subservice type includes the capability to request a one-shot or a periodic time distribution. The time distribution subservice only specifies the distribution to on-board users of the absolute time .

6.9.2 Service layout

6.9.2.1 Subservice

6.9.2.1.1 Time reporting subservice

a. Each time management service shall contain exactly one time reporting subservice.

NOTE the time reporting subservice is hosted by the application that provides the spacecraft time reference based on the on-board clock

6.9.2.1.2 Time reporting control subservice

a. Each time management service shall contain at most one time reporting control subservice.

6.9.2.1.3 Time control subservice

a. Each time management service shall contain at least one time control subservice

NOTE 1 as a minimum the application maintaining the spacecraft reference time is a subservice provider

NOTE 2 on-board users are also subservice providers if they receive the on-board distributed time

6.9.2.1.4 Time distribution subservice

a. Each time management service shall contain at most one time distribution subservice

6.9.2.2 Application process

a. The time reporting subservice provider shall be hosted by the on-board application process that is identified by APID 0.

b. Each application process shall host at most one time reporting control subservice provider.

NOTE The time reporting subservice and the time reporting control subservice can be hosted by different application processes.

c. If the time reporting control subservice has the capability to generate reports, that subservice shall not be hosted by the application process that hosts the time reporting subservice.

NOTE All reports generated by the application process that is identified by APID 0 are time reports. These time reports are transported in CCSDS space packets that have no secondary header, resulting in no adequate means to identify reports of any other type.

d. Each application process shall host at most one time control subservice provider.

NOTE The time reporting subservice and the time control subservice can be hosted by different application processes.

e. If the time control subservice has the capability to generate reports, that subservice shall not be hosted by the application process that hosts the time reporting subservice.

NOTE All reports generated by the application process that is identified by APID 0 are time reports. These time reports are transported in CCSDS space packets that have no secondary header, resulting in no adequate means to identify reports of any other type.

f. Each application process shall host at most one time distribution subservice provider.

NOTE The time reporting subservice and the time distribution subservice can be hosted by different application processes.

g. If the time distribution subservice has the capability to generate reports, that subservice shall not be hosted by the application process that hosts the time reporting subservice.

NOTE All reports generated by the application process that is identified by APID 0 are time reports. These time reports are transported in CCSDS space packets that have no secondary header, resulting in no adequate means to identify reports of any other type.

6.9.3 Spacecraft time reference

a. The time management service shall have access to the spacecraft time reference.

NOTE For the spacecraft time reference, refer to clause 5.4.3.2.

b. The default time report generation rate used by the time management service shall be declared when specifying that subservice.

c. The time report generation rates supported by the spacecraft shall be declared when specifying the time management service.

NOTE 1 The possible time report generation rates are 1, 2, 4, 8, 16, 32, 64, 128 or 256.

NOTE 2 The report generation rate is relative to the generation of telemetry transfer frames on virtual channel 0. For example, if the report generation rate is 16, then every 16th transfer frame on virtual channel 0 causes the generation of a time report packet.

d. The spacecraft time reference sampling accuracy shall be declared when specifying the spacecraft architecture.

e. The accuracy of the time difference between the transmission time of a reference frame and the on-board time sampled and reported in the corresponding time report shall be declared when specifying the time management service.

NOTE 3 The spacecraft time reference sampling accuracy contributes to the time correlation accuracy.

NOTE 4 This Standard does not assume the downlinking of the time packet in the same transfer frame as the one that causes its generation.

f. The time management service shall provide the synchronized timing information used to timestamp the reports generated by all services of the mission.

NOTE For time stamping the reports, refer to requirement 5.4.2.1f.

g. The spacecraft time reference along with its spacecraft time reference status shall be accessible on-board by all services.

6.9.4 Time reporting subservice

6.9.4.1 Capability

a. The time reporting subservice shall provide exactly one of the following capabilities:

1. the capability for generating time reports in CUC format specified in clause 6.9.4.2;

2. the capability for generating time reports in CDS format specified in clause 6.9.4.3.

b. Whether the time reporting subservice supports the capability to report the time report generation rate in the time reports shall be declared when specifying that time reporting subservice.

c. Whether the time reporting subservice supports the capability to report the spacecraft time reference status in the time reports, shall be declared when specifying that time reporting subservice.

d. The spacecraft time reference status shall consist of the following fields:

1. the validity flag (bit 0)

2. the auxiliary data (bits 1 to 3)

NOTE 1 validity_flag = 0 means "valid", validity_flag = 1 means "invalid or lacking accuracy"

NOTE 2 auxiliary data definition is mission-specific

6.9.4.2 Time reporting in CUC format

a. The time reporting subservice capability to generate time reports in CUC time format shall be declared when specifying that subservice.

NOTE 1 The corresponding reports are data reports of message type "TM[9,2] CUC time report".

NOTE 2 For that declaration, refer to requirement 6.9.4.1a.

b. Whether the time reporting subservice includes the P-field in the CUC time reports shall be declared when specifying that subservice.

NOTE If the P-field is not explicitly included, the P-field value is considered implicit.

c. If the time reporting subservice does not include the P-field in the CUC time reports, the implicit P-field value shall be declared when specifying that subservice.

d. The time reporting subservice shall use the CUC time code format specified in CCSDS 301.0-B-4 when generating the CUC time reports.

e. When generating a time report in CUC time format, the time reporting subservice shall:

1. generate a CUC time notification containing:

(a) if supported, the time report generation rate, represented by the rate exponential value;

(b) the spacecraft time;

(c) the spacecraft time reference status if the spacecraft supports the capability to report that status;

2. generate a single CUC time report containing the CUC time notification.

NOTE 1 For item 1(a):

- refer to requirements 6.9.4.1b;

- the rate exponential value is a value that is greater than or equal to 0, and less than or equal to 8, see also requirement 6.9.4.2e.

NOTE 2 For item 1(b), refer to clause 6.9.4.4.

NOTE 3 For item 1(c), refer to requirements 6.9.4.1b and 6.9.4.1c.

NOTE 4 The time reporting subservice generates CUC time reports at a time report generation rate that is equal to:

$$2^{\text{rate exponential value}}$$

The time report generation rate is defined in requirement 6.9.3c.

6.9.4.3 Time reporting in CDS format

a. The time reporting subservice capability to generate time reports in CDS time format shall be declared when specifying that subservice.

NOTE 1 The corresponding reports are data reports of message type "TM[9,3] CDS time report".

NOTE 2 For that declaration, refer to requirement 6.9.4.1a.

b. Whether the time reporting subservice includes the P-field in the CDS time reports shall be declared when specifying that subservice.

NOTE If the P-field is not explicitly included, the P-field value is considered implicit.

c. If the time reporting subservice does not include the P-field in the CDS time reports, the implicit P-field value shall be declared when specifying that subservice.

d. The time reporting subservice shall use the CDS time code format specified in CCSDS 301.0-B-4 when generating the CDS time report.

e. When generating a time report in CDS time format, the time reporting subservice shall:

1. generate a CDS time notification containing:
   (a) if supported, the time report generation rate, represented by the rate exponential value;
   (b) the spacecraft time;
   (c) the spacecraft time reference status if the spacecraft supports the capability to report that status;
2. generate a single CDS time report containing the CDS time notification.

NOTE 1 For item 1(a):

- refer to requirement 6.9.4.1b;
- the rate exponential value is a value that is greater than or equal to 0, and less than or equal to 8, see also requirement 6.9.4.3e.

NOTE 2 For item 1(b), refer to clause 6.9.4.4.

NOTE 3 For item 1(c), refer to requirements 6.9.4.1.b and 6.9.4.1c.

NOTE 4 The time reporting subservice generates CDS time reports at a time report generation rate that is equal to:

$$2^{\text{rate exponential value}}$$

The time report generation rate is defined in requirement 6.9.3c.

6.9.4.4 Time report generation process

a. The time reporting subservice shall sample the spacecraft time reference once for each telemetry transfer frame on virtual channel 0 that satisfies the following condition:

1. the virtual channel frame count carried in the header of a telemetry transfer frame *modulo* the time report generation rate equals to 0.

b. When the time report generation rate is changed, the time reporting subservice shall immediately use the new time report generation rate to determine the next transfer frame that triggers a sampling of the spacecraft time reference.

c. When a telemetry transfer frame triggers the time reporting subservice to sample the spacecraft time reference, the subservice shall sample the time reference at the instant of occurrence of the leading edge of the first bit of the attached synchronization marker of the frame.

d. When a telemetry transfer frame triggers the time reporting subservice to sample the spacecraft time reference, the subservice shall generate the resulting time report at the required time report generation rate.

NOTE 1 The time reports are of message report type "TM[9,2] CUC time report" or "TM[9,3] CDS time report" as derived from requirement 6.9.4.1a.

NOTE 2 For the default time report generation rate, refer to requirement 6.9.3b.

NOTE 3 The time report generation rate can also be set by request, refer to clause 6.9.5.1.1.

e. For each generated time report, the time reporting subservice shall set the T-field of that time report to the sampled value of the spacecraft time reference, formatted according to the time code format.

6.9.4.5 Subservice observables

a. The following observables shall be defined for the time reporting subservice:

1. if the time reporting subservice supports the capability to report the spacecraft time reference status, the spacecraft time reference status.

NOTE For the capability to report the spacecraft time reference status, refer to requirement 6.9.4.1b..

6.9.5 Time reporting control subservice

6.9.5.1 Controlling the time reporting rate

6.9.5.1.1 Set the time report generation rate

a. The time reporting control subservice shall provide the capability to set the time report generation rate.

NOTE The corresponding requests are of message type "TC[9,1] set the time report generation rate".

b. Each request to set the time report generation rate shall contain exactly one instruction to set the time report generation rate.

c. Each instruction to set the time report generation rate shall contain:

1. the rate exponential value representation of the time report generation rate.

NOTE The rate exponential value is calculated as follows:

$$\text{time report generation rate} = 2^{\text{rate exponential value}}$$

The time report generation rate is defined in requirement 6.9.3c.

d. The time reporting control subservice shall reject any request to set the time report generation rate if:

1. that request contains an instruction that contains an invalid time report generation rate.

e. For each request to set the time report generation rate that is rejected, the time reporting control subservice shall generate a failed start of execution notification.

f. For each valid instruction to set the time report generation rate, the time reporting control subservice shall:

1. set the time report generation rate used by the time reporting subservice to the new value in that instruction.

6.9.5.2 Subservice observables

a. The following observables shall be defined for the time reporting control subservice:

1. the time report generation rate.

6.9.6 Time control subservice

6.9.6.1 Setting the reference time

6.9.6.1.1 Set the reference time with absolute time

a. The time control subservice shall provide the capability to set the reference time with absolute time.

NOTE The corresponding requests are of message type "TC[9,4] set the reference time with absolute time".

b. Whether the time control subservice supports the spacecraft time reference status in the set the reference time with absolute time request, shall be declared when specifying that time reporting subservice.

c. Each request to set the reference time with absolute time shall contain exactly one instruction to set the reference time with absolute time.

d. Each instruction to set the reference time with absolute time shall contain:

1. the new reference time expressed as absolute time;
2. if the time reference status is supported, the time reference status

e. For each valid instruction to set the reference time, the time control subservice shall:

1. set the reference time to the new absolute value in that instruction.

NOTE if the time reference status is supported, the decision on how to process the time reference status value, in particular for updating the reference time, is specific to the application processing the request

6.9.6.1.2 Set the reference time with relative time

a. The time control subservice shall provide the capability to set the reference time with relative time.

NOTE The corresponding requests are of message type "TC[9,5] set the reference time with relative time".

b. Whether the time control subservice supports the spacecraft time reference status in the set the reference time with relative time request, shall be declared when specifying that time reporting subservice.

c. Each request to set the reference time with relative time shall contain exactly one instruction to set the reference time with relative time.

d. Each instruction to set the reference time with relative time shall contain:

1. a time offset, positive or negative, to add to the current reference time
2. if the time reference status is supported, the time reference status

e. For each valid instruction to set the reference time with relative time, the time control subservice shall:

1. set the reference time to the sum of the current spacecraft reference time and the time offset value in that instruction.

NOTE if the time reference status is supported, the decision on how to process the time reference status value, in particular for updating the reference time, is specific to the application processing the request

6.9.6.2 Subservice observables

This Standard does not define any observables for the time control subservice.

6.9.7 Time distribution subservice

a. The periodic time distribution intervals supported by the subservice shall be declared when specifying the time distribution subservice

NOTE The periodic time distribution interval shall be expressed as number of seconds, with value 0 meaning a one-shot time distribution request

6.9.7.1 Accessibility

6.9.7.1.1 Application process

a. The list of application processes that can be addressed by the time distribution subservice when releasing requests shall be declared when specifying that subservice

NOTE The application process that hosts the time distribution subservice is not an addressable application process

6.9.7.2 Reference time distribution to on-board users

6.9.7.2.1 Start time distribution to on-board users

a. The time distribution subservice shall provide the capability to start the distribution of the reference time to one or more application process, on a single shot or on a periodic basis.

NOTE 1 The corresponding requests are of message type "TC[9,6] start time distribution to on-board users"

NOTE 2 For the capability to stop the time distribution to on-board users, refer to clause 6.9.7.2.2

b. Each request to start the time distribution to on-board users shall contain:

1. the periodic time distribution interval expressed as an integer number of seconds
2. an ordered list of one or more instructions to start the time distribution to an on-board user

c. Each instruction to start the time distribution to an on-board user shall contain:

1. the application process identifier addressed by that instruction

d. The time distribution subservice shall reject any request to start the time distribution to on-board users if:

1. that request contains an instruction that refers to an invalid periodic time distribution interval

2. that request contains an instruction that refers to an application process that is not controlled by that subservice

e. For each request to start the time distribution to on-board users that is rejected, the time distribution subservice shall generate a failed start of execution notification

f. For each request to start the time distribution to on-board users that contains only valid instructions, the time distribution subservice shall execute those instructions in the order of their appearance in that request

g. For each valid instruction to start the time distribution to an on-board user, the time distribution subservice shall:

1. if the periodic time distribution interval parameter is set to 0, start a one-shot time distribution to the application process addressed by that instruction

2. if the periodic time distribution interval parameter is different from 0, start a periodic time distribution to the application process addressed by that instruction with the requested periodicity

NOTE distributed time to on-board users can be transported in a TC[9,4] packet or by any other on-board communication mean

6.9.7.2.2 Stop time distribution to on-board users

a. The time distribution subservice shall provide the capability to stop the distribution of the reference time to one or more application process.

NOTE 1 The corresponding requests are of message type "TC[9,7] stop time distribution to on-board users"

NOTE 2 For the capability to start the time distribution to on-board users, refer to clause 6.9.7.2.1

b. Each request to stop the time distribution to on-board users shall contain:

1. an ordered list of one or more instructions to stop the time distribution to an on-board user

c. Each instruction to stop the time distribution to an on-board user shall contain:

1. the application process identifier addressed by that instruction

d. The time distribution subservice shall reject any request to stop the time distribution to an on-board user if:

1. that instruction refers to an application process that is not controlled by that subservice

e. For each request to stop the time distribution to on-board users that is rejected, the time distribution subservice shall generate a failed start of execution notification

f. For each request to stop the time distribution to on-board users that contains only valid instructions, the time distribution subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to stop the time distribution to an on-board user, the time distribution subservice shall:

1. stop the time distribution to the application process addressed by that instruction

6.9.7.3 Subservice observables

a. The following observables shall be defined for the time distribution subservice:

1. for each application process for which periodic time distribution has been started

(a) the time distribution periodicity

6.10 ST[10] (reserved)

6.11 ST[11] time-based scheduling

6.11.1 Scope

6.11.1.1 General

The time-based scheduling service type provides the capability to command on-board application processes using requests pre-loaded on-board the spacecraft and released at their due time.

The time-based scheduling service type defines a single standardized subservice type, i.e. the time-based scheduling subservice type.

6.11.1.2 Time-based scheduling subservice

The time-based scheduling subservice type includes the capability to maintain an on-board time-based schedule of requests and to ensure the timely release of those requests.

This provides an extension of the ground monitoring and control. As such, the application process that executes a request released by the time-based scheduling subservice directly sends the request verification reports, if any, to the source identified by the source identifier specified in the request. The release of a request by the subservice is not conditional on the successful or unsuccessful execution of earlier requests released by the subservice.

The time-based scheduling subservice type provides the optional concept of sub-schedules. If the time-based scheduling subservice supports sub-schedules, each request in the time-based schedule is associated to a sub-schedule. Each sub-schedule reflects a coherent on-board operation. Once that operation is completed, the sub-schedule has no further reason to exist. Therefore, sub-schedules are automatically created when used and deleted when empty. The time-based scheduling subservice type includes the capability for enabling and disabling the execution of each sub-schedule.

The time-based scheduling subservice type also provides the optional concept of groups. If the time-based scheduling subservice supports groups, each request in the time-based schedule is associated to a group. The time-based scheduling subservice type includes the capability for enabling and disabling the execution of grouped requests, independently of the application processes they are released to and of the sub-schedules they belong to. Groups are typically related to spacecraft entities (e.g. hardware or software). Groups can be created and deleted by request and can exist even if empty. They can be used, for example, to group all requests associated to a specific instrument and disable their release when the conditions for their execution are not fulfilled, while other requests for the same application process are associated to a different group and enabled for release.

The term "scheduled activity" is used in the time-based scheduling service type to refer to each entry of the time-based schedule. A scheduled activity consists of:

- scheduling data, e.g. the identifier of the sub-schedule, the identifier of the group, the release time;
- the request that is scheduled for later release.

Each scheduled activity is identified by the identifier of the request that is scheduled for later release.

The time-based scheduling subservice type includes optional capability to use a delta time value to time-shift the release times of a set of the activities in the time-based schedule.

6.11.2 Service layout

6.11.2.1 Subservice

6.11.2.1.1 Time-based scheduling subservice

a. Each time-based scheduling service shall contain at least one time-based scheduling subservice.

6.11.2.2 Application process

a. Each application process shall host at most one time-based scheduling subservice provider.

6.11.3 Accessibility

6.11.3.1 Application process

a. The list of application processes that can be addressed by the time-based scheduling subservice when releasing requests shall be declared when specifying that subservice.

NOTE 1 This Standard assumes that all requests of addressable application processes can be used by the time-based scheduling subservice. The application process that hosts the time-based scheduling subservice is, by nature, an addressable application process.

NOTE 2 When the time-based scheduling subservice releases a request, the request is processed by the service that is indicated by the service type and hosted by the application process identified within the request.

NOTE 3 Requests released by the time-based scheduling subservice are not generated by that subservice but by the source that initiated the insert activities into schedule request, i.e. the original source.

6.11.3.2 Service

a. Each time-based scheduling subservice shall be associated to exactly one event reporting subservice.

NOTE 1 This event reporting subservice (refer to clause 6.5) is responsible for catching the events raised by the

time-based scheduling subservice and issuing the corresponding event notifications.

NOTE 2 The events that can be raised by the time-based scheduling subservice are identified by the combination of the identifier of the application process that hosts the event reporting subservice and an event definition identifier.

b. The event reporting subservice that is associated to the time-based scheduling subservice shall be declared when specifying that time-based scheduling subservice.

6.11.4 Managing the time-based schedule

6.11.4.1 Capability

a. Whether the time-based scheduling subservice supports the capability for managing sub-schedules shall be declared when specifying that subservice.

NOTE See clause 6.11.5.

b. Whether the time-based scheduling subservice supports the capability for managing groups specified shall be declared when specifying that subservice.

NOTE See clause 6.11.6.

6.11.4.2 General

a. Each scheduled activity definition shall consist of:

1. the request;
2. the release time of that request;
3. if sub-schedules are supported, the identifier of the sub-schedule to which that scheduled activity is associated;
4. if groups are supported, the identifier of the group to which that scheduled activity is associated.

NOTE 1 For item 3, refer to requirement 6.11.4.1a.

NOTE 2 For item 4, refer requirement 6.11.4.1b.

b. Each scheduled activity definition shall be identified by a scheduled activity identifier that corresponds to the identifier of the request contained in that definition.

NOTE For the request identifier, refer to requirement 5.4.11.2.1.c.

c. The maximum number of scheduled activity definitions that the time-based scheduling subservice can insert within the time-based schedule and contemporaneously process at any time shall be declared when specifying that subservice.

NOTE 1 This Standard assumes that the resources allocated to the time-based scheduling subservice are

sufficient to support this maximum number of scheduled activities independently of the size of the requests they contain.

NOTE 2 The subservice does not constrain the number of scheduled activity having the same release time.

d. The time margin that the time-based scheduling subservice uses when inserting activities in the time-based schedule or time-shifting activities shall be declared when specifying that subservice.

NOTE 1 The time margin is present in order to ensure the consistency and operability of the schedule at any time. Inserting activities or time-shifting them can only be performed if the release time of these activities is greater than or equal to the current time plus a time margin.

NOTE 2 The time margin parameter is called the "time-based schedule time margin".

e. The maximum activity release time that the time-based scheduling subservice uses to release a request contained within a scheduled activity definition once the release time specified in that scheduled activity definition has occurred shall be declared when specifying that subservice.

NOTE The number of activities that the subservice can release at a given time depends on the time taken onboard to activate the execution of any activity and the maximum delta time expressed.

f. The time resolution used by the time-based scheduling subservice shall be declared when specifying that subservice.

g. The configuration policy to apply when starting and restarting the time-based scheduling subservice shall be declared when specifying the subservice.

NOTE 1 this includes if a sub-schedule is automatically disabled or not after the last scheduled activity belonging to that sub-schedule is released

NOTE 2 this includes if a group is automatically disabled or not after the last scheduled activity belonging to that group is released

NOTE 3 this covers the ability of loading an alternative content which is necessary in order to address the non-nominal situations under which the initialization takes place, e.g. a safe-mode.

NOTE 4 in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

6.11.4.3 Controlling the time-based schedule execution function

6.11.4.3.1 Status

a. The time-based scheduling subservice shall maintain a status indicating whether the overall time-based schedule execution function is enabled or disabled.

NOTE This status is named "time-based schedule execution function status".

b. When starting the time-based scheduling subservice, whether the time-based schedule execution function status is set to "enabled" or "disabled" shall be declared when specifying that subservice.

6.11.4.3.2 Enable the time-based schedule execution function

a. The time-based scheduling subservice shall provide the capability to enable the time-based schedule execution function.

NOTE 1 The corresponding requests are of message type "TC[11,1] enable the time-based schedule execution function".

NOTE 2 For the capability to disable the time-based schedule execution function, refer to clause 6.11.4.3.3.

b. Each request to enable the time-based schedule execution function shall contain exactly one instruction to enable the time-based schedule execution function.

NOTE The instructions to enable the time-based schedule execution function contain no argument.

c. For each valid instruction to enable the time-based schedule execution function, the time-based scheduling subservice shall:

1. set the time-based schedule execution function status to "enabled".

NOTE Enabling the time-based schedule execution function does not depend on the presence of scheduled activities in the schedule.

6.11.4.3.3 Disable the time-based schedule execution function

a. The time-based scheduling subservice shall provide the capability to disable the time-based schedule execution function.

NOTE 1 The corresponding requests are of message type "TC[11,2] disable the time-based schedule execution function".

NOTE 2 For the capability to enable the time-based schedule execution function, refer to clause 6.11.4.3.2.

b. Each request to disable the time-based schedule execution function shall contain exactly one instruction to disable the time-based schedule execution function.

NOTE The instructions to disable the time-based schedule execution function contain no argument.

c. For each valid instruction to disable the time-based schedule execution function, the time-based scheduling subservice shall:

1. set the time-based schedule execution function status to "disabled".

NOTE Disabling the time-based schedule execution function does not depend on the presence of scheduled activities in the schedule.

6.11.4.4 Reset the time-based schedule

a. The time-based scheduling subservice shall provide the capability to reset the time-based schedule.

NOTE 1 The corresponding requests are of message type "TC[11,3] reset the time-based schedule".

NOTE 2 This request is accepted regardless of the time-based schedule execution function status.

b. Each request to reset the time-based schedule shall contain exactly one instruction to reset the time-based schedule.

NOTE The instructions to reset the time-based schedule contain no argument.

c. For each valid instruction to reset the time-based schedule, the time-based scheduling subservice shall:

1. set the time-based schedule execution function status to "disabled";
2. delete all scheduled activities from the schedule;
3. if sub-schedules are supported, enable all sub-schedules;
4. if groups are supported, enable all groups.

NOTE 1 Refer to 6.11.4.3.a

NOTE 2 For item 3, refer to requirement 6.11.4.1a.

NOTE 3 For item 4, refer to requirement 6.11.4.1b.

6.11.4.5 Insert activities into the time-based schedule

a. The time-based scheduling subservice shall provide the capability to insert activities into the time-based schedule.

NOTE 1 The corresponding requests are of message type "TC[11,4] insert activities into the time-based schedule".

NOTE 2 Each valid instruction to insert an activity into the time-based schedule results in the creation of a new scheduled activity in the time-based schedule.

NOTE 3 If sub-schedules are supported, the new scheduled activity is associated to the specified sub-schedule.

NOTE 4 If groups are supported, the new scheduled activity is associated to the specified group.

b. Each request to insert activities into the time-based schedule shall contain:

1. if sub-schedules are supported, the sub-schedule identifier;
2. an ordered list of one or more instructions to insert an activity into the time-based schedule.

NOTE For item 1, refer to requirement 6.11.4.1a.

c. Each instruction to insert an activity into the time-based schedule shall contain:
   1. if groups are supported, the group identifier associated to the new scheduled activity;
   2. the release time of that new scheduled activity;
   3. the request associated to that new scheduled activity.

   NOTE For item 1, refer to requirement 6.11.4.1b.

d. The list of verification checks that the time-based scheduling subservice shall perform on the requests associated to the new scheduled activities shall be declared when specifying that subservice.

e. The time-based scheduling subservice shall reject any instruction to insert an activity into the time-based schedule if any of the following conditions occurs:
   1. the activity cannot be added since the maximum number of scheduled activities that can be contemporaneously processed is already reached ;
   2. the release time of the activity is earlier than the time obtained by adding the time-based schedule time margin to the current time;
   3. that instruction refers to a group that is unknown;
   4. that instruction refers to a sub-schedule that is unknown;
   5. the request contained in that instruction fails any of the verification checks.

   NOTE 1 For item 1, refer to requirement 6.11.4.2c.

   NOTE 2 For item 2, refer to requirement 6.11.4.2d.

f. For each instruction to insert an activity into the time-based schedule that is rejected, the time-based scheduling subservice shall generate the failed start of execution notification for that instruction.

g. For each request to insert activities into the time-based schedule that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to insert an activity into the time-based schedule, the time-based scheduling subservice shall:
   1. create a new scheduled activity in the schedule;
   2. place the request specified in that instruction into the new scheduled activity;
   3. set the release time of the new scheduled activity to the release time specified in that instruction;
   4. if sub-schedules are supported, associate the new scheduled activity to the sub-schedule specified in that instruction;

5. if groups are supported, associate the new scheduled activity to the group specified in that instruction.

NOTE 1 For item 3, when a new scheduled activity is set to be released at the same release time than some scheduled activities, the "insert into schedule" order corresponds to the release order used by the subservice.

NOTE 2 For item 4, refer to requirement 6.11.4.1a.

NOTE 3 For item 5, refer to requirement 6.11.4.1b.

6.11.4.6 Schedule execution logic

a. The time-based scheduling subservice shall process the scheduled activities in the order of their release times.

b. When several scheduled activities use the same release time, the time-based scheduling subservice shall release these scheduled activities in the order in which they have been inserted in the time-based schedule.

c. The time-based scheduling subservice shall consider that a scheduled activity is disabled if any of the following conditions occurs:

1. the time-based schedule execution function status is "disabled";
2. that scheduled activity is associated to a disabled sub-schedule;
3. that scheduled activity is associated to a disabled group.

d. For each scheduled activity whose release time is reached, the time-based scheduling subservice shall, in sequence:

1. if that scheduled activity is not disabled and that scheduled activity can be released within the period expressed by the maximum activity release time, release the related request;
2. delete that scheduled activity from the schedule;
3. if the scheduled activity could not be released within the period expressed by the maximum activity release time, generate an event report and discard the related request.

NOTE 4 For item 1, the maximum activity release time is specified in requirement 6.11.4.2e.

NOTE 5 Items 2 and 3 ensure that scheduled activities that cannot be released when their release time is reached are deleted from the schedule.

e. The release of any request by the time-based scheduling subservice shall allow the processing of any request received by that subservice.

NOTE For example, a request to report the content of a sub-schedule is accepted on-board even is that sub-schedule is executing.

6.11.5 Managing time-based sub-schedules

6.11.5.1 Time-based sub-schedules

a. The maximum number of sub-schedules that the time-based scheduling subservice can contemporaneously manage shall be declared when specifying that subservice.

b. For each sub-schedule, the time-based scheduling subservice shall maintain a status indicating whether the schedule execution function for that sub-schedule is enabled or disabled.

NOTE This status is named "sub-schedule status".

6.11.5.2 Enabling and disabling time-based sub-schedules

6.11.5.2.1 Enable time-based sub-schedules

a. The time-based scheduling subservice capability to enable time-based sub-schedules shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,20] enable time-based sub-schedules".

NOTE 2 For the capability to disable time-based sub-schedules, refer to clause 6.11.5.2.2.

b. Each request to enable time-based sub-schedules shall contain exactly one of:

1. an ordered list of one or more instructions to enable a time-based sub-schedule;
2. a single instruction to enable all time-based sub-schedules.

NOTE The instructions to enable all time-based sub-schedules contain no argument.

c. Each instruction to enable a time-based sub-schedule shall contain:

1. the identifier of the sub-schedule to enable.

d. The time-based scheduling subservice shall reject any request to enable time-based sub-schedules if:

1. that instruction refers to an unknown sub-schedule.

e. For each request to enable time-based sub-schedules that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to enable time-based sub-schedules that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable a time-based sub-schedule, the time-based scheduling subservice shall:

1. set the status of that sub-schedule to "enabled".

h. For each valid instruction to enable all time-based sub-schedules, the time-based scheduling subservice shall:

1. for each sub-schedule maintained by the subservice, set its status to "enabled".

6.11.5.2.2 Disable time-based sub-schedules

a. The time-based scheduling subservice shall provide the capability to disable time-based sub-schedules if the capability to enable time-based sub-schedule is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,21] disable time-based sub-schedules".

NOTE 2 For the capability to enable time-based sub-schedule, refer to clause 6.11.5.2.1.

b. Each request to disable time-based sub-schedules shall contain exactly one of:

1. an ordered list of one or more instructions to disable a time-based sub-schedule;
2. a single instruction to disable all time-based sub-schedules.

NOTE The instructions to disable all time-based sub-schedules contain no argument.

c. Each instruction to disable a time-based sub-schedule shall contain:

1. the identifier of the sub-schedule to disable.

d. The time-based scheduling subservice shall reject any request to disable time-based sub-schedules if:

1. that instruction refers to an unknown sub-schedule.

e. For each request to disable time-based sub-schedules that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable a time-based sub-schedule, the time-based scheduling subservice shall:

1. set the status of that sub-schedule to "disabled".

h. For each valid instruction to disable all time-based sub-schedules, the time-based scheduling subservice shall:

1. for each sub-schedule maintained by the subservice, set its status to "disabled".

6.11.5.2.3 Report the status of each time-based sub-schedule

a. The time-based scheduling subservice capability to report the status of each time-based sub-schedule shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,18] report the status of each time-based sub-

schedule". The responses are data reports of message type "TM[11,19] time-based sub-schedule status report".

NOTE 2 That capability requires the capability for that subservice to enable time-based sub-schedules (refer to clause 6.11.5.2.1).

b. Each request to report the status of each time-based sub-schedule shall contain exactly one instruction to report the status of each time-based sub-schedule.

NOTE The instructions to report the status of each time-based sub-schedule contain no argument.

c. For each valid instruction to report the status of each time-based sub-schedule, the time-based scheduling subservice shall:

1. generate, for each time-based sub-schedule managed by the time-based scheduling subservice, a single time-based sub-schedule status notification that includes:

(a) its identifier;

(b) its status.

d. For each valid request to report the status of each time-based sub-schedule, the time-based scheduling subservice shall generate a single time-based sub-schedule status report that includes all related time-based sub-schedule status notifications.

e. The capability to generate autonomously a single time-based sub-schedule status report on a change of a sub-schedule status or sub-schedule deletion shall be declared when specifying the time-based scheduling subservice.

6.11.6 Managing time-based scheduling groups

6.11.6.1 Time-based scheduling groups

a. The maximum number of groups that the time-based scheduling subservice can contemporaneously manage shall be declared when specifying that subservice.

b. For each group, the time-based scheduling subservice shall maintain a status indicating whether the schedule execution function for that group is enabled or disabled.

NOTE This status is named "group status".

6.11.6.2 Creating and deleting time-based scheduling groups

6.11.6.2.1 Create time-based scheduling groups

a. The time-based scheduling subservice capability to create time-based scheduling groups shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,22] create time-based scheduling groups".

NOTE 2 For the capability to delete time-based scheduling groups, refer to clause 6.11.6.2.2.

b. Each request to create time-based scheduling groups shall contain an ordered list of one or more instructions to create a time-based scheduling group.

c. Each instruction to create a time-based scheduling group shall contain:

1. the identifier of the group;
2. the group status at creation time.

d. The time-based scheduling subservice shall reject any request to create time-based scheduling groups if any of the following conditions occurs:

1. that instruction refers to an already existing group;
2. the maximum number of groups that can be contemporaneously managed is already reached.

e. For each request to create time-based scheduling groups that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to create time-based scheduling groups that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to create a time-based scheduling group, the time-based scheduling subservice shall:

1. add the group identifier to the list of groups maintained by that sub-service;
2. set the group status to the value specified in the instruction.

6.11.6.2.2 Delete time-based scheduling groups

a. The time-based scheduling subservice shall provide the capability to delete time-based scheduling groups if the capability to create time-based scheduling groups is provided by that subservice.

NOTE 3 The corresponding requests are of message type "TC[11,23] delete time-based scheduling groups".

NOTE 4 For the capability to create time-based scheduling groups, refer to clause 6.11.6.2.1.

b. Each request to delete time-based scheduling groups shall contain exactly one of:

1. an ordered list of one or more instructions to delete a time-based scheduling group;
2. a single instruction to delete all time-based scheduling groups.

NOTE The instructions to delete all time-based scheduling groups contain no argument.

c. Each instruction to delete a time-based scheduling group shall contain:

1. the identifier of the group to delete.

d. The time-based scheduling subservice shall reject any request to delete time-based scheduling groups if any of the following conditions occurs:

1. that instruction refers to a group that does not exist;
2. that request contains an instruction that refers to a group that has associated activities.

NOTE If there are scheduled activities associated to a group, the group cannot be deleted.

e. For each request to delete a time-based scheduling group that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to delete time-based scheduling groups that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a time-based scheduling group, the time-based scheduling subservice shall:

1. delete the group identifier from the list of groups maintained by that subservice.

h. For each valid instruction to delete all time-based scheduling groups, the time-based scheduling subservice shall:

1. for each group that has no associated activity, delete the identifier of that group;
2. for each group that has associated activities, generate a failed execution notification for that group.

6.11.6.3 Enabling and disabling time-based scheduling groups

6.11.6.3.1 Enable time-based scheduling groups

a. The time-based scheduling subservice shall provide the capability to enable time-based scheduling groups if the capability to create time-based scheduling groups is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,24] enable time-based scheduling groups".

NOTE 2 For the capability to disable time-based scheduling groups, refer to clause 6.11.6.3.2.

b. Each request to enable time-based scheduling groups shall contain exactly one of:

1. an ordered list of one or more instructions to enable a time-based scheduling group;
2. a single instruction to enable all time-based scheduling groups.

NOTE The instructions to enable all time-based scheduling groups contain no argument.

c. Each instruction to enable a time-based scheduling group shall contain:

1. the identifier of the group to enable.

d. The time-based scheduling subservice shall reject any request to enable a time-based scheduling groups if:

1. that instruction refers to an unknown group.

e. For each request to enable time-based scheduling groups that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to enable time-based scheduling groups that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable a time-based scheduling group, the time-based scheduling subservice shall:

1. set the status of that group to "enabled".

h. For each valid instruction to enable all time-based scheduling groups, the time-based scheduling subservice shall:

1. for each group maintained by that subservice, set its status to "enabled".

6.11.6.3.2 Disable time-based scheduling groups

a. The time-based scheduling subservice shall provide the capability to disable time-based scheduling groups if the capability to enable time-based scheduling groups is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,25] disable time-based scheduling groups".

NOTE 2 For the capability to enable time-based scheduling groups, refer to clause 6.11.6.3.1.

b. Each request to disable time-based scheduling groups shall contain exactly one of:

1. an ordered list of one or more instructions to disable a time-based scheduling group;
2. a single instruction to disable all time-based scheduling groups.

NOTE The instructions to disable all time-based scheduling groups contain no argument.

c. Each instruction to disable a time-based scheduling group shall contain:

1. the identifier of the group to disable.

d. The time-based scheduling subservice shall reject any request to disable time-based scheduling groups if:

1. that instruction refers to an unknown group.

e. For each request to disable time-based scheduling groups that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to disable time-based scheduling groups that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable a time-based scheduling group, the time-based scheduling subservice shall:

1. set the status of that group to "disabled".

h. For each valid instruction to disable all time-based scheduling groups, the time-based scheduling subservice shall:

1. for each group maintained by that subservice, set its status to "disabled".

6.11.6.3.3 Report the status of each time-based scheduling group

a. The time-based scheduling subservice capability to report the status of each time-based scheduling group shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,26] report the status of each time-based scheduling group". The responses are data reports of message type "TM[11,27] time-based scheduling group status report".

NOTE 2 That capability requires the capability for that subservice to create time-based scheduling groups, refer to clause 6.11.6.2.1.

b. Each request to report the status of each time-based scheduling group shall contain exactly one instruction to report the status of each time-based scheduling group.

NOTE The instructions to report the status of each time-based scheduling group contain no argument.

c. For each valid instruction to report the status of each time-based scheduling group, the time-based scheduling subservice shall:

1. generate, for each group managed by the time-based scheduling subservice, a single time-based scheduling group status notification that includes:

(a) the group identifier;

(b) its status.

d. For each valid request to report the status of each time-based scheduling group, the time-based scheduling subservice shall generate a single time-based scheduling group status report that includes all related time-based scheduling group status notifications.

e. The capability to generate autonomously a single time-based scheduling group status report on a change of a group status or group deletion shall be declared when specifying the time-based scheduling subservice.

6.11.7 Reports of time-based scheduled activities

6.11.7.1 Time-based schedule summary report

a. The time-based scheduling subservice shall provide the capability to generate time-based schedule summary reports if any of the capabilities to summary-report scheduled activities is provided by that subservice.

NOTE 1 The corresponding reports are data reports of message type "TM[11,13] time-based schedule summary report".

NOTE 2 The capabilities to summary-report scheduled activities are:

- the capability to summary-report all time-based scheduled activities (refer to clause 6.11.8.2);
- the capability to summary-report time-based scheduled activities identified by request identifier (refer to clause 6.11.9.4);
- the capability to summary-report the time-based scheduled activities identified by a filter (refer to clause 6.11.10.5).

b. Each time-based schedule summary report shall contain, for each scheduled activity to summary report, a notification consisting of:

1. if sub-schedules are supported, the identifier of the sub-schedule;
2. if groups are supported, the identifier of the group;
3. the release time;
4. the identifier of the related request consisting of:
   (a) its source identifier;
   (b) its application process identifier;
   (c) its sequence count.

NOTE 1 For item 1, refer to requirement 6.11.4.1a.

NOTE 2 For item 2, refer to requirement 6.11.4.1b.

NOTE 3 The time-based scheduled activities to summary report are determined by one of the requests specified in clauses 6.11.8.2, 6.11.9.4 and 6.11.10.5.

c. The notifications contained in a time-based schedule summary report shall be ordered according to the release time of the reported scheduled activities.

6.11.7.2 Time-based schedule detail report

a. The time-based scheduling subservice shall provide the capability to generate time-based schedule detail reports if any of the capabilities to detail-report scheduled activities is provided by that subservice.

NOTE 1 The corresponding reports are data reports of message type "TM[11,10] time-based schedule detail report".

NOTE 2 The capabilities to detail-report scheduled activities are:

- the capability to detail-report all time-based (refer to clause 6.11.8.3);
- the capability to detail-report time-based scheduled activities identified by request identifier (refer to clause 6.11.9.5);
- the capability to detail-report the time-based scheduled activities identified by a filter (refer to clause 6.11.10.6).

b. Each time-based schedule detail report shall contain, for each scheduled activity to detail report, a notification consisting of:
   1. if sub-schedules are supported, the identifier of the sub-schedule;
   2. if groups are supported, the identifier of the group;
   3. the release time;
   4. the request.

      NOTE 1 For item 1, refer to requirement 6.11.4.1a.

      NOTE 2 For item 2, refer to requirement 6.11.4.1b.

      NOTE 3 The time-based scheduled activities to detail report are determined by one of the requests specified in clauses 6.11.8.3, 6.11.9.5 and 6.11.10.6.

      NOTE 4 The time-based schedule summary report in clause 6.11.7.1 includes only the identifier of the request contained in the scheduled activity. The time-based schedule detail report specified here includes the complete request.

c. The notifications contained in a time-based schedule detail report shall be ordered according to the release time of the reported scheduled activities.

6.11.8 Managing all time-based scheduled activities

6.11.8.1 Time-shift all scheduled activities

a. The time-based scheduling subservice capability to time-shift all scheduled activities shall be declared when specifying that subservice.

   NOTE The corresponding requests are of message type "TC[11,15] time-shift all scheduled activities".

b. Each request to time-shift all scheduled activities shall contain exactly one instruction to time-shift all scheduled activities.

c. Each instruction to time-shift all scheduled activities shall contain:
   1. a time offset, positive or negative, to add to the release time of all scheduled activities.

d. The time-based scheduling subservice shall reject any request to time-shift all scheduled activities if:
   1. the time obtained by adding the time offset to the release time of the earliest activity contained within the time-based schedule is earlier than the time obtained by adding the time-based schedule time margin to the current time.

      NOTE If the time offset is sufficient to result in a scheduled activity with a release time in the past or with a release time that is too close to the current time, that instruction is rejected and no activities are time-shifted.

e. For each request to time-shift all scheduled activities that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to time-shift all scheduled activities, the time-based scheduling subservice shall:

1. for each scheduled activity contained within the time-based schedule:

(a) set the release time of that scheduled activity to the sum of the current release time of that activity and the time offset.

6.11.8.2 Summary-report all time-based scheduled activities

a. The time-based scheduling subservice capability to summary-report all time-based scheduled activities shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[11,17] summary-report all time-based scheduled activities". The responses are data reports of message type "TM[11,13] time-based schedule summary report" (refer to clause 6.11.7.1).

b. Each request to summary-report all time-based scheduled activities shall contain exactly one instruction to summary-report all time-based scheduled activities.

NOTE The instructions to summary-report all time-based scheduled activities contain no argument.

c. For each valid instruction to summary-report all time-based scheduled activities, the time-based scheduling subservice shall generate, for each scheduled activity contained within the time-based schedule, a single time-based schedule summary notification.

NOTE The time-based schedule summary notification content is specified in clause 6.11.7.1.

d. For each valid request to summary-report all time-based scheduled activities, the time-based scheduling subservice shall generate a single time-based schedule summary report that includes all related time-based schedule summary notifications.

NOTE The time-based schedule summary report is specified in clause 6.11.7.1.

6.11.8.3 Detail-report all time-based scheduled activities

a. The time-based scheduling subservice capability to detail-report all time-based scheduled activities shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[11,16] detail-report all time-based scheduled activities". The responses are data reports of message type "TM[11,10] time-based schedule detail report"(refer to clause 6.11.7.2).

b. Each request to detail-report all time-based scheduled activities shall contain exactly one instruction to detail-report all time-based scheduled activities.

NOTE The instructions to detail-report all time-based scheduled activities contain no argument.

c. For each valid instruction to detail-report all time-based scheduled activities, the time-based scheduling subservice shall generate, for each scheduled activity contained within the time-based schedule, a single time-based schedule detail notification.

NOTE The time-based schedule detail notification content is specified in clause 6.11.7.2.

d. For each valid request to detail-report all time-based scheduled activities, the time-based scheduling subservice shall generate a single time-based schedule detail report that includes all related time-based schedule detail notifications.

NOTE The time-based schedule detail report is specified in clause 6.11.7.2.

6.11.9 Managing time-based scheduled activities identified by request identifier

6.11.9.1 General

a. Whether the time-based scheduling subservice supports the identification of scheduled activities by request identifier shall be declared when specifying that subservice.

NOTE That support is required for the capabilities to manage scheduled activities identified by request identifier, i.e.:

- the capability to delete time-based scheduled activities identified by request identifier (refer to clause 6.11.9.2);
- the capability to time-shift scheduled activities identified by request identifier (refer to clause 6.11.9.3);
- the capability to summary-report time-based scheduled activities identified by request identifier (refer to clause 6.11.9.4);
- the capability to detail-report time-based scheduled activities identified by request identifier (refer to clause 6.11.9.5).

6.11.9.2 Delete time-based scheduled activities identified by request identifier

a. The time-based scheduling subservice capability to delete time-based scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[11,5] delete time-based scheduled activities identified by request identifier".

NOTE That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to requirement 6.11.9.1a).

b. Each request to delete time-based scheduled activities identified by request identifier shall contain an ordered list of one or more instructions to delete a time-based scheduled activity identified by request identifier.

c. Each instruction to delete a time-based scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to delete.

NOTE See requirement 6.11.4.2b.

d. The time-based scheduling subservice shall reject any request to delete time-based scheduled activities identified by request identifier if:

1. that instruction contains a request identifier is unknown.

e. For each request to delete time-based scheduled activities identified by request identifier that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to delete time-based scheduled activities identified by request identifier that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a time-based scheduled activity identified by request identifier, the time-based scheduling subservice shall:

1. delete the scheduled activity corresponding to the request identifier;

6.11.9.3 Time-shift scheduled activities identified by request identifier

a. The time-based scheduling subservice capability to time-shift scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,7] time-shift scheduled activities identified by request identifier".

NOTE 2 That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to requirement 6.11.9.1a).

b. Each request to time-shift scheduled activities identified by request identifier shall contain:

1. a time offset, positive or negative, to add to the release time of the specified scheduled activities;

2. an ordered list of one or more instructions to time-shift a scheduled activity identified by request identifier.

NOTE The time offset in a request to time-shift scheduled activities identified by request identifier applies to all the instructions in that request.

c. Each instruction to time-shift a scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to time-shift.

NOTE See requirement 6.11.4.2b.

d. The time-based scheduling subservice shall reject any request to time-shift scheduled activities identified by request identifier if any of the following conditions occurs:

1. the time obtained by adding the time offset to the release time of the earliest activity identified by an instruction in the request is earlier than the time obtained by adding the time-based schedule time margin to the current time;

2. that request identifier is unknown.

NOTE If the time offset is sufficient to result in a scheduled activity with a release time in the past or with a release time that is too close to the current time, that request is rejected and no activities are time-shifted.

e. For each request to time-shift scheduled activities identified by request identifier that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to time-shift scheduled activities identified by request identifier that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to time-shift a scheduled activity identified by request identifier, the time-based scheduling subservice shall:

1. set the release time of the scheduled activity specified in the instruction to the sum of the current release time of that activity and the time offset.

6.11.9.4 Summary-report time-based scheduled activities identified by request identifier

a. The time-based scheduling subservice capability to summary-report time-based scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,12] summary-report time-based scheduled activities identified by request identifier". The responses are data reports of message type "TM[11,13] time-based schedule summary report" (refer to clause 6.11.7.1).

NOTE 2 That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to 6.11.9.1a).

b. Each request to Summary-report time-based scheduled activities identified by request identifier shall contain an ordered list of one or more instructions to summary-report a time-based scheduled activity identified by request identifier.

c. Each instruction to summary-report a time-based scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to report.

NOTE See requirement 6.11.4.2b.

d. The time-based scheduling subservice shall reject any request to summary-report time-based scheduled activities identified by request identifier if:

1. that request identifier is unknown.

e. For each request to summary-report time-based scheduled activities identified by request identifier that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to summary-report time-based scheduled activities identified by request identifier that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their apearance in that request.

g. For each valid instruction to summary-report a time-based scheduled activity identified by request identifier, the time-based scheduling subservice shall generate a single time-based schedule summary notification for that scheduled activity.

NOTE The time-based schedule summary notification content is specified in clause 6.11.7.1

h. For each valid request to Summary-report time-based scheduled activities identified by request identifier, the time-based scheduling subservice shall generate a single time-based schedule summary report that contains all related time-based schedule summary notifications.

NOTE The time-based schedule summary report is specified in clause 6.11.7.1.

6.11.9.5 Detail-report time-based scheduled activities identified by request identifier

a. The time-based scheduling subservice capability to detail-report time-based scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,9] detail-report time-based scheduled activities identified by request identifier". The responses are data reports of message type "TM[11,10] time-based schedule detail report"(refer to clause 6.11.7.2).

NOTE 2 That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to 6.11.9.1a).

b. Each request to detail-report time-based scheduled activities identified by request identifier shall contain an ordered list of one or more instructions to detail-report a time-based scheduled activity identified by request identifier.

c. Each instruction to detail-report a time-based scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to report.

NOTE The activity identifier is specified in requirement 6.11.4.2b.

d. The time-based scheduling subservice shall reject any request to detail-report time-based scheduled activities identified by request identifier if:

1. that request identifier is unknown;

e. For each request to detail-report time-based scheduled activities identified by request identifier that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to detail-report time-based scheduled activities identified by request identifier that contains only valid instructions, the time-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to detail-report a time-based scheduled activity identified by request identifier, the time-based scheduling subservice shall generate a single time-based schedule detail notification for that scheduled activity.

NOTE The time-based schedule detail notification content is specified in clause 6.11.7.2.

h. For each valid request to detail-report time-based scheduled activities identified by request identifier, the time-based scheduling subservice shall generate a single time-based schedule detail report that contains all related time-based schedule detail notifications.

NOTE The time-based schedule detail report is specified in clause 6.11.7.2.

6.11.10 Managing the time-based scheduled activities identified by a filter

6.11.10.1 General

a. Whether the time-based scheduling subservice supports selecting scheduled activity using a time-window filtering function shall be declared when specifying that subservice.

NOTE 1 For the time-window filtering function refer to clause 6.11.10.2.

NOTE 2 That support is required for the capabilities to manage time-based scheduled activities identified by a filter, i.e.:

- the capability to delete the time-based scheduled activities identified by a filter (refer to clause 6.11.10.3);
- the capability to time-shift the time-based scheduled activities identified by a filter (refer to clause 6.11.10.4);
- the capability to summary-report the time-based scheduled activities identified by a filter (refer to clause 6.11.10.5);
- the capability to detail-report the time-based scheduled activities identified by a filter (refer to clause 6.11.10.6).

6.11.10.2 Time-window filtering function

6.11.10.2.1 Overview

Each request that uses the time-window filtering function contains a single filter that identifies which scheduled activities are concerned in that request, based on a combination of:

- a time window;
- if sub-schedules are supported, zero or more sub-schedules;
- if groups are supported, zero or more groups.

6.11.10.2.2 Time window filtering

a. The time window filtering function shall support the following filtering mechanisms:
   1. "select all activities",
   2. "select all activities scheduled from time tag to time tag",
   3. "select all activities scheduled from time tag",
   4. "select all activities scheduled up to time tag".

b. The set of scheduled activities identified by the "select all activities scheduled from time tag to time tag" filtering mechanism shall be all activities that are scheduled between and including the specified "from time tag" and "to time tag".

c. The set of scheduled activities identified by the "select all activities scheduled from time tag" filtering mechanism shall be all activities that are scheduled at and after that specified "from time tag".

d. The set of scheduled activities identified by the "select all activities scheduled up to time tag" filtering mechanism shall be all activities that are scheduled before and at that specified "to time tag".

6.11.10.2.3 Sub-schedule filtering

a. The set of scheduled activities identified by the sub-schedule filtering function shall be all activities that are associated to that sub-schedule.

b. The sub-schedule filtering function shall ignore any unknown sub-schedule that appears in a filter.

6.11.10.2.4 Group filtering

a. The set of scheduled activities identified by the group filtering function shall be all activities that are associated to that group.

6.11.10.2.5 Overall filtering

a. If the overall filtering only includes the time window filtering, the set of scheduled activities identified by the overall filtering function is the set of scheduled activities identified by the time window filtering function.

b. If the overall filtering includes both the time window filtering and the sub-schedule filtering, the set of scheduled activities identified by the overall filtering function is the scheduled activities that result from the intersection of the sets of scheduled activities:
   1. identified by the time window filtering function;
   2. identified by the sub-schedule filtering function.

      NOTE The set of scheduled activities identified by the sub-schedule filtering function consists of the sum of all activities that are associated to the specified sub-schedules. Unknown sub-schedules are ignored.

c. If the overall filtering includes both the time window filtering and the group filtering, the set of scheduled activities identified by the overall filtering function is the scheduled activities that result from the intersection of the sets of scheduled activities:
   1. identified by the time window filtering function;
   2. identified by the group filtering function.

      NOTE The set of scheduled activities identified by the group filtering function consists of the sum of all activities that are associated to the specified groups.

d. If the overall filtering includes the time window filtering, the sub-schedule filtering and the group filtering, the set of scheduled activities identified by the overall filtering function is the scheduled activities that result from the intersection of the sets of scheduled activities:
   1. identified by the time window filtering function;
   2. identified by the sub-schedule filtering function;
   3. identified by the group filtering function.

6.11.10.3 Delete the time-based scheduled activities identified by a filter

a. The time-based scheduling subservice capability to delete the time-based scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,6] delete the time-based scheduled activities identified by a filter".

NOTE 2 That capability implies that the subservice provides the capability of the time-window filtering function (refer to requirement 6.11.10.1a).

b. Each request to delete the time-based scheduled activities identified by a filter shall contain exactly one instruction to delete the time-based scheduled activities identified by a filter.

c. Each instruction to delete the time-based scheduled activities identified by a filter shall contain the filter to identify the scheduled activities to delete consisting of:

1. a time window, consisting of:
   (a) the type of the time window that is one of "select all", "from time tag", "to time tag", "from time tag to time tag";
   (b) for "from time tag" and "from time tag to time tag", the from time tag;
   (c) for "to time tag" and "from time tag to time tag", the to time tag;
2. if sub-schedules are supported, zero or more sub-schedules;
3. if groups are supported, zero or more groups.

NOTE 1 For item 2, refer to requirement 6.11.4.1a.

NOTE 2 For item 3, refer to requirement 6.11.4.1b.

NOTE 3 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.11.10.2.

d. The time-based scheduling subservice shall reject any request to delete the time-based scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid time window type;
2. that request contains an instruction that refers to a "from time tag" that is greater than a "to time tag".

e. For each request to delete the time-based scheduled activities identified by a filter that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to delete the time-based scheduled activities identified by a filter, the time-based scheduling subservice shall:

1. for each scheduled activity identified by that instruction:
   (a) delete that scheduled activity;

6.11.10.4 Time-shift the scheduled activities identified by a filter

a. The time-based scheduling subservice capability to time-shift the scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,8] time-shift the scheduled activities identified by a filter".

NOTE 2 That capability implies that the subservice provides the capability of the time-window filtering function (refer to requirement 6.11.10.1a).

b. Each request to time-shift the scheduled activities identified by a filter shall contain exactly one instruction to time-shift the scheduled activities identified by a filter.

c. Each instruction to time-shift the scheduled activities identified by a filter shall contain:

1. a time offset, positive or negative, to add to the release time of the identified scheduled activities;
2. the time window, consisting of:
   (a) the type of the time window that is one of "select all", "from time tag", "to time tag", "from time tag to time tag";
   (b) for "from time tag" and "from time tag to time tag", the from time tag;
   (c) for "to time tag" and "from time tag to time tag", the to time tag;
3. if sub-schedules are supported, zero or more sub-schedules;
4. if groups are supported, zero or more groups.

NOTE 1 For item 3, refer to requirement 6.11.4.1a.

NOTE 2 For item 4, refer to requirement 6.11.4.1b.

NOTE 3 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.11.10.2.

d. The time-based scheduling subservice shall reject any request to time-shift the scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid time window type;
2. that request contains an instruction that refers to a "from time tag" that is greater than a "to time tag";
3. that request contains an instruction that refers to an unknown sub-schedule;
4. that request contains an instruction refers to an unknown group;
5. the time obtained by adding the time offset to the release time of the earliest activity identified by the filter is earlier than the time obtained by adding the time-based schedule time margin to current time.

NOTE If the time offset is sufficient to result in a scheduled activity with a release time in the past or with a release time that is too close to the current time, no activities are time-shifted.

e. For each request to time-shift the scheduled activities identified by a filter that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to time-shift the scheduled activities identified by a filter, the time-based scheduling subservice shall:

1. for each scheduled activity identified by that instruction:

(a) set the release time of that scheduled activity to the sum of the current release time of that activity and the time offset.

6.11.10.5 Summary-report the time-based scheduled activities identified by a filter

a. The time-based scheduling subservice capability to summary-report the time-based scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,14] summary-report the time-based scheduled activities identified by a filter". The responses are data reports of message type "TM[11,13] time-based schedule summary report" (refer to clause 6.11.7.1).

NOTE 2 That capability implies that the subservice provides the capability of the time-window filtering function (refer to requirement 6.11.10.1a).

b. Each request to summary-report the time-based scheduled activities identified by a filter shall contain exactly one instruction to summary-report the time-based scheduled activities identified by a filter.

c. Each instruction to summary-report the time-based scheduled activities identified by a filter shall contain the filter to identify the scheduled activities to report consisting of:

1. a time window, consisting of:

(a) the type of the time window that is one of "select all", "from time tag", "to time tag", "from time tag to time tag";

(b) for "from time tag" and "from time tag to time tag", the from time tag;

(c) for "to time tag" and "from time tag to time tag", the to time tag;

2. if sub-schedules are supported, zero or more sub-schedules;

3. if groups are supported, zero or more groups.

NOTE 1 For item 2, refer to requirement 6.11.4.1a.

NOTE 2 For item 3, refer to requirement 6.11.4.1b.

NOTE 3 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.11.10.2.

d. The time-based scheduling subservice shall reject any request to summary-report the time-based scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid time window type;
2. that request contains an instruction that refers to a "from time tag" that is greater than a "to time tag".

e. For each request to summary-report the time-based scheduled activities identified by a filter that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to summary-report the time-based scheduled activities identified by a filter, the time-based scheduling subservice shall generate, for each scheduled activity identified by that instruction a single time-based schedule summary notification.

NOTE The time-based schedule summary notification content is specified in clause 6.11.7.1.

g. For each valid request to summary-report the time-based scheduled activities identified by a filter, the time-based scheduling subservice shall generate a single time-based schedule summary report that includes all related time-based schedule summary notifications.

NOTE The time-based schedule summary report is specified in clause 6.11.7.1.

6.11.10.6 Detail-report the time-based scheduled activities identified by a filter

a. The time-based scheduling subservice capability to detail-report the time-based scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[11,11] detail-report the time-based scheduled activities identified by a filter". The responses are data reports of message type "TM[11,10] time-based schedule detail report"(refer to clause 6.11.7.2).

NOTE 2 That capability implies that the subservice provides the capability of the time-window filtering function (refer to requirement 6.11.10.1a).

b. Each request to detail-report the time-based scheduled activities identified by a filter shall contain exactly one instruction to detail-report the time-based scheduled activities identified by a filter.

c. Each instruction to detail-report the time-based scheduled activities identified by a filter shall contain the filter to identify the scheduled activities to report, consisting of:

1. a time window, consisting of:

(a) the type of the time window, that is one of "select all", "from time tag", "to time tag", "from time tag to time tag";

(b) for "from time tag" and "from time tag to time tag", the from time tag;

(c) for "to time tag" and "from time tag to time tag", the to time tag;

2. if sub-schedules are supported, zero or more sub-schedules;

3. if groups are supported, zero or more groups.

NOTE 1 For item 2, refer to requirement 6.11.4.1a.

NOTE 2 For item 3, refer to requirement 6.11.4.1b.

NOTE 3 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.11.10.2.

d. The time-based scheduling subservice shall reject any request to detail-report the time-based scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid time window type;

2. that request contains an instruction that refers to a "from time tag" that is greater than a "to time tag".

e. For each request to detail-report the time-based scheduled activities identified by a filter that is rejected, the time-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to detail-report the time-based scheduled activities identified by a filter, the time-based scheduling subservice shall generate, for each scheduled activity identified by that instruction, a single time-based schedule detail notification.

NOTE The time-based schedule detail notification content is specified in clause 6.11.7.2.

g. For each valid request to detail-report the time-based scheduled activities identified by a filter, the time-based scheduling subservice shall generate a single time-based schedule detail report that includes all related time-based schedule detail notifications.

NOTE The time-based schedule detail report is specified in clause 6.11.7.2.

6.11.11 Subservice observables

a. The following observables shall be defined for the time-based scheduling subservice:

1. the time-based schedule execution function status (enabled or disabled);

2. the current number of scheduled activities in the time-based schedule;

3. if sub-schedules are supported, the current number of sub-schedules;

4. if groups are supported, the current number of groups;
5. the execution time of the first and last schedule entries, independently of their allocation to suc-schedule or group.

6.12 ST[12] on-board monitoring

6.12.1 Scope

6.12.1.1 General

The on-board monitoring service type provides the capability to monitor on-board parameters or groups of parameters and react to the violations of the related monitoring conditions by raising events. The resulting event reports can be sent to ground and caught on-board, e.g. by an event-action subservice.

The on-board monitoring service type defines two standardized subservice types, i.e.:

- the parameter monitoring subservice type;
- the functional monitoring subservice type.

6.12.1.2 Parameter monitoring subservice

The parameter monitoring subservice type provides the capability to monitor on-board parameters with respect to checks defined by the ground system, to report any parameter check transitions to the ground and when monitoring conditions are violated to raise events.

The types of check that can be applied for an on-board parameter depend on the parameter and its type. The subservice type provides the capability to check that a parameter value lies within specified limits or that a parameter has the expected value. It provides optional capability to check that the delta change in a parameter value lies within a pair of threshold values.

For each parameter and associated check, a parameter monitoring definition is specified. This Standard does not introduce any limitation on the number of checks that can be performed on an on-board parameter. A parameter monitoring definition can specify warning limits for an on-board parameter when another definition can specify danger limits for that same parameter. A parameter can be at the same time limit checked and delta checked, using two different parameter monitoring definitions.

The parameter monitoring subservice type provides optional capability to include a conditional check in a parameter monitoring definition. If the conditional check is false, the parameter monitoring check in that definition is not performed. For example, this can be used to disable the monitoring of an on-board parameter when the associated equipment is inactive.

6.12.1.3 Functional monitoring subservice

The functional monitoring subservice type provides the capability to monitor the functional health of on-board elements (e.g. software applications, hardware).

A functional monitoring definition includes a set of one or more parameter monitoring definitions: when a minimum number of these definitions is contemporaneously violated, that functional monitoring definition is considered violated and the associated event is raised.

The behaviour of the functional monitoring subservice type relies on the parameter monitoring subservice type.

6.12.2 Service layout

6.12.2.1 Subservice

6.12.2.1.1 Parameter monitoring subservice

a. Each on-board monitoring service shall contain exactly one parameter monitoring subservice.

6.12.2.1.2 Functional monitoring subservice

a. Each on-board monitoring service shall contain at most one functional monitoring subservice.

6.12.2.2 Application process

a. For each on-board monitoring service that contains both, a parameter monitoring subservice and a functional monitoring subservice, the two subservice providers of that service shall be hosted by the same application process.

6.12.2.3 Accessibility

6.12.2.3.1 Service

a. Each on-board monitoring service shall be associated to exactly one event reporting subservice.

NOTE 1 This event reporting subservice (refer to clause 6.5) is responsible for catching the events raised by the on-board monitoring service and issuing the corresponding event notifications.

NOTE 2 The events that can be raised by the on-board monitoring service are identified by the combination of the identifier of the application process that hosts the event reporting subservice and an event definition identifier.

b. The event reporting subservice that is associated to the on-board monitoring service shall be declared when specifying that on-board monitoring service.

6.12.3 Parameter monitoring subservice

6.12.3.1 Parameter accessibility

a. The parameter monitoring subservice shall be able to monitor all on-board parameters that are accessible to the application process that hosts the subservice.

6.12.3.2 Check types

6.12.3.2.1 Minimum capability

a. The parameter monitoring subservice shall support the evaluation of the following minimum check types:
   1. Limit-check,
   2. Expected-value-check.

b. When performing a limit-check, the parameter monitoring subservice shall:
   1. check that the value of a parameter lies within a pair of limit values;
   2. declare the check successful when the value of the parameter is less than or equal to the high limit value and greater than or equal to the low limit value.

c. When performing an expected-value-check, the parameter monitoring subservice shall:
   1. check that the value resulting from applying a bit mask to a parameter is equal to the expected value;
   2. declare the check successful when these two values are equal.

6.12.3.2.2 Additional capability

a. The parameter monitoring subservice may support the evaluation of the delta-check type.

b. Whether the parameter monitoring subservice supports the delta-check type shall be declared when specifying that subservice.

c. When performing a delta-check, the parameter monitoring subservice shall:
   1. calculate the delta value between two consecutive values of a parameter;
   2. declare the check successful when the delta value is less than or equal to the high threshold value and greater than or equal to the low threshold value.

      NOTE For item 1, the delta value is the difference between the two values.

6.12.3.3 Parameter monitoring definition

a. The maximum number of parameter monitoring definitions that the parameter monitoring subservice can contemporaneously evaluate at any time shall be declared when specifying that subservice.

   NOTE This maximum represents the maximum number of entries in the parameter monitoring definition list. The parameter monitoring definition list is named "PMON list".

b. The parameter monitoring subservice shall provide the capability to process several parameter monitoring definitions for the same on-board parameter.

NOTE For example, with this capability, the monitoring plan can be adapted to specific spacecraft mode conditions using different check validity conditions.

c. Whether the parameter monitoring subservice supports conditional checking of parameter monitoring definitions shall be declared when specifying that subservice.

NOTE This conditional checking depends on a Boolean condition, named "check validity condition". When that Boolean condition is true, the check in the parameter monitoring definition is performed.

d. Whether the parameter monitoring subservice uses a single, subservice-specific monitoring interval for all parameter monitoring definitions or uses a definition-specific monitoring interval for each parameter monitoring definition shall be declared when specifying that subservice.

NOTE The monitoring interval corresponds to the time between two consecutive evaluations of the same parameter monitoring definition.

e. If the parameter monitoring subservice uses a subservice-specific monitoring interval, that monitoring interval shall be declared when specifying that subservice.

f. Monitoring intervals shall be expressed in "on-board parameter minimum sampling interval" units.

NOTE The on-board parameter minimum sampling interval is driven by requirement 5.4.3.2c..

g. Each parameter monitoring definition shall contain:

1. the identifier of the parameter monitoring definition;
2. the identifier of the on-board parameter to monitor;
3. if the parameter monitoring subservice supports the conditional checking of parameter monitoring definitions, a check validity condition that yielding false prevents the check being performed;
4. if the parameter monitoring subservice uses definition-specific monitoring intervals, a monitoring interval;
5. a check definition.

NOTE 1 For item 3, refer to requirements 6.12.3.3c and 6.12.3.3h.

NOTE 2 For item 4, refer to requirement d.

NOTE 3 For item 5, refer to requirement 6.12.3.3j.

h. Each check validity condition shall contain:

1. the identifier of an on-board parameter to use as a validity parameter;
2. a bit-mask;
3. an expected value.

i. When computing the check validity condition, the parameter monitoring subservice shall:

1. perform a bitwise-and between the bit-mask and the sampled value of the validity parameter;
2. declare the condition true when the masked value equals the expected value.

j. Each check definition shall contain:

1. the repetition number that is the number of successive and consistent checks that establishes a new checking status;
2. the check type that is one of:
   (a) limit-check,
   (b) expected-value-check,
   (c) delta-check;
3. for a limit-check:
   (a) the low limit;
   (b) if establishment of a new "below low limit" checking status causes the parameter monitoring subservice to raise an event, the event definition identifier corresponding to that event;
   (c) the high limit;
   (d) if establishment of a new "above high limit" checking status causes the parameter monitoring subservice to raise an event, the event definition identifier corresponding to that event;
4. for an expected-value-check:
   (a) the expected value;
   (b) the mask to apply to the sampled value;
   (c) if establishment of a new "unexpected value" checking status causes the parameter monitoring subservice to raise an event, the event definition identifier corresponding to that event;
5. for a delta-check:
   (a) the number of consecutive delta values, each one calculated between two consecutive values of the parameter, used to calculate the average value of these consecutive delta values that is compared to the low delta threshold value and to the high delta threshold value to determine the PMON checking status;
   (b) the low delta threshold value;
   (c) if establishment of a new "below low threshold" checking status causes the parameter monitoring subservice to raise an event, the event definition identifier corresponding to that event;
   (d) the high delta threshold value;
   (e) if establishment of a new "above high threshold" checking status causes the parameter monitoring subservice to raise an event, the event definition identifier corresponding to that event.

NOTE 1 The types of check that can be applied to parameters depend on their nature, e.g. parameters of analogue

nature can be limit or delta checked, status parameters can be expected value checked.

NOTE 2 the minimum value for the repetition number is 1, meaning that the new checking status is reached on the first consistent check occurrence

6.12.3.4 Statuses

a. The parameter monitoring subservice shall maintain a status indicating whether the overall parameter monitoring function is "enabled" or "disabled".

NOTE This status is named "PMON function status".

b. For each parameter monitoring definition, the parameter monitoring subservice shall maintain a status indicating whether that parameter monitoring definition is "enabled" or "disabled".

NOTE This status is named "PMON status".

c. For each parameter monitoring definition, the parameter monitoring subservice shall maintain a status indicating the established status of the checks performed on the monitored parameter.

NOTE 1 This status is named "PMON checking status".

NOTE 2 For an expected-value-check, the PMON checking status can have any of the following values: "unchecked", "invalid", "expected value" or "unexpected value".

NOTE 3 For a limit-check, the PMON checking status can have any of the following values: "unchecked", "invalid", "within limits", "below low limit" or "above high limit".

NOTE 4 For a delta-check, the PMON checking status can have any of the following values: "unchecked", "invalid", "within threshold", "below low threshold" or "above high threshold".

NOTE 5 The value of the PMON checking status is changed when a number of successive and consistent parameter checks establish a new checking status, see requirement 6.12.3.3j.1. The status values "unchecked" and "invalid" indicate that no checking status is currently established for the parameter.

d. The configuration policy to apply when starting and restarting the parameter monitoring subservice shall be declared when specifying the subservice.

NOTE 1 this includes the overall functional monitoring function status, , "PMON function status"

NOTE 2 this includes the PMON status for each parameter monitoring definition

NOTE 3 this includes the PMON checking status for each parameter monitoring definition

NOTE 4 in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

6.12.3.5 Controlling the parameter monitoring function

6.12.3.5.1 Enable the parameter monitoring function

a. The parameter monitoring subservice shall provide the capability to enable the parameter monitoring function.

NOTE 1 The corresponding requests are of message type "TC[12,15] enable the parameter monitoring function".

NOTE 2 For the capability to disable the parameter monitoring function, refer to clause 6.12.3.5.2.

b. Each request to enable the parameter monitoring function shall contain exactly one instruction to enable the parameter monitoring function.

NOTE The instructions to enable the parameter monitoring function contain no argument.

c. For each valid instruction to enable the parameter monitoring function, the parameter monitoring subservice shall:

1. set the PMON function status to "enabled";
2. for each parameter monitoring definition that is enabled:
   (a) set its PMON checking status to "unchecked";
   (b) reset the repetition counter;
3. start the parameter monitoring process.

NOTE Enabling the parameter monitoring function does not affect the PMON status of the parameter monitoring definitions.

6.12.3.5.2 Disable the parameter monitoring function

a. The parameter monitoring subservice shall provide the capability to disable the parameter monitoring function.

NOTE 1 The corresponding requests are of message type "TC[12,16] disable the parameter monitoring function".

NOTE 2 For the capability to enable the parameter monitoring function, refer to clause 6.12.3.5.1.

b. Each request to disable the parameter monitoring function shall contain exactly one instruction to disable the parameter monitoring function.

NOTE The instructions to disable the parameter monitoring function contain no argument.

c. The parameter monitoring subservice shall reject any request to disable the parameter monitoring function if:

1. the on-board monitoring service includes a functional monitoring subservice whose functional monitoring function is enabled.

NOTE See clause 6.12.4.4.1.

d. For each request to disable the parameter monitoring function that is rejected, the parameter monitoring subservice shall generate a failed start of execution notification.

e. For each valid instruction to disable the parameter monitoring function, the parameter monitoring subservice shall:

1. set the PMON function status to "disabled";
2. stop the parameter monitoring process.

NOTE Disabling the parameter monitoring function affects neither the PMON status nor the PMON checking status of the parameter monitoring definitions.

6.12.3.6 Controlling the parameter monitoring definitions

6.12.3.6.1 Enable parameter monitoring definitions

a. The parameter monitoring subservice shall provide the capability to enable parameter monitoring definitions.

NOTE 1 The corresponding requests are of message type "TC[12,1] enable parameter monitoring definitions".

NOTE 2 For the capability to disable parameter monitoring definitions, refer to clause 6.12.3.6.2.

b. Each request to enable parameter monitoring definitions shall contain an ordered list of one or more instructions to enable a parameter monitoring definition.

c. Each instruction to enable a parameter monitoring definition shall contain:

1. the identifier of the parameter monitoring definition.

d. The parameter monitoring subservice shall reject any request to enable parameter monitoring definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a parameter monitoring definition identifier that is not in the PMON list;
2. that request contains an instruction that refers to a parameter monitoring definition that is used by a protected functional monitoring definition.

NOTE For item 2, the existence of protected functional monitoring definitions depends on the presence of a functional monitoring subservice with support for protecting functional monitoring definitions. See also clause 6.12.4.6.

e. For each request to enable parameter monitoring definitions that is rejected, the parameter monitoring subservice shall generate a failed start of execution notification.

f. For each request to enable parameter monitoring definitions that contains only valid instructions, the parameter monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable a parameter monitoring definition, the parameter monitoring subservice shall:

1. reset the repetition counter of that parameter monitoring definition;
2. set the PMON status of that parameter monitoring definition to "enabled".

NOTE Enabling the PMON status of the parameter monitoring definition does not affect the PMON checking status of that definition.

6.12.3.6.2 Disable parameter monitoring definitions

a. The parameter monitoring subservice shall provide the capability to disable parameter monitoring definitions.

NOTE 1 The corresponding requests are of message type "TC[12,2] disable parameter monitoring definitions".

NOTE 2 For the capability to enable parameter monitoring definitions, refer to clause 6.12.3.6.1.

b. Each request to disable parameter monitoring definitions shall contain an ordered list of one or more instructions to disable a parameter monitoring definition.

c. Each instruction to disable a parameter monitoring definition shall contain:

1. the identifier of the parameter monitoring definition.

d. The parameter monitoring subservice shall reject any request to disable parameter monitoring definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a parameter monitoring definition identifier that is not in the PMON list;
2. that request contains an instruction that refers to a parameter monitoring definition that is used by a protected functional monitoring definition.

NOTE For item 2, the existence of protected functional monitoring definitions depends on the presence of a functional monitoring subservice with support for protecting functional monitoring definitions. See clause 6.12.4.6.

e. For each request to disable parameter monitoring definitions that is rejected, the parameter monitoring subservice shall generate a failed start of execution notification.

f. For each request to disable parameter monitoring definitions that contains only valid instructions, the parameter monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable a parameter monitoring definition, the parameter monitoring subservice shall:

1. set the PMON status of the parameter monitoring definition to "disabled";

2. set the PMON checking status of the parameter monitoring definition to "unchecked".

6.12.3.6.3 Parameter monitoring process

a. If the PMON function status is "disabled", the parameter monitoring subservice shall not perform the parameter monitoring process for any parameter monitoring definitions.

b. If the PMON status of a parameter monitoring definition is disabled, the parameter monitoring subservice shall not perform the parameter monitoring process for that definition.

c. When performing the parameter monitoring process for a parameter monitoring definition, at the end of the monitoring interval, the parameter monitoring subservice shall, in sequence:

1. if the subservice supports the conditional checking of parameter monitoring definitions, compute the check validity condition;

2. if the computed check validity condition yields false:

   (a) set the PMON checking status to "invalid";

   (b) reset the repetition counter of that parameter monitoring definition;

3. if the subservice does not support the conditional checking of parameter monitoring definitions, or if the check validity condition yields true:

   (a) perform the check specified by the check definition, using a newly sampled value of the monitored parameter;

   (b) if the specified "repetition number" of consecutive checks of the monitored parameter have all produced the same checking status output, establish a new PMON checking status.

   NOTE 1 With regards to delta-check, in order to calculate a new delta (the monitored entity) two newly sampled values are needed since the check becomes valid again

   NOTE 2 This Standard does not specify how to react to error cases where the monitored parameter fluctuates between "below low limit" and "above high limit" or between "below low threshold" and "above high threshold".

d. When a new PMON checking status is established, if that status differs from the previous PMON checking status, the parameter monitoring subservice shall:

1. record a check transition by adding that transition to the check transition list;

2. if an event definition is associated to that transition, raise the corresponding event.

   NOTE For item 2, the auxiliary data associated to the raised event includes the same structure and

content that the current out-of-limit notification specified in 6.12.3.11c.

e. When a new PMON checking status is established for an expected-value-check, the parameter monitoring subservice shall set the PMON checking status to:

1. "unexpected value" if the specified "repetition number" of consecutive checks were declared unsuccessful;
2. "expected value", if the specified "repetition number" of consecutive checks were declared successful.

NOTE See requirement 6.12.3.2.1c for the conditions to declare success for an expected-value check.

f. When a new PMON checking status is established for a limit-check, the parameter monitoring subservice shall set the PMON checking status to:

1. "above high limit", if the specified "repetition number" of consecutive checks were declared unsuccessful and the parameter value in each check was greater than the high limit value;
2. "below low limit", if the specified "repetition number" of consecutive checks were declared unsuccessful and the parameter value in each check was less than the low limit value;
3. "within limits", if the specified "repetition number" of consecutive checks were declared successful.

NOTE See requirement 6.12.3.2.1b for the conditions to declare success for a limit check.

g. When a new PMON checking status is established for a delta-check, the parameter monitoring subservice shall set the PMON checking status to:

1. "above high threshold", if the specified "repetition number" of consecutive checks were declared unsuccessful and the delta value in each check was greater than the high threshold value;
2. "below low threshold", if the specified "repetition number" of consecutive checks were declared unsuccessful and the delta value in each check was less than the low threshold value;
3. "within thresholds", if the specified "repetition number" of consecutive checks were declared successful.

NOTE See requirement 6.12.3.2.2c for the conditions to declare success for a delta check.

6.12.3.7 Controlling the check transitions

6.12.3.7.1 Reporting the check transitions

a. The parameter monitoring subservice shall provide the capability to report the contents of the check transition list.

NOTE The corresponding reports are data reports of message type "TM[12,12] check transition report".

b. When reporting the contents of the check transition list, the parameter monitoring subservice shall:

1. for each check transition in the check transition list, generate a check transition notification containing:
   (a) the identifier of the parameter monitoring definition for which the check transition is recorded;
   (b) the identifier of the monitored parameter;
   (c) the check type;
   (d) the parameter value that has caused the transition;
   (e) the limit crossed;
   (f) the PMON checking status before the transition;
   (g) the PMON checking status resulting from the transition;
   (h) the transition time;
2. generate a single check transition report containing all the generated check transition notifications;
3. remove all the reported check transitions from the check transition list.

NOTE 1 For item 1(d) it is the sampled value of the monitored parameter that was used for the last check, in case of expected-value-check, the value is reported after check mask application

NOTE 2 For item 1(e), it is the specified check value of the parameter monitoring definition that was violated.

NOTE 3 For item 1(h), it is the sampling time of the first parameter sample which was used to establish the new checking status.

c. The maximum number of transitions required for issuing a check transition report shall be declared when specifying the parameter monitoring subservice.

d. The parameter monitoring subservice shall report the contents of the check transition list whenever one of the following condition occurs:
   1. the maximum number of transitions required for issuing a check transition report is reached;
   2. at the maximum transition reporting delay after the occurrence of the first check transition recorded in the check transition list.

e. The maximum transition reporting delay shall be expressed in "on-board parameter minimum sampling interval" units.

NOTE The on-board parameter minimum sampling interval is driven by requirement 5.4.3.2.c

f. The default maximum transition reporting delay shall be declared when specifying the parameter monitoring subservice.

NOTE For changing the maximum transition reporting delay, refer to 6.12.3.7.2

6.12.3.7.2 Change the maximum transition reporting delay

a. The parameter monitoring subservice capability to change the maximum transition reporting delay shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[12,3] change the maximum transition reporting delay".

b. Each request to change the maximum transition reporting delay shall contain exactly one instruction to change the maximum transition reporting delay.

c. Each instruction to change the maximum transition reporting delay shall contain:

1. the maximum transition reporting delay.

d. For each valid instruction to change the maximum transition reporting delay, the parameter monitoring subservice shall:

1. set the maximum transition reporting delay to the value specified in that instruction.

6.12.3.7.3 Enable the check transition list logging

a. The parameter monitoring subservice shall provide the capability to enable the check transtions list logging.

NOTE 1 The corresponding requests are of message type "TC[12,29] enable check transition list logging".

NOTE 2 For the capability to disable the check transition list logging, refer to clause 6.12.3.7.4

b. Each request to enable the check transtions list logging shall contain exactly one instruction to enable the check transtions list logging.

NOTE The instructions to enable the check transtions list logging contain no argument.

c. For each valid instruction to enable the check transtions list logging, the parameter monitoring subservice shall:

1. start logging check transitions in the check transtions list.

6.12.3.7.4 Disable the check transition list logging

a. The parameter monitoring subservice shall provide the capability to disable the check transtions list logging.

NOTE 1 The corresponding requests are of message type "TC[12,30] disable check transition list logging".

NOTE 2 For the capability to enable the check transition list logging, refer to clause 6.12.3.7.3

b. Each request to disable the check transtions list logging shall contain exactly one instruction to disable the check transtions list logging.

NOTE The instructions to disable the check transtions list logging contain no argument.

c. For each valid instruction to enable the check transtions list logging, the parameter monitoring subservice shall:

1. stop logging check transitions in the check transtions list;
2. empty the check transtions list.

6.12.3.8 Managing parameter monitoring definitions

6.12.3.8.1 Add parameter monitoring definitions

a. The parameter monitoring subservice capability to add parameter monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,5] add parameter monitoring definitions".

NOTE 2 For the capability to delete all parameter monitoring definitions, refer to clause 6.12.3.8.2.

NOTE 3 For the capability to delete parameter monitoring definitions, refer to clause 6.12.3.8.3.

b. If the capability to add parameter monitoring definitions is provided by the parameter monitoring subservice, that subservice shall provide at least one of the following capabilities:

1. the capability to delete all parameter monitoring definitions specified in clause 6.12.3.8.2;
2. the capability to delete parameter monitoring definitions specified in clause 6.12.3.8.3.

c. Each request to add parameter monitoring definitions shall contain an ordered list of one or more instructions to add a parameter monitoring definition.

d. Each instruction to add a parameter monitoring definition shall contain:

1. the contents of the parameter monitoring definition.

NOTE The contents of a parameter monitoring definition are specified in clause 6.12.3.3g.

e. The parameter monitoring subservice shall reject any request to add parameter monitoring definitions if any of the following conditions occurs:

1. that instruction cannot be added since the PMON list is full;
2. that request contains an instruction that refers to a parameter monitoring definition identifier that is already in the PMON list;
3. that request contains an instruction that refers to a parameter to monitor that is not accessible;
4. that request contains an instruction that refers to a validity parameter that is not accessible;
5. that instruction refers to a limit check for which the high limit is lower than the low limit;
6. that request contains an instruction that refers to a delta check for which the high threshold is lower than the low threshold.

f. For each request to add parameter monitoring definitions that is rejected, the parameter monitoring subservice shall generate a failed start of execution notification.

g. For each request to add parameter monitoring definitions that contains only valid instructions, the parameter monitoring subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to add a parameter monitoring definition, the parameter monitoring subservice shall:

1. add a new parameter monitoring definition to the PMON list, using data from that instruction;
2. set the PMON checking status of the new parameter monitoring definition to "unchecked";
3. set the PMON status of the new parameter monitoring definition to "disabled".

6.12.3.8.2 Delete all parameter monitoring definitions

a. The parameter monitoring subservice capability to delete all parameter monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,4] delete all parameter monitoring definitions".

NOTE 2 For that declaration, refer to requirement 6.12.3.8.1b.

b. Each request to delete all parameter monitoring definitions shall contain exactly one instruction to delete all parameter monitoring definitions.

NOTE The instructions to delete all parameter monitoring definitions contain no argument.

c. The parameter monitoring subservice shall reject any request to delete all parameter monitoring definitions if any of the following conditions occurs:

1. the PMON list contains one or more parameter monitoring definitions that are used by the functional monitoring subservice;
2. the PMON function status is "enabled".

d. For each request to delete all parameter monitoring definitions that is rejected, the parameter monitoring subservice shall generate a failed start of execution notification.

e. For each valid instruction to delete all parameter monitoring definitions, the parameter monitoring subservice shall:

1. delete all entries in the PMON list;
2. delete all entries in the check transition list.

6.12.3.8.3 Delete parameter monitoring definitions

a. The parameter monitoring subservice capability to delete parameter monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,6] delete parameter monitoring definitions".

NOTE 2 For that declaration, refer to requirement 6.12.3.8.1b.

b. Each request to delete parameter monitoring definitions shall contain an ordered list of one or more instructions to delete a parameter monitoring definition.

c. Each instruction to delete a parameter monitoring definition shall contain:

1. the identifier of the parameter monitoring definition.

d. The parameter monitoring subservice shall reject any request to delete parameter monitoring definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a parameter monitoring definition identifier that is not in the PMON list;
2. that request contains an instruction that refers to a parameter monitoring definition whose PMON status is "enabled";
3. that request contains an instruction that refers to a parameter monitoring definition that is used by a functional monitoring definition.

e. For each request to delete parameter monitoring definitions that is rejected, the parameter monitoring subservice shall generate the failed start of execution notification.

f. For each request to delete parameter monitoring definitions that contains only valid instructions, the parameter monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a parameter monitoring definition, the parameter monitoring subservice shall:

1. remove the parameter monitoring definition that is referred to by that instruction from the PMON list.

6.12.3.8.4 Modify parameter monitoring definitions

a. The parameter monitoring subservice capability to modify parameter monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,7] modify parameter monitoring definitions".

NOTE 2 That capability requires the capability for that subservice to add parameter monitoring definitions (refer to clause 6.12.3.8.1).

b. Each request to modify parameter monitoring definitions shall contain an ordered list of one or more instructions to modify a parameter monitoring definition.

c. Each instruction to modify a parameter monitoring definition shall contain:

1. the identifier of the parameter monitoring definition;
2. the identifier of the monitored parameter used by that parameter monitoring definition;
3. the means to modify:

(a) the repetition number;

(b) for a limit-check, its low limit, its high limit and the event definition identifier of each associated event;

(c) for an expected-value-check, its expected-value-check mask, its expected value and the event definition identifier of its associated event;

(d) for a delta-check, its low delta threshold, its high delta threshold and the event definition identifier of each associated event.

NOTE In order to modify the other parameter monitoring definition characteristics, e.g. the check type, this Standard promotes the scenario to delete the parameter monitoring definition and create a new one.

d. The parameter monitoring subservice shall reject any request to modify parameter monitoring definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a parameter monitoring definition identifier that is not in the PMON list;
2. that request contains an instruction that refers to a monitored parameter that is not the one used in that parameter monitoring definition;
3. that request contains an instruction that refers to a limit check for which the high limit is lower than the low limit;
4. that request contains an instruction that refers to a delta check for which the high threshold is lower than the low threshold;
5. that request contains an instruction that refers to a parameter monitoring definition that is used by a protected functional monitoring definition.

NOTE 1 For item 5, the existence of protected functional monitoring definitions depends on the presence of a functional monitoring subservice with support for protecting functional monitoring definitions. See clause 6.12.4.6.

NOTE 2 See clause 8.12.2.7 for additional constraints due to the interface specification.

e. For each request to modify parameter monitoring definitions that is rejected, the parameter monitoring subservice shall generate a failed start of execution notification.

f. For each request to modify parameter monitoring definitions that contains only valid instructions, the parameter monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to modify a parameter monitoring definition, the parameter monitoring subservice shall:

1. modify the parameter monitoring definition that is referred to by that instruction, using data from that instruction;
2. set the PMON checking status of the modified parameter monitoring definition to "unchecked";

3. reset the repetition counter of that parameter monitoring definition.

6.12.3.9 Report parameter monitoring definitions

a. The parameter monitoring subservice capability to report parameter monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,8] report parameter monitoring definitions". The responses are data reports of message type "TM[12,9] parameter monitoring definition report".

NOTE 2 That capability requires the capability for that subservice to provide at least one of:

- the capability to add parameter monitoring definitions (refer to clause 6.12.3.8.1);
- the capability to modify parameter monitoring definitions (refer to clause 6.12.3.8.4).

b. Each request to report parameter monitoring definitions shall contain exactly one of:

1. an ordered list of one or more instructions to report a parameter monitoring definition;
2. a single instruction to report all parameter monitoring definitions.

NOTE The instructions to report all parameter monitoring definitions contain no argument.

c. Each instruction to report a parameter monitoring definition shall contain:

1. the identifier of the parameter monitoring definition.

d. The parameter monitoring subservice shall reject any request to report parameter monitoring definitions if:

1. that request contains an instruction that refers to a parameter monitoring definition identifier that is not in the PMON list.

e. For each request to report parameter monitoring definitions that is rejected, the parameter monitoring subservice shall generate a failed start of execution notification.

f. For each request to report parameter monitoring definitions that contains only valid instructions, the parameter monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report a parameter monitoring definition, the parameter monitoring subservice shall generate a single parameter monitoring definition notification that includes:

1. the parameter monitoring definition that is referred to by that instruction;
2. the PMON status of that parameter monitoring definition.

NOTE The parameter monitoring definition is specified in requirement 6.12.3.3g.

h. For each valid instruction to report all parameter monitoring definitions, the parameter monitoring subservice shall generate, for each parameter

monitoring definition maintained by that subservice, a single parameter monitoring definition notification.

i. For each valid request to report parameter monitoring definitions, the parameter monitoring subservice shall generate a single parameter monitoring definition report that contains:

1. if changing the maximum transition reporting delay is supported, the current value of that delay;
2. all related parameter monitoring definition notifications.

NOTE For item 1, refer to requirement 6.12.3.7.2. For item 2, refer to 6.12.3.3.g.

6.12.3.10 Report the status of each parameter monitoring definition

a. The parameter monitoring subservice capability to report the status of each parameter monitoring definition shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,13] report the status of each parameter monitoring definition". The responses are data reports of message type "TM[12,14] parameter monitoring definition status report".

NOTE 2 That capability requires the capability for that subservice to enable parameter monitoring definitions, refer to clause 6.12.3.6.1.

b. Each request to report the status of each parameter monitoring definition shall contain exactly one instruction to report the status of each parameter monitoring definition.

NOTE The instructions to report the status of each parameter monitoring definition contain no argument.

c. For each valid instruction to report the status of each parameter monitoring definition, the parameter monitoring subservice shall:

1. generate, for each parameter monitoring definition in the PMON list, a single parameter monitoring definition status notification that includes:
   (a) the identifier of the parameter monitoring definition;
   (b) its PMON status;
   (c) its check type;
   (d) its PMON checking status.

d. For each valid request to report the status of each parameter monitoring definition, the parameter monitoring subservice shall generate a single parameter monitoring definition status report that includes all related parameter monitoring definition status notifications.

6.12.3.11 Report the current out-of-limits

a. The parameter monitoring subservice capability to report the current out-of-limits shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[12,10] report the current out-of-limits". The responses are data reports of message type "TM[12,11] current out-of-limits report".

b. Each request to report the current out-of-limits shall contain exactly one instruction to report the current out-of-limits.

NOTE The instructions to report the current out-of-limits contain no argument.

c. For each valid instruction to report the current out-of-limits, the parameter monitoring subservice shall generate:

1. for each parameter monitoring definition that has a faulty PMON checking status, a single current out-of-limit notification that includes:
   (a) the identifier of the parameter monitoring definition;
   (b) the identifier of the monitored parameter;
   (c) the check type;
   (d) the parameter value that has caused the last transition to that faulty PMON checking status;
   (e) the limit crossed;
   (f) the PMON checking status before the last transition to that faulty PMON checking status;
   (g) the PMON checking status resulting from the last transition to that faulty PMON checking status;
   (h) the transition time.

   NOTE 1 For item 1 (g), the faulty PMON checking statuses are:
   - for an expected-value-check: “unexpected value”;
   - for a limit-check: “below low limit”, “above high limit”;
   - for a delta-check: “below low threshold”, “above high threshold”.

   NOTE 2 For item 1(d), it is the sampled value of the monitored parameter that was used for the last check. In case of expected-value-check, the value is reported after check mask application.

   NOTE 3 For item 1(e), it is the specified check value of the parameter monitoring definition that was violated.

   NOTE 4 For item 1(h), it is the sampling time of the first parameter sample that was used to establish the new checking status.

d. For each valid request to report the current out-of-limits, the parameter monitoring subservice shall generate a single current out-of-limits report that includes all related current out-of-limit notifications.

6.12.3.12 Subservice observables

a. The following observables shall be defined for the parameter monitoring subservice:
   1. the number of defined entries in the parameter monitoring definition list;
   2. the number of enabled parameter monitoring definitions;
   3. the PMON function status.

6.12.4 Functional monitoring subservice

6.12.4.1 Accessibility

6.12.4.1.1 Parameter monitoring definition

a. The functional monitoring subservice shall be able to observe, at any time, the PMON checking status of each parameter monitoring definition of the parameter monitoring subservice of the parent on-board monitoring service.

6.12.4.2 Functional monitoring definition

6.12.4.2.1 General

a. The maximum number of functional monitoring definitions that the functional monitoring subservice can contemporaneously evaluate at any time shall be declared when specifying that subservice.

   NOTE This maximum is the maximum number of entries in the functional monitoring definition list. The functional monitoring definition list is named "FMON list".

b. The maximum number of parameter monitoring definitions that a functional monitoring definition can refer to shall be declared when specifying the functional monitoring subservice.

   NOTE This Standard does not limit the number of times a parameter monitoring definition can be called by a functional monitoring definition.

c. Whether the functional monitoring subservice supports conditional checking of functional monitoring definitions shall be declared when specifying that subservice.

d. Whether the functional monitoring subservice supports specifying, for each functional monitoring definition, the minimum number of contemporaneously violated parameter monitoring definitions that establishes a functional monitoring checking failure shall be declared when specifying that subservice.

NOTE 1 This minimum number is named "minimum PMON failing number".

NOTE 2 A minimum PMON failing number that equals 1 means that a functional monitoring definition fails as soon as one of its parameter monitoring definitions fails. This is equivalent to a logical OR of the PMON conditions.

NOTE 3 If a functional monitoring definition has a minimum PMON failing number that is equal to the number of its parameter monitoring definitions, then the functional monitoring definition fails when all its parameter monitoring definitions fail. This is equivalent to a logical AND of the PMON conditions.

e. If the functional monitoring subservice does not support specifying, for each functional monitoring definition, the minimum PMON failing number, the subservice shall use a value of 1 as the minimum PMON failing number for all functional monitoring definitions.

f. Each functional monitoring definition shall contain:

1. its identifier;
2. if the functional monitoring subservice supports the conditional checking of functional monitoring definitions, a check validity condition that yielding false prevents the check being performed;
3. the event definition identifier of the event to raise;
4. if the subservice supports specifying the minimum PMON failing number, a minimum PMON failing number;
5. a set of one or more parameter monitoring definition identifiers.

NOTE 1 For item 2, refer to requirement 6.12.4.2.1c.

NOTE 2 For item 4, refer to requirement 6.12.4.2.1d.

6.12.4.3 Statuses

a. The functional monitoring subservice shall maintain a status indicating whether the overall functional monitoring function is "enabled" or "disabled".

NOTE This status is named "FMON function status".

b. For each functional monitoring definition, the functional monitoring subservice shall maintain a status indicating whether that functional monitoring definition is "enabled" or "disabled".

NOTE This status is named "FMON status".

c. For each functional monitoring definition, the functional monitoring subservice shall maintain a status indicating the result of the check performed.

NOTE 1 This status is named "FMON checking status".

NOTE 2 The FMON checking status can have any of the following values: "unchecked", "invalid", "running" or "failed".

d. If the functional monitoring subservice supports the capability for protecting functional monitoring definitions, the functional monitoring subservice shall maintain, for each functional monitoring definition, a status indicating whether that functional monitoring definition is protected or unprotected.

NOTE 1 For that capability, refer to requirement 6.12.4.6.1a.

NOTE 2 This status is named "FMON protection status".

NOTE 3 When a functional monitoring definition is protected, it cannot be deleted. The parameter monitoring definitions used by a protected functional monitoring definition cannot be enabled, disabled or modified

NOTE 4 If the subservice does not support that capability, all functional monitoring definitions are implicitly unprotected.

e. The configuration policy to apply when starting and restarting the functional monitoring subservice shall be declared when specifying the subservice.

NOTE 1 this includes the overall functional monitoring function status, "FMON function status"

NOTE 2 this includes the FMON status for each functional monitoring definition

NOTE 3 this includes the FMON checking status for each functional monitoring definition

NOTE 4 in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

6.12.4.4 Controlling the functional monitoring function

6.12.4.4.1 Enable the functional monitoring function

a. The functional monitoring subservice shall provide the capability to enable the functional monitoring function.

NOTE 1 The corresponding requests are of message type "TC[12,17] enable the functional monitoring function".

NOTE 2 For the capability to disable the functional monitoring function, refer to clause 6.12.4.4.2.

b. Each request to enable the functional monitoring function shall contain exactly one instruction to enable the functional monitoring function.

NOTE The instructions to enable the functional monitoring function contain no argument.

c. The functional monitoring subservice shall reject any request to enable the functional monitoring function if:

1. the parameter monitoring function of the associated parameter monitoring subservice is disabled.

NOTE See clause 6.12.3.5.1.

d. For each request to enable the functional monitoring function that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

e. For each valid instruction to enable the functional monitoring function, the functional monitoring subservice shall:

1. set the FMON function status to "enabled";
2. for each functional monitoring definition that is enabled:
   (a) set its FMON checking status to "unchecked";
3. start immediately the monitoring of the enabled functional monitoring definitions.

NOTE Enabling the functional monitoring function has no impact on the FMON and FMON protection statuses of the functional monitoring definitions.

6.12.4.4.2 Disable the functional monitoring function

a. The functional monitoring subservice shall provide the capability to disable the functional monitoring function.

NOTE 1 The corresponding requests are of message type "TC[12,18] disable the functional monitoring function".

NOTE 2 For the capability to enable the functional monitoring function, refer to clause 6.12.4.4.1.

b. Each request to disable the functional monitoring function shall contain exactly one instruction to disable the functional monitoring function.

NOTE The instructions to disable the functional monitoring function contain no argument.

c. For each valid instruction to disable the functional monitoring function, the functional monitoring subservice shall:

1. set the FMON function status to "disabled".
2. stop immediately the monitoring of the functional monitoring definitions.

NOTE Disabling the functional monitoring function has no impact on the FMON, FMON protection and FMON checking statuses of the functional monitoring definitions.

6.12.4.5 Controlling the functional monitoring definitions

6.12.4.5.1 Monitoring transitions

a. When the FMON function status is "enabled", for each functional monitoring definition, if the FMON status is "enabled" and the current FMON checking status is not "failed", when the check validity condition is false, the functional monitoring subservice shall set the FMON checking status to “invalid”.

b. When the FMON function status is "enabled", for each functional monitoring definition, if the FMON status is "enabled" and the current

FMON checking status is not "failed", when the check validity condition is true, the functional monitoring subservice shall:

1. check the number of related parameter monitoring definitions that are contemporaneously in violation equals or exceeds the minimum PMON failing number;
2. if that check is true, in sequence:

   (d) set the FMON checking staus to “failed”;

   (e) raise the corresponding event.
3. If that check is false, set the FMON checking status to “running”

NOTE 1 If there is no check validity condition, by nature, it means that the check validity condition is true.

NOTE 2 For item b.2.(b), the auxiliary data associated to the event includes the FMON identifier, the minimum PMON failing number and the list of PMON identifiers that have triggered the FMON.

6.12.4.5.2 Enable functional monitoring definitions

a. The functional monitoring subservice shall provide the capability to enable functional monitoring definitions.

NOTE 1 The corresponding requests are of message type "TC[12,19] enable functional monitoring definitions".

NOTE 2 For the capability to disable functional monitoring definitions, refer to clause 6.12.4.5.3.

b. Each request to enable functional monitoring definitions shall contain an ordered list of one or more instructions to enable a functional monitoring definition.

c. Each instruction to enable a functional monitoring definition shall contain:

1. the identifier of the functional monitoring definition.

d. The functional monitoring subservice shall reject any request to enable functional monitoring definitions if:

1. that request contains an instruction that refers to a functional monitoring definition identifier that is not in the FMON list.

e. For each request to enable functional monitoring definitions that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

f. For each request to enable functional monitoring definitions that contains only valid instructions, the functional monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable a functional monitoring definition, the functional monitoring subservice shall:

1. set the FMON status of the functional monitoring definition to "enabled".

NOTE Enabling the FMON status of the functional monitoring definition does not affect the FMON checking status of that definition.

6.12.4.5.3 Disable functional monitoring definitions

a. The functional monitoring subservice shall provide the capability to disable functional monitoring definitions.

NOTE 1 The corresponding requests are of message type "TC[12,20] disable functional monitoring definitions".

NOTE 2 For the capability to enable functional monitoring definitions, refer to clause 6.12.4.5.2.

b. Each request to disable functional monitoring definitions shall contain an ordered list of one or more instructions to disable a functional monitoring definition.

c. Each instruction to disable a functional monitoring definition shall contain:

1. the identifier of the functional monitoring definition.

d. The functional monitoring subservice shall reject any request to disable a functional monitoring definition if:

1. that request cotains an instruction that refers to a functional monitoring definition identifier that is not in the FMON list.

e. For each request to disable a functional monitoring definition that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

f. For each request to disable functional monitoring definitions that contains only valid instructions, the functional monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable a functional monitoring definition, the functional monitoring subservice shall:

1. set the FMON status of the functional monitoring definition to "disabled";
2. set the FMON checking status of the functional monitoring definition to "unchecked".

6.12.4.6 Protecting functional monitoring definitions

6.12.4.6.1 Protect functional monitoring definitions

a. The functional monitoring subservice capability to protect functional monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,21] protect functional monitoring definitions".

NOTE 2 For the capability to unprotect functional monitoring definitions, refer to clause 6.12.4.6.2.

b. Each request to protect functional monitoring definitions shall contain an ordered list of one or more instructions to protect a functional monitoring definition.

c. Each instruction to protect a functional monitoring definition shall contain:

1. the identifier of the functional monitoring definition.

d. The functional monitoring subservice shall reject any request to protect functional monitoring definitions if:

1. that request contains an instruction that refers to a functional monitoring definition identifier that is not in the FMON list.

e. For each request to protect functional monitoring definitions that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

f. For each request to protect functional monitoring definitions that contains only valid instructions, the functional monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to protect a functional monitoring definition, the functional monitoring subservice shall:

1. set the FMON protection status of the functional monitoring definition to "protected".

NOTE When a functional monitoring definition is protected, it cannot be deleted and it prevents the enabling, disabling or modifying of any parameter monitoring definition that is used in that functional monitoring definition. See clauses 6.12.4.7.2, 6.12.3.6.1, 6.12.3.6.2 and 6.12.3.8.4.

6.12.4.6.2 Unprotect functional monitoring definitions

a. The functional monitoring subservice capability to unprotect functional monitoring definitions shall be provided if the capability to protect functional monitoring definitions is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,22] unprotect functional monitoring definitions".

NOTE 2 For the capability to protect functional monitoring definitions, refer to clause 6.12.4.6.1.

b. Each request to unprotect functional monitoring definitions shall contain an ordered list of one or more instructions to unprotect a functional monitoring definition.

c. Each instruction to unprotect a functional monitoring definition shall contain:

1. the identifier of the functional monitoring definition.

d. The functional monitoring subservice shall reject any request to unprotect a functional monitoring definition if:

1. that request contains an instruction that refers to a functional monitoring definition identifier that is not in the FMON list.

e. For each request to unprotect functional monitoring definitions that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

f. For each request to unprotect functional monitoring definitions that contains only valid instructions, the functional monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to unprotect a functional monitoring definition, the functional monitoring subservice shall:

1. set the FMON protection status of the functional monitoring definition to "unprotected".

6.12.4.7 Modifying functional monitoring definitions

6.12.4.7.1 Add functional monitoring definitions

a. The functional monitoring subservice capability to add functional monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,23] add functional monitoring definitions".

NOTE 2 For the capability to delete functional monitoring definitions, refer to clause 6.12.4.7.2.

b. Each request to add functional monitoring definitions shall contain an ordered list of one or more instructions to add a functional monitoring definition.

c. Each instruction to add a functional monitoring definition shall contain:

1. the contents of the functional monitoring definition.

NOTE The contents of a functional monitoring definition are specified in requirement 6.12.4.2.1f.

d. The functional monitoring subservice shall reject any request to add functional monitoring definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a functional monitoring definition identifier that is already in the FMON list;
2. that request contains more than one instruction for the same functional monitoring definition;
3. that instruction cannot be added since the FMON list is full;
4. that request contains an instruction that refers to a parameter monitoring definition identifier that is not in the PMON list;
5. that request contains an instruction that refers to a validity parameter that is not accessible.

e. For each request to add functional monitoring definitions that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

f. For each request to add a functional monitoring definition that contains only valid instructions, the functional monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to add a functional monitoring definition, the functional monitoring subservice shall:

1. add a new functional monitoring definition to the FMON list, using data from that instruction;
2. set the FMON checking status of the new functional monitoring definition to "unchecked";
3. set the FMON status of the new functional monitoring definition to "disabled";
4. if the functional monitoring subservice supports the capability for protecting functional monitoring definitions, set the FMON protection status of the new functional monitoring definition to "protected".

NOTE For the capability in item 4 refer to requirement 6.12.4.6.1a.

6.12.4.7.2 Delete functional monitoring definitions

a. The functional monitoring subservice shall provide the capability to delete functional monitoring definitions if the capability to add functional monitoring definitions is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,24] delete functional monitoring definitions".

NOTE 2 For the capability to add functional monitoring definitions, refer to clause 6.12.4.7.1.

b. Each request to delete functional monitoring definitions shall contain an ordered list of one or more instructions to delete a functional monitoring definition.

c. Each instruction to delete a functional monitoring definition shall contain:

1. the identifier of the functional monitoring definition.

d. The functional monitoring subservice shall reject any request to delete functional monitoring definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a functional monitoring definition identifier that is not in the FMON list;
2. that request contains an instruction that refers to a functional monitoring definition whose FMON status is "enabled";
3. that request contains an instruction that refers to a functional monitoring definition whose FMON protection status is "protected".

e. For each request to delete functional monitoring definitions that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

f. For each request to delete functional monitoring definitions that contains only valid instructions, the functional monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a functional monitoring definition, the functional monitoring subservice shall:

1. remove the functional monitoring definition that is referred to by that instruction from the FMON list.

6.12.4.8 Report functional monitoring definitions

a. The functional monitoring subservice capability to report functional monitoring definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[12,25] report functional monitoring definitions". The responses are data reports of message type "TM[12,26] functional monitoring definition report".

NOTE 2 That capability requires the capability for that subservice to add functional monitoring definitions. refer to clause 6.12.4.7.1.

b. Each request to report functional monitoring definitions shall contain exactly one of:

1. an ordered list of one or more instructions to report a functional monitoring definition;
2. a single instruction to report all functional monitoring definitions.

NOTE The instructions to report all functional monitoring definitions contain no argument.

c. Each instruction to report a functional monitoring definition shall contain:

1. the identifier of the functional monitoring definition.

d. The functional monitoring subservice shall reject any request to report a functional monitoring definition if:

1. that request contains an instruction that refers to a functional monitoring definition identifier that is not in the FMON list.

e. For each request to report functional monitoring definitions that is rejected, the functional monitoring subservice shall generate a failed start of execution notification.

f. For each request to report functional monitoring definitions that contains only valid instructions, the functional monitoring subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report a functional monitoring definition, the functional monitoring subservice shall generate a single functional monitoring definition notification that includes:

1. the content of the functional monitoring definition that is referred to by that instruction;
2. if the functional monitoring subservice supports the capability for protecting functional monitoring definitions, the FMON protection status of that functional monitoring definition;
3. the FMON status of that functional monitoring definition.

NOTE 1 For item 1, the content of a functional monitoring definition is specified in requirement 6.12.4.2.1f.

NOTE 2 For item 2, refer to requirement 6.12.4.6.1a.

h. For each valid request to report functional monitoring definitions, the functional monitoring subservice shall generate a single functional monitoring definition report that contains all related functional monitoring definition notifications.

6.12.4.9 Report the status of each functional monitoring definition

a. The functional monitoring subservice capability to report the status of each functional monitoring definition shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[12,27] report the status of each functional monitoring definition". The responses are data reports of message type "TM[12,28] functional monitoring definition status report".

b. Each request to report the status of each functional monitoring definition shall contain exactly one instruction to report the status of each functional monitoring definition.

NOTE The instructions to report the status of each functional monitoring definition contain no argument.

c. For each valid instruction to report the status of each functional monitoring definition, the functional monitoring subservice shall:

1. generate, for each functional monitoring definition in the FMON list, a single functional monitoring definition status notification that includes:
   (a) the identifier of that functional monitoring definition;
   (b) if the functional monitoring subservice supports the capability for protecting functional monitoring definitions, its FMON protection status;
   (c) its FMON status;
   (d) its FMON checking status.

   NOTE For item 1(b), refer to requirement 6.12.4.6.1a.

d. For each valid request to report the status of each functional monitoring definition, the functional monitoring subservice shall generate a single functional monitoring definition status report that includes all related functional monitoring definition status notifications.

6.12.4.10 Subservice observables

a. The following observables shall be defined for the functional monitoring subservice:

1. the number of defined entries in the functional monitoring definition list;
2. the number of enabled functional monitoring definitions;
3. the FMON function status.

6.13 ST[13] large packet transfer

6.13.1 Scope

6.13.1.1 General

The ground to space protocol implementations usually limit the maximum length of the CCSDS telemetry and telecommand packets that can be transferred on the downlink and uplink of a spacecraft. These limits are frequently less that the maximum packet size supported by the definition of the CCSDS packet format.

Large requests sent by ground for on-board services can exceed the mission maximum telecommand packet length and large reports generated by the on-board services for the ground can exceed the mission maximum telemetry packet length. These limitations imply that large packets need a specific protocol to manage their transfer.

The large packet transfer service type implements such protocol by splitting a large packet into smaller packets, each containing a part of the large packet. The smaller packets can be transferred between ground and space, and the large packet can be reconstructed from its parts at the receiving end.

The large packet transfer service type defines two standardized subservice types, of similar capabilities and architecture, i.e.:

- the large packet downlink subservice type;
- the large packet uplink subservice type.

As a matter of principle, the services that issue large packets (the sources) and those that receive large packets (the destinations) do not need to know the ground to space constraints and, as such, do not need to know that their large packets are processed by the large packet transfer service.

6.13.1.2 Large packet downlink subservice

The large packet downlink subservice type is composed of:

- a sending entity type which includes the capability, on-board, to:
  - receive and decompose a large packet into an ordered sequence of parts, taking into account the mission maximum telemetry packet length constraint;
  - encapsulate each part within a report, i.e. a "downlink part report";
  - downlink the part reports.
- a receiving entity type which includes the capability, for the ground, to:
  - collect all received part reports for a given large packet;
  - rebuild the large packet from the parts extracted from these part reports;
  - deliver the large packet to its destination.

6.13.1.3 Large packet uplink subservice

The large packet uplink subservice type is composed of:

- a sending entity type which includes the capability, for the ground, to:
  - receive and decompose a large packet into an ordered sequence of parts taking into account the mission maximum telecommand packet length constraint;
  - encapsulate each part within a request, i.e. an "uplink part request";
  - uplink the part requests.
- a receiving entity type which includes the capability, on-board, to:
  - collect all part requests received from the uplink sending entity for a given large packet;
  - rebuild the large packet from the parts extracted from the part requests;
  - deliver the large packet to its destination.

6.13.2 Service layout

6.13.2.1 Subservice

6.13.2.1.1 General

a. Each large packet transfer service shall contain at least one of:
   1. the large packet downlink subservice;
   2. the large packet uplink subservice.

6.13.2.1.2 Large packet downlink subservice

a. Each large packet transfer service shall contain at most one large packet downlink subservice.

6.13.2.1.3 Large packet uplink subservice

a. Each large packet transfer service shall contain at most one large packet uplink subservice.

6.13.2.2 Application process

a. Each large packet transfer subservice provider shall be hosted by exactly one application process.

   NOTE This implies that when both the large packet downlink subservice and the large packet uplink subservice are supported, the sending entity of the downlink subservice and the receiving entity of the uplink subservice are both hosted by that same on-board application process.

b. Each application process shall host at most one large packet transfer subservice provider.

6.13.3 Large packet downlink subservice

6.13.3.1 Configuration

a. The maximum number of large packets that can be downlinked concurrently shall be declared when specifying the large packet downlink subservice.

b. The part size used by the large packet downlink subservice to decompose large packets shall be declared when specifying that subservice.

NOTE This part size is called "downlink maximum part size" and is constrained by the packet field size of a telemetry packet whose length equals the maximum telemetry packet length used to communicate with the spacecraft.

c. The maximum time allocated to the receiving entity for receiving a subsequent downlink part report after the reception of the previous one shall be declared when specifying the large packet downlink subservice.

NOTE This maximum time is called "downlink reception timeout"

6.13.3.2 Resources

a. The resources allocated to the sending entity of the large packet downlink subservice to process large packets shall be declared when specifying the spacecraft architecture and its operations.

6.13.3.3 Downlink process

6.13.3.3.1 Generating downlink part reports

a. The sending entity of the large packet downlink subservice shall have the capability to process each large packet that it receives.

NOTE This Standard assumes that on-board, the large packets are not duplicated. The synchronization between the source of the large packets and the large packet downlink subservice is beyond the scope of this Standard.

b. For each large packet that it processes, the sending entity of the large packet downlink subservice shall:

1. assign a unique large message transaction identifier to that large packet;
2. split the large packet into parts;
3. associate to each part, a unique part sequence number;
4. encapsulate each part into a single "downlink part report".

NOTE 1 The large message transaction identifier is used to uniquely identify the large packet during its overall downlink operation.

NOTE 2 All parts resulting from the decomposition of a large packet have a size that equals the downlink

maximum part size (refer to requirement 6.13.3.1b), except for the last part, which has a size less than or equal to the downlink maximum part size.

NOTE 3 For each large packet, the part sequence number is a counter starting from 1 that specifies, for each part, its position within that large packet. This counter is used by the receiving end when reconstructing the packet, to identify the sequence and position for each part.

c. Each part report shall contain:

1. exactly one part notification made of:

(a) an identifier of whether the part report contains the "First" part, an "Intermediate" part or the "Last" part of the large packet;

(b) the large message transaction identifier;

(c) the part sequence number;

(d) the part itself.

NOTE The corresponding reports are data reports of message type:

- "TM[**13,1**] first downlink part report" for the first part,
- "TM[**13,2**] intermediate downlink part report" for the intermediate parts, and
- "TM[**13,3**] last downlink part report" for the last part.

d. The destination of the part reports generated by the large packet downlink subservice shall be declared when specifying the space to ground architecture.

NOTE The destination referred to in that requirement is the ground application process that hosts the large packet downlink subservice receiving entity.

e. The sending entity of the large packet downlink subservice shall generate the part reports related to each large packet, in increasing order of the part sequence number and at the highest frequency supported under the prevailing operation constraints.

6.13.3.3.2 Accepting part reports and reconstructing large packets

a. The receiving entity of the large packet downlink subservice shall have the capability to process all part reports that it receives.

NOTE This process is called "large packet acceptance and reconstruction process".

b. The receiving entity of the large packet downlink subservice shall initiate the downlink operation when it receives the first part report of the large packet.

c. The receiving entity of the large packet downlink subservice shall initiate the reception timer after the successful reception of a first or intermediate part report.

d. For each part report that is received, the receiving entity of the large packet downlink subservice shall include that part in the reconstruction process of the related large packet.

e. The receiving entity of the large packet downlink subservice shall end the downlink operation when the last part report of the large packet has been successfully received.

f. The receiving entity of the large packet downlink subservice shall abort the downlink operation when the reception timer reaches the downlink reception timeout.

NOTE See requirement 6.13.3.1c.

g. Upon completion of the downlink operation, if all part reports have been successfully received, the receiving entity of the large packet downlink subservice shall:

1. generate that large packet for subsequent routing to its destination.

NOTE The receiving entity is not in charge of checking the checksum of the reconstructed large packet.

h. For each large packet reconstruction that is aborted or that completes without having successfully received all parts, the receiving entity of the large packet downlink subservice shall:

1. notify the ground monitoring and control system of that large packet downlink abortion and the missing parts;
2. discard that large packet and related part reports.

6.13.3.4 Subservice Observables

a. The following observables shall be defined for the on-board large packet downlink subservice:

1. the number of on-going downlinks;
2. the list of large message transaction identifiers associated to the on-going downlinks in an array of size corresponding to the maximum number of large packets that can be downlinked concurrently.

NOTE For item 2, refer to requirements 6.13.3.3.1b.1 and 6.13.3.1a.

6.13.4 Large packet uplink subservice

6.13.4.1 Configuration

a. The maximum number of large packets that can be uplinked concurrently shall be declared when specifying the large packet uplink subservice.

b. The part size used by the large packet uplink subservice to decompose large packets shall be declared when specifying that subservice.

NOTE This part size is called "uplink maximum part size". It corresponds to the packet field size used by the part of a telecommand packet which packet size equals to the maximum telecommand packet length used for communicating with the spacecraft.

c. The maximum time allocated to the uplink receiving entity for receiving a subsequent uplink part request after the reception of the previous one shall be declared when specifying the large packet uplink subservice.

NOTE This maximum time is called "Uplink reception timeout".

6.13.4.2 Resources

a. The resources allocated to the uplink receiving entity of the large packet uplink subservice to process large packets shall be declared when specifying the spacecraft architecture and its operations.

6.13.4.3 Uplink process

6.13.4.3.1 Generating uplink part requests

a. For each large packet that it processes, the sending entity of the large packet uplink subservice shall:

1. assign a unique large message transaction identifier to that large packet;
2. split the large packet into parts;
3. associate to each part, a unique part sequence number;
4. encapsulate each part into a single "uplink part request".

NOTE 1 The large message transaction identifier is used to uniquely identify the large packet during its overall uplink operation.

NOTE 2 All parts resulting from the decomposition of a large packet have a size that is equal to the uplink maximum part size (refer to requirement 6.13.4.1b), except for the last part, which has a size less than or equal to the uplink maximum part size.

NOTE 3 For each large packet, the part sequence number is a counter starting from 1 that specifies, for each part, its position within that large packet. This counter is used by the receiving end when reconstructing the packet, to identify the sequence and position for each part.

b. Each part request shall contain:

1. exactly one part instruction made of:
   (a) an identifier of whether the part request is the "First" part, an "Intermediate" part or the "Last" part of the large packet;
   (b) the large message transaction identifier;
   (c) the part sequence number;

(d) the part itself.

NOTE The corresponding requests are of message type:

- "**TC[13,9**] uplink the first part" for the first part,
- "**TC[13,10**] uplink an intermediate part" for the intermediate parts, and
- "**TC[13,11**] uplink the last part" for the last part.

c. The destination of the uplink part requests generated by the large packet uplink subservice shall be declared when specifying the space to ground architecture.

NOTE The destination referred in that requirement is the on-board application process that hosts the large packet uplink subservice receiving entity.

d. The sending entity of the large packet uplink subservice shall generate the uplink part requests related to each large packet, in increasing order of part sequence number and at the highest frequency supported under the prevailing operation constraints.

6.13.4.3.2 Accepting uplink part requests and reconstructing large packets

a. The receiving entity of the large packet uplink subservice shall be able to process all uplink part requests that it receives.

NOTE This process is called "large packet acceptance and reconstruction process".

b. The receiving entity of the large packet uplink subservice shall initiate the uplink operation when it receives the request to uplink the first part of the large packet.

c. The receiving entity of the large packet uplink subservice shall initiate the reception timer after the successful reception of the request to uplink the first part or the request to uplink an intermediate part.

d. The receiving entity of the large packet uplink subservice shall end the uplink operation when the request to uplink the last part of the large packet has successfully been received.

e. The receiving entity of the large packet uplink subservice shall abort the uplink operation when the reception timer reaches the uplink reception timeout.

NOTE See requirement 6.13.4.1c.

f. The receiving entity of the large packet uplink subservice shall abort the uplink operation when a discontinuity is detected in the uplink reception sequence.

g. For each uplink part request that is received, the receiving entity of the large packet uplink subservice shall include that part in the reconstruction process of the related large packet.

h. Upon successful completion of the uplink operation, the receiving entity of the large packet uplink subservice shall:

1. generate that large packet for subsequent routing to its destination.

NOTE The receiving entity is not in charge of checking the checksum of the reconstructed large packets.

i. For each large packet uplink that is aborted, the receiving entity of the large packet uplink subservice shall:

1. generate a single large packet uplink abortion notification that includes the reason of that abortion;
2. discard that large packet and the related uplink part requests.

6.13.4.3.3 Large packet uplink abortion report

a. The receiving entity of the large packet uplink shall provide the capability to generate large packet uplink abortion reports.

NOTE The corresponding data reports are of message type "TM[13,16] large packet uplink abortion report".

b. For each large packet uplink abortion notification that it generates, the receiving entity of the large packet uplink subservice shall generate a single large packet uplink abortion report that contains that notification.

c. Each large packet uplink abortion notification shall contain:

1. the large message transaction identifier;
2. the abortion reason.

6.13.4.4 Subservice Observables

a. The following observables shall be defined for the large packet uplink subservice:

1. the number of on-going uplinks;
2. the list of the large message transaction identifiers associated to the on-going uplinks in an array of size corresponding to the maximum number of large packets that can be uplinked concurrently.

NOTE For item 2, refer to requirements 6.13.4.3.1a.1 and 6.13.4.1a.

6.14 ST[14] real-time forwarding control

6.14.1 Scope

6.14.1.1 General

The real-time forwarding control service type provides the capability to control the real-time forwarding to the ground of reports (verification reports, responses and data) generated by on-board services. The reports are forwarded to ground within a real-time telemetry channel.

The real-time forwarding control service type defines a single standardized subservice type, i.e. the real-time forwarding control subservice type.

It is noted that the capabilities offered by the real-time forwarding control service type are fully independent of those provided by the on-board storage and retrieval service type specified in clause 6.15.

6.14.1.2 Real-time forwarding control subservice

The real-time forwarding control subservice type can be used to control the real-time forwarding of reports generated by the application process that hosts the subservice and by other application processes.

This subservice type provides means to control the forwarding of reports taking into account their operational use. That control is performed, per application process, by defining application process related forwarding control conditions that when met authorize or not the forwarding of related reports.

This subservice type includes the capability for defining control conditions at application process related report type level i.e.:

- conditions for all report types of an application process;
- conditions for all report types of a specific service type of an application process;
- conditions for a specific report type of an application process.

If no application process forward-control definition is defined for an application process, this implies that no report from that application process is forwarded to ground in real-time.

This subservice type includes optional capabilities for defining control conditions at:

- housekeeping parameter report structure level;
- event definition level.

If no housekeeping parameter report structure forwarding control conditions are defined for an application process, this implies that no housekeeping parameter report from that application process is forwarded to ground in real-time.

If no event definition blocking control conditions are defined for an application process, this implies that all event reports from that application process are forwarded to ground in real-time.

6.14.2 Service layout

6.14.2.1 Subservice

6.14.2.1.1 Real-time forwarding control subservice

a. Each real-time forwarding control service shall contain at least one real-time forwarding control subservice.

6.14.2.2 Application process

a. Each application process shall host at most one real-time forwarding control subservice provider.

6.14.3 Real-time forwarding control subservice

6.14.3.1 Accessibility

6.14.3.1.1 Application process

a. The list of application processes that are controlled by the real-time forwarding control subservice shall be declared when specifying that subservice.

NOTE The real-time forwarding control subservice always controls the report forwarding for reports generated by the application process that hosts that subservice.

b. The real-time forwarding control subservice shall be able to handle, at any time, all reports that are generated by each application process that is controlled by that subservice.

6.14.3.2 Real-time forwarding control configuration

a. The real-time forwarding control subservice configuration policy to apply when starting and restarting the real-time forwarding control subservice shall be declared when specifying the subservice.

NOTE in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

6.14.3.3 Forward-control definitions

6.14.3.3.1 Capability

a. Whether the real-time forwarding control subservice provides the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports shall be declared when specifying that subservice.

NOTE 1 See clause 6.14.3.3.3.

NOTE 2 For the housekeeping parameter reports, refer to requirement 6.3.3.4a.

b. Whether the real-time forwarding control subservice provides the capability to control, per event definition, the forwarding of event reports shall be declared when specifying that subservice.

NOTE 1 See clause 6.14.3.3.4.

NOTE 2 For the event reports, refer to requirement 6.5.4a.

c. If the real-time forwarding control subservice provides the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports, the subservice capability to subsample the forwarding of the parameter reports shall be declared when specifying that subservice.

NOTE Refer to requirement 6.14.3.3.1a.

6.14.3.3.2 Application process forward-control configuration

a. The maximum number of application process forward-control definitions that the real-time forwarding control subservice can contemporaneously control shall, at any time, correspond to the number of application processes that are controlled by that subservice.

NOTE 1 See requirement 6.14.3.1.1a.

NOTE 2 The application process forward-control configuration contains the application process forward-control definitions of the real-time forwarding control subservice.

b. Each application process forward-control definition shall contain:

1. the identifier of the application process to control;
2. a list of zero or more application process related "service type forward-control definitions", each one containing:
   (a) the identifier of the service type to control;
   (b) a list of zero or more application process and service type related "report type forward-control definitions", each one containing the message subtype identifier of a report type.

NOTE 1 The real-time forwarding control subservice has knowledge about the application processes that it controls but no knowledge about the service types and report types that they can generate. This lack of knowledge results in the possibility for the subservice to handle on-board, service type forward-control definitions or report type forward-control definitions that can be meaningless. It is of ground operations responsibility to ensure consistency in this respect.

NOTE 2 An empty application process forward-control configuration (i.e. no application process forward-control definition is defined) implies that the subservice blocks all reports. Blocking means that these reports are not forwarded to ground.

NOTE 3 If the subservice provides none of the capabilities specified in requirements 6.14.3.3.1a, 1.1.1.1.1a and 6.14.3.3.1b, a report is forwarded to ground only if it fulfils one of the following conditions:

- an application process forward-control definition with no service type forward-control definition is defined for the application process identifier of that report;
- an application process forward-control definition with a service type forward-control definition that has no report type forward-control definition is defined for the application process identifier and the service type of that report;
- an application process forward-control definition with a service type forward-control definition is defined that has a report type forward-control definition for the application process identifier and the service type and the message subtype identifier of that report.

c. The maximum number of service type forward-control definitions that can be contained within an application process forward-control definition shall be declared when specifying the real-time forwarding control subservice.

d. The maximum number of report type forward-control definitions that can be contained within a service type forward-control definition shall be declared when specifying the real-time forwarding control subservice.

6.14.3.3.3 Housekeeping parameter report forward-control configuration

a. The maximum number of housekeeping parameter report forward-control definitions that the real-time forwarding control subservice can contemporaneously control shall, at any time, correspond to the number of application processes that are controlled by that subservice and that provide the capability for generating housekeeping parameter reports.

NOTE 1 For the number of application processes, see requirement 6.14.3.1.1a.

NOTE 2 The housekeeping parameter report forward-control configuration contains the housekeeping parameter report forward-control definitions of the real-time forwarding control subservice.

b. Each housekeeping parameter report forward-control definition shall contain:

1. the identifier of the application process;
2. a list of zero or more related housekeeping parameter report structure identifiers.

NOTE 1 An empty housekeeping parameter report forward-control configuration (i.e. no housekeeping

parameter report forward-control definition is defined) implies that the subservice blocks all housekeeping parameter reports.

NOTE 2 A housekeeping parameter report is forwarded to ground only if the application process forward-control configuration does not block that report and one of the following conditions occurs:

- a housekeeping parameter report forward-control definition with no housekeeping parameter report structure identifiers is defined for the application process identifier of that report;
- a housekeeping parameter report forward-control definition with a housekeeping parameter report structure identifier is defined for the application process identifier and the housekeeping parameter report structure identifier of that report.

c. The maximum number of housekeeping parameter report structure identifiers that can be contained within a housekeeping parameter report forward-control definition shall be declared when specifying the real-time forwarding control subservice.

6.14.3.3.4 Event report blocking forward-control configuration

a. The maximum number of event report blocking forward-control definitions that the real-time forwarding control subservice can contemporaneously control shall, at any time, correspond to the number of application processes that are controlled by that subservice and that provide the capability for generating event reports.

NOTE 1 For the number of application processes, see requirement 6.14.3.1.1a.

NOTE 2 The event report blocking forward-control configuration contains the event report blocking forward-control definitions of the real-time forwarding control subservice.

b. Each event report blocking forward-control definition shall contain:

1. the identifier of the application process;
2. a list of zero or more related event definition identifiers.

NOTE 1 An empty event report blocking forward-control configuration (i.e. no event report blocking forward-control definition is defined) implies that an event report is forwarded to ground if the application process forward-control configuration does not block that report.

NOTE 2 The forwarding of an event report to ground is blocked if any of the following conditions occurs:

- the application process forward-control configuration blocks that report;

- the application process forward-control configuration does not block that report and an event report blocking forward-control definition with no event definition identifiers is defined for the application process identifier of that report;
- the application process forward-control configuration does not block that report and an event report blocking forward-control definition with an event definition identifier is defined for the application process identifier and the event definition identifier of that report.

c. The maximum number of event definition identifiers that can be contained within an event report blocking forward-control definition shall be declared when specifying that real-time forwarding control subservice.

6.14.3.4 Forwarding control processing logic

a. The real-time forwarding control subservice shall block the forwarding to ground of a report if the application process identifier of that report is not contained within an application process forward-control definition.

b. The real-time forwarding control subservice shall block the forwarding to ground of each report that fulfils all of the following conditions:
   1. the application process identifier of that report is contained within an application process forward-control definition, and
   2. that application process forward-control definition contains at least one service type forward-control definition, and
   3. that application process forward-control definition does not contain a service type forward-control definition for the service type of that report.

c. The real-time forwarding control subservice shall block the forwarding to ground of each report that fulfils all of the following conditions:
   1. the application process identifier of that report is contained within an application process forward-control definition, and
   2. that application process forward-control definition contains a service type forward-control definition for the service type of that report, and
   3. that service type forward-control definition contains at least one report type forward-control definition, and
   4. that service type forward-control definition does not contain a report type forward-control definition for the report type of that report.

d. If the real-time forwarding control subservice provides the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports, the subservice shall block the forwarding to ground of a housekeeping report if the application process identifier of that report is not contained within a housekeeping parameter report forward-control definition,

e. If the real-time forwarding control subservice provides the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports, the subservice shall block the forwarding to ground of each housekeeping report that fulfils all of the following conditions:

1. the application process identifier of that report is contained within a housekeeping parameter report forward-control definition and
2. that housekeeping parameter report forward-control definition contains at least one housekeeping parameter report structure identifier, and
3. that housekeeping parameter report forward-control definition does not contain the housekeeping parameter report structure identifier of that report.

NOTE The real-time forwarding to ground of a housekeeping parameter report structure of an application process is enabled if it is blocked neither by the application process forwarding control configuration nor by the housekeeping parameter report forwarding control configuration.

f. If the real-time forwarding control subservice provides the capability to control, per event definition, the forwarding of event reports, the subservice shall block the forwarding to ground of an event report if that report fulfils all of the following conditions:

1. the application process identifier of that report is contained within an event report blocking forward-control definition, and
2. that event report blocking forward-control definition has no event definition identifier.

g. If the real-time forwarding control subservice provides the capability to control, per event definition, the forwarding of event reports, the subservice shall block the forwarding to ground of an event report if that report fulfils all of the following conditions:

1. the application process identifier of that report is contained within an event report blocking forward-control definition, and
2. that event report blocking forward-control definition contains the event definition identifier of that report.

NOTE The real-time forwarding to ground of an event definition of an application process is enabled if it is blocked neither by the application process forwarding control configuration nor by the event report blocking control configuration.

6.14.3.5 Managing the application process forward-control configuration

6.14.3.5.1 Add report types to the application process forward-control configuration

a. The real-time forwarding control subservice shall provide the capability to add report types to the application process forward-control configuration.

NOTE 1 The corresponding requests are of message type "TC[14,1] add report types to the application process forward-control configuration".

NOTE 2 For the capability to delete report types from the application process forward-control configuration, refer to clause 6.14.3.5.2.

b. Each request to add report types to the application process forward-control configuration shall contain any combination of ordered lists of one or more instructions:

1. to add a report type to the application process forward-control configuration,
2. to add all report types of a service type to the application process forward-control configuration,
3. to add all report types of an application process to the application process forward-control configuration.

c. Each instruction to add a report type to the application process forward-control configuration shall contain:

1. the application process identifier addressed by that instruction,
2. the message type identifier consisting of:
   (a) the service type identifier;
   (b) the message subtype identifier.

d. Each instruction to add all report types of a service type to the application process forward-control configuration shall contain:

1. the application process identifier addressed by that instruction,
2. the service type identifier.

e. Each instruction to add all report types of an application process to the application process forward-control configuration shall contain:

1. the application process identifier addressed by that instruction.

f. The real-time forwarding control subservice shall reject any request to add report types to the application process forward-control configuration if of the following conditions occurs:

1. that request contains an instruction that implies the addition of a service type forward-control definition and the maximum number of service type forward-control definitions for the corresponding application process forward-control definition is already reached;
2. the maximum number of report type forward-control definitions that can be contained within the corresponding service type forward-control definition is already reached;
3. the corresponding service type forward-control definition has no report type forward-control definition already defined;
4. the corresponding application process forward-control definition has no service type forward-control definition already defined;

NOTE 1 For item 3, if the forwarding of all report types of a service type is enabled, it is meaningless to ask for the addition of a report type for that service type.

NOTE 2 For item 4, if the forwarding of all report types of an application process is enabled, it is meaningless to ask for the addition of a report type for that application process.

5. that request contains an instruction that implies the addition of a service type forward-control definition and the maximum number of service type forward-control definition for the corresponding application process forward-control definition is already reached;
6. the corresponding application process forward-control definition has no service type forward-control definition already defined;

NOTE For item 2, if the forwarding of all report types of an application process is enabled, it is meaningless to ask for the addition of a service type for that application process.

7. that instruction refers to an application process that is not controlled by that subservice.

g. For each request to add report types to the application process forward-control configuration that is rejected, the real-time forwarding control subservice shall generate a failed start of execution notification.

h. For each request to add report types to the application process forward-control configuration that contains only valid instructions, the real-time forwarding control subservice shall execute those instructions in the order of their appearance in that request

i. For each valid instruction to add a report type to the application process forward-control configuration, the real-time forwarding control subservice shall:

1. add, for the specified application process identifier, an application process forward-control definition if not already existing;
2. add, for the related application process forward-control definition and the specified service type identifier, a service type forward-control definition, if not already existing;
3. add, for the related service type forward-control definition and the specified message subtype identifier, a report type forward-control definition, if not already existing.

j. For each valid instruction to add all report types of a service type to the application process forward-control configuration, the real-time forwarding control subservice shall:

1. add, for the specified application process identifier, an application process forward-control definition if not already existing;
2. add, for the related application process forward-control definition and the specified service type identifier, a service type forward-control definition to the related application process forward-control definition, if not already existing;
3. delete, if any, all report type forward-control definitions of the related service type forward-control definition.

k. For each valid instruction to add all report types of an application process to the application process forward-control configuration, the real-time forwarding control subservice shall:

1. add, for the specified application process identifier, an application process forward-control definition if not already existing;
2. delete, if any, all service type forward-control definitions of the related application process forward-control definition.

6.14.3.5.2 Delete report types from the application process forward-control configuration

a. The real-time forwarding control subservice shall provide the capability to delete report types from the application process forward-control configuration.

NOTE 1 The corresponding requests are of message type "TC[14,2] delete report types from the application process forward-control configuration".

NOTE 2 For the capability to add report types to the application process forward-control configuration, refer to clause 6.14.3.5.1.

b. Each request to delete report types from the application process forward-control configuration shall contain exactly one of:

1. any combination of ordered lists of one or more instructions:
   (a) to delete a report type from the application process forward-control configuration,
   (b) to delete a service type from the application process forward-control configuration,
   (c) to delete an application process from the application process forward-control configuration,
2. a single instruction to empty the application process forward-control configuration.

NOTE The instructions to empty the application process forward-control configuration contain no argument.

c. Each instruction to delete a report type from the application process forward-control configuration shall contain:

1. the application process identifier addressed by that instruction,
2. the report type identifier consisting of:
   (a) the service type identifier;
   (b) the message subtype identifier.

d. Each instruction to delete a service type from the application process forward-control configuration shall contain:

1. the application process identifier addressed by that instruction,
2. the service type identifier.

e. Each instruction to delete an application process from the application process forward-control configuration shall contain:

1. the application process identifier addressed by that instruction.

f. The real-time forwarding control subservice shall reject any request to delete report types from the application process forward-control configuration if any of the following conditions occurs:

1. that request contains an instruction that refers to a report type identifier that is not in the application process forward-control configuration;
2. that request contains an instruction that refers to a service type identifier that is not in the application process forward-control configuration;
3. that request contains an instruction that refers to an application process identifier that is not in the application process forward-control configuration.

g. For each request to delete report types from the application process forward-control configuration that is rejected, the real-time forwarding control subservice shall generate a failed start of execution notification.

h. For each request to delete report types from the application process forward-control configuration that contains only valid instructions, the real-time forwarding control subservice shall execute those instructions in the order of their appearance in that request.

i. For each valid instruction to delete a report type from the application process forward-control configuration, the real-time forwarding control subservice shall:

1. delete the report type forward-control definition related to that specified application process identifier, service type identifier and message subtype identifier;
2. if that report type forward-control definition deletion results in an emptied service type forward-control definition, delete that service type forward-control definition;
3. if that service type forward-control definition deletion results in an emptied application process forward-control definition, delete that application process forward-control definition.

j. For each valid instruction to delete a service type from the application process forward-control configuration, the real-time forwarding control subservice shall:

1. delete the service type forward-control definition related to that specified application process identifier and service type identifier;
2. if that service type forward-control definition deletion results in an emptied application process forward-control definition, delete that application process forward-control definition.

k. For each valid instruction to delete an application process from the application process forward-control configuration, the real-time forwarding control subservice shall:

1. delete the application process forward-control definition related to that specified application process identifier.

l. For each valid instruction to empty the application process forward-control configuration, the real-time forwarding control subservice shall:

1. delete, if any, all application process forward-control definitions.

6.14.3.5.3 Report the content of the application process forward-control configuration

a. The real-time forwarding control subservice capability to report the content of the application process forward-control configuration shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[14,3] report the content of the application process forward-control configuration". The responses are data reports of message type "TM[14,4] application process forward-control configuration content report".

NOTE 2 That capability requires the capability for that subservice to add report types to the application process forward-control configuration, refer to clause 6.14.3.5.1.

b. Each request to report the content of the application process forward-control configuration shall contain exactly one instruction to report the content of the application process forward-control configuration.

NOTE The instructions to report the content of the application process forward-control configuration contain no argument.

c. For each valid instruction to report the content of the application process forward-control configuration, the real-time forwarding control subservice shall generate, for each existing application process forward-control definition, a single application process forward-control definition notification that includes:

1. the related application process identifier;
2. for each related service type forward-control definition, if any:
   (a) the related service type identifier;
   (b) for each related report type forward-control definition, if any, the related message subtype identifier.

d. For each valid request to report the content of the application process forward-control configuration, the real-time forwarding control subservice shall generate a single application process forward-control configuration content report that includes all related application process forward-control definition notifications.

6.14.3.6 Managing the housekeeping parameter report forward-control configuration

6.14.3.6.1 Add structure identifiers to the housekeeping parameter report forward-control configuration

a. The real-time forwarding control subservice shall provide the capability to add structure identifiers to the housekeeping parameter report forward-control configuration if that subservice provides the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports.

NOTE 1 The corresponding requests are of message type "TC[14,5] add structure identifiers to the housekeeping parameter report forward-control configuration".

NOTE 2 For the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports, refer to requirement 6.14.3.3.1a.

NOTE 3 For the capability to delete structure identifiers from the housekeeping parameter report forward-control configuration, refer to clause 6.14.3.6.2.

b. Each request to add structure identifiers to the housekeeping parameter report forward-control configuration shall contain exactly one of:

1. an ordered list of one or more structure identifiers to the application process housekeeping parameter report forward-control configuration;
2. an instruction to add all structure identifiers to the housekeeping parameter report forward-control configuration.

c. Each instruction to add a structure identifier to the housekeeping parameter report forward-control configuration shall contain:

1. the application process identifier addressed by that instruction;
2. the housekeeping parameter report structure identifier;
3. if subsampling is supported, the subsampling rate.

NOTE For item 3, refer to requirement 6.14.3.3.1c.

d. Each instruction to add all structure identifiers to the housekeeping parameter report forward-control configuration shall contain:

1. the application process identifier addressed by that instruction.

e. The real-time forwarding control subservice shall reject any request to add structure identifiers to the housekeeping parameter report forward-control configuration if any of the following conditions occurs:

1. that request contains an instruction that refers to an application process that is not controlled by that subservice;
2. the maximum number of housekeeping parameter report structure identifiers that can be contained within a housekeeping parameter report forward-control definition is already reached;

3. the corresponding housekeeping parameter report forward-control definition has no structure identifier already defined;

4. that request contains an instruction that refers to an application process that is not controlled by that subservice.

f. For each request contained within a request to add structure identifiers to the housekeeping parameter report forward-control configuration that is rejected, the real-time forwarding control subservice shall generate a failed start of execution notification.

g. The real-time forwarding control subservice shall shall reject any instruction contained within a request to add structure identifiers to the housekeeping parameter report forward-control configuration if:

1. that instruction refers to an application process that is not controlled by that subservice.

h. For each valid instruction to add a structure identifier to the housekeeping parameter report forward-control configuration, the real-time forwarding control subservice shall:

1. add, for the specified application process identifier, a housekeeping parameter report forward-control definition if not already existing;

2. add, to the related housekeeping parameter report forward-control definition, the specified housekeeping parameter report structure identifier, if not already existing;

3. if subsampling is supported, set, to the related housekeeping parameter report forward-control definition and the specified housekeeping parameter report structure identifier, the specified subsampling rate.

NOTE For item 3, refer to requirement 6.14.3.3.1c.

i. For each valid instruction to add all structure identifiers to the housekeeping parameter report forward-control configuration, the real-time forwarding control subservice shall:

1. add, for the specified application process identifier, a housekeeping parameter report forward-control definition if not already existing;

2. delete, if any, all housekeeping parameter report structure identifiers of the related housekeeping parameter report forward-control definition.

NOTE For item 2, deleting a housekeeping parameter report structure identifier implies deleting the corresponding subsampling rate if any (see also requirement 6.14.3.3.1c).

6.14.3.6.2 Delete structure identifiers from the housekeeping parameter report forward-control configuration

a. The real-time forwarding control subservice shall provide the capability to delete structure identifiers from the housekeeping parameter report forward-control configuration if that subservice provides the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports.

NOTE 1 The corresponding requests are of message type "TC[14,6] delete structure identifiers from the housekeeping parameter report forward-control configuration".

NOTE 2 For the capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports, refer to requirement 6.14.3.3.1a.

NOTE 3 For the capability to add structure identifiers to the housekeeping parameter report forward-control configuration, refer to clause 6.14.3.6.1.

b. Each request to delete structure identifiers from the housekeeping parameter report forward-control configuration shall contain exactly one of:

1. any combination of ordered lists of one or more instructions:
   (a) to delete a structure identifier from the housekeeping parameter report forward-control configuration,
   (b) to delete an application process from the housekeeping parameter report forward-control configuration;
2. a single instruction to empty the housekeeping parameter report forward-control configuration.

NOTE The instructions to empty the housekeeping parameter report forward-control configuration contain no argument.

c. Each instruction to delete a structure identifier from the housekeeping parameter report forward-control configuration shall contain:

1. the application process identifier addressed by that instruction;
2. the housekeeping parameter report structure identifier;

d. Each instruction to delete an application process from the housekeeping parameter report forward-control configuration shall contain:

1. the application process identifier addressed by that instruction.

e. The real-time forwarding control subservice shall reject any request to delete a structure identifier from the housekeeping parameter report forward-control configuration if:

1. that request contains an instruction that refers to an application process identifier that is not in the housekeeping parameter report forward-control configuration;
2. that request contains an instruction that refers to a housekeeping parameter report structure identifier that is not in the housekeeping parameter report forward-control definition for the specified application process identifier;
3. that request contains an instruction that refers to an application process identifier that is not in the housekeeping parameter report forward-control configuration.

f. For each request to delete structure identifiers from the housekeeping parameter report forward-control configuration that is rejected, the real-

time forwarding control subservice shall generate a failed start of execution notification.

g. For each request to delete structure identifiers from the housekeeping parameter report forward-control configuration that contains only valid instructions, the real-time forwarding control subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to delete a structure identifier from the housekeeping parameter report forward-control configuration , the real-time forwarding control subservice shall:

1. delete the housekeeping parameter report structure identifier related to the specified application process identifier;

2. if that housekeeping parameter report structure identifier deletion results in an emptied housekeeping parameter report forward-control definition, delete that housekeeping parameter report forward-control definition.

NOTE Deleting a housekeeping parameter report structure identifier implies deleting the corresponding subsampling rate if any (see also requirement 6.14.3.3.1c).

i. For each valid instruction to delete an application process from the housekeeping parameter report forward-control configuration, the real-time forwarding control subservice shall:

1. delete the housekeeping parameter report forward-control definition for the specified application process identifier.

j. For each valid instruction to empty the housekeeping parameter report forward-control configuration, the real-time forwarding control subservice shall:

1. delete all housekeeping parameter report forward-control definitions.

6.14.3.6.3 Report the content of the housekeeping parameter report forward-control configuration

a. The real-time forwarding control subservice capability to report the content of the housekeeping parameter report forward-control configuration shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[14,7] report the content of the housekeeping parameter report forward-control configuration". The responses are data reports of message type "TM[14,8] housekeeping parameter report forward-control configuration content report".

NOTE 2 That capability requires the capability for that subservice to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports (refer to requirement 6.14.3.3.1a).

b. Each request to report the content of the housekeeping parameter report forward-control configuration shall contain exactly one instruction to

report the content of the housekeeping parameter report forward-control configuration.

NOTE The instructions to report the content of the housekeeping parameter report forward-control configuration contain no argument.

c. For each valid instruction to report the content of the housekeeping parameter report forward-control configuration, the real-time forwarding control subservice shall generate, for each existing housekeeping parameter report forward-control definition, a single housekeeping parameter report forward-control definition notification that includes:

1. the related application process identifier;
2. for each housekeeping parameter report structure identifier entry:
   (a) the housekeeping parameter report structure identifier;
   (b) if subsampling is supported, the subsampling rate.

   NOTE For item 2(b), refer to requirement 6.14.3.3.1c.

d. For each valid request to report the content of the housekeeping parameter report forward-control configuration, the real-time forwarding control subservice shall generate a single housekeeping parameter report forward-control configuration content report that includes all related housekeeping parameter report forward-control definition notifications.

6.14.3.7 Managing the event report blocking forward-control configuration

6.14.3.7.1 Delete event definition identifiers from the event report blocking forward-control configuration

a. The real-time forwarding control subservice shall provide the capability to delete event definition identifiers from the event report blocking forward-control configuration if that subservice provides the capability to control, per event definition, the forwarding of event reports.

NOTE 1 The corresponding requests are of message type "TC[14,13] delete event definition identifiers from the event report blocking forward-control configuration".

NOTE 2 For the capability to control, per event definition, the forwarding of event reports, refer to requirement 6.14.3.3.1b.

NOTE 3 For the capability to add event definition identifiers to the event report blocking forward-control configuration, refer to clause 6.14.3.7.2.

b. Each request to delete event definition identifiers from the event report blocking forward-control configuration shall include exactly one of:

1. any combination of ordered list of one or more instructions:
   (a) to delete an event definition identifier from the event report blocking forward-control configuration,
   (b) to delete an application process from the event report blocking forward-control configuration;

2. a single instruction to empty the event report blocking forward-control configuration.

NOTE The instructions to empty the event report blocking forward-control configuration contain no argument.

c. Each instruction to delete an event definition identifier from the event report blocking forward-control configuration shall contain:

1. the application process identifier addressed by that instruction;
2. the event definition identifier.

d. Each instruction to delete an application process from the event report blocking forward-control configuration shall contain:

1. the application process identifier addressed by that instruction.

e. The real-time forwarding control subservice shall reject any request to delete event definition identifiers from the event report blocking forward-control configuration if any of the following conditions occurs:

1. that request contains an instruction that refers to an application process identifier that is not in the event report blocking forward-control configuration;
2. that request contains an instruction that refers to an event definition identifier that is not in the event report blocking forward-control definition for the specified application process;
3. that request contains an instruction that refers to an application process identifier that is not in the event report blocking forward-control configuration.

f. For each instruction to delete an event definition identifier from the event report blocking forward-control configuration that is rejected, the real-time forwarding control subservice shall generate a failed start of execution notification.

g. For each instruction to delete an event definition identifier from the event report blocking forward-control configuration that contains only valid instructions, the real-time forwarding control subservice shall execute those instructions in the order of their appearance in that request

h. For each valid instruction to delete an event definition identifier from the event report blocking forward-control configuration, the real-time forwarding control subservice shall:

1. delete the event definition identifier related to the specified application process identifier;
2. if that event definition identifier deletion results in an emptied event report blocking forward-control definition, delete that event report blocking forward-control definition.

i. For each valid instruction to delete an application process from the event report blocking forward-control configuration, the real-time forwarding control subservice shall:

1. delete the event report blocking forward-control definition for the specified application process identifier.

j. For each valid instruction to empty the event report blocking forward-control configuration, the real-time forwarding control subservice shall:

1. delete all event report blocking forward-control definitions.

6.14.3.7.2 Add event definition identifiers to the event report blocking forward-control configuration

a. The real-time forwarding control subservice shall provide the capability to add event definition identifiers to the event report blocking forward-control configuration if that subservice provides the capability to control, per event definition, the forwarding of event reports.

NOTE 1 The corresponding requests are of message type "TC[14,14] add event definition identifiers to the event report blocking forward-control configuration".

NOTE 2 For the capability to control, per event definition, the forwarding of event reports, refer to requirement 6.14.3.3.1b.

NOTE 3 For the capability to delete event definition identifiers from the event report blocking forward-control configuration, refer to clause 6.14.3.7.1.

b. Each request to add event definition identifiers to the event report blocking forward-control configuration shall contain exactly one of:

1. an ordered list of one or more instructions to add an event definition identifier to the event report blocking forward-control configuration,
2. an instruction to add all event definition identifiers to the event report blocking forward-control configuration.

c. Each instruction to add an event definition identifier to the event report blocking forward-control configuration shall contain:

1. the application process identifier addressed by that instruction;
2. the event definition identifier.

d. Each instruction to add all event definition identifiers to the event report blocking forward-control configuration shall contain:

1. the application process identifier addressed by that instruction.

e. The real-time forwarding control subservice shall reject any request to add event definition identifiers to the event report blocking forward-control configuration if any of the following conditions occurs:

1. that request contains an instruction that refers to an application process that is not controlled by that subservice;
2. the maximum number of event definition identifiers that can be contained within an event report blocking forward-control definition is already reached;
3. the corresponding event report blocking forward-control definition has no event definition identifier already defined;
4. that request contains an instruction that refers to an application process that is not controlled by that subservice.

f. For each request to add event definition identifiers to the event report blocking forward-control configuration that is rejected, the real-time forwarding control subservice shall generate a failed start of execution notification.

g. The real-time forwarding control subservice shall reject any instruction contained within a request to add event definition identifiers to the event report blocking forward-control configuration if:

1. that instruction refers to an application process that is not controlled by that subservice.

h. For each valid instruction to add an event definition identifier to the event report blocking forward-control configuration, the real-time forwarding control subservice shall:

1. add, for the specified application process identifier, an event report blocking forward-control definition if not already existing;
2. add, to the related event report blocking forward-control definition, the specified event definition identifier, if not already existing.

i. For each valid instruction to add all event definition identifiers to the event report blocking forward-control configuration, the real-time forwarding control subservice shall:

1. add, for the specified application process identifier, an event report blocking forward-control definition if not already existing;
2. delete, if any, all event definition identifiers of the related event report blocking forward-control definition.

6.14.3.7.3 Report the content of the event report blocking forward-control configuration

a. The real-time forwarding control subservice capability to report the content of the event report blocking forward-control configuration shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[14,15] report the content of the event report blocking forward-control configuration". The responses are data reports of message type "TM[14,16] event report blocking forward-control configuration content report".

NOTE 2 That capability requires the capability for that subservice to control, per event definition, the forwarding of event reports, refer to requirement 6.14.3.3.1b.

b. Each request to report the content of the event report blocking forward-control configuration shall contain exactly one instruction to report the content of the event report blocking forward-control configuration.

NOTE The instructions to report the content of the event report blocking forward-control configuration contain no argument.

c. For each valid instruction to report the content of the event report blocking forward-control configuration, the real-time forwarding control

subservice shall generate, for each existing event report blocking forward-control definition, a single event report blocking forward-control definition notification that includes:

1. the related application process identifier;
2. for each event definition identifier entry:
   (a) the event definition identifier.

d. For each valid request to report the content of the event report blocking forward-control configuration, the real-time forwarding control subservice shall generate a single event report blocking forward-control configuration content report that includes all related event report blocking forward-control definition notifications.

6.14.4 Subservice observables

a. The following observables shall be defined for the real-time forwarding control subservice:
   1. counter of packets that were programmed for real-time forwarding but could not be transmitted to ground.

6.15 ST[15] on-board storage and retrieval

6.15.1 Scope

6.15.1.1 General

The on-board storage and retrieval service type provides the capability:

- to select reports generated by other on-board services and store them in packet stores;
- to allow the ground system to manage the reports in the packet stores and request their downlink.

The capability to store telemetry packets on-board and dump them, on request, to the ground is especially appropriate under the following circumstances:

- When ground station coverage is intermittent or when real-time telemetry bandwidth is limited. In this case, the on-board storage capacity is sized to store all packets generated on-board for spacecraft monitoring and control purposes, for a duration at least equal to the longest non-coverage period plus a mission-dependent margin. These packets are then retrieved during subsequent ground station passes according to a selection strategy based upon the operational significance of the stored packets.
- To recover lost packets. For missions with continuous ground coverage, the loss of packets can be solved by retaining on-board the set of the most recent packets. The size of the set is a mission-specific configuration parameter.

The on-board storage and retrieval service type defines two standardized subservice types, i.e.:

- the storage and retrieval subservice type;
- the packet selection subservice type.

An on-board storage and retrieval service contains one storage and retrieval subservice and one or more packet selection subservices. The interaction between the storage and retrieval subservice and the packet selection subservices is beyond the scope of this Standard.

It is noted that the capabilities offered by the on-board storage and retrieval service type are fully independent of those provided by the real-time forwarding control service type specified in clause 6.14.

6.15.1.2 Storage and retrieval subservice

The storage and retrieval subservice type provides the capability for storing the telemetry packets in the packet stores and retrieving them later, on ground request.

Each packet store can be managed according to either:

- the ‘circular management’ policy: the oldest packets are overwritten when the packet store is full, hence the packet store contains the most recently generated packets, or

- the ‘bounded management’ policy: storage terminates when the packet store is full.

Two retrieval modes are provided by the storage and retrieval subservice type:

- the "open retrieval mode" that retrieves the packets that are newer than the date of the last transmitted packet before the retrieval stopped;
- the "by-time-range mode" that retrieves the packets that are time stamped within two dates. This retrieval model is used to recover from packet loss during the downlink.

6.15.1.3 Packet selection subservice

The packet selection subservice type can be used to control the storage into packet stores of reports generated by any application process.

This subservice type provides means to control the storage of reports taking into account their operational use. That control is performed, per packet store, by defining application process related storage control conditions that when met authorize or not the storage of related reports to the corresponding packet store.

This subservice type includes the capability for defining packet store control conditions at application process related report type level i.e.:

- conditions for all report types of an application process;
- conditions for all report types of a specific service type of an application process;
- conditions for a specific report type of an application process.

If no application process storage-control definition is defined for an application process, this implies that no report from that application process is stored in the packet stores.

This subservice type includes optional capabilities for defining control conditions at:

- housekeeping parameter report structure level;
- event definition level.

For a packet store, if no housekeeping parameter report structure storage control conditions are defined for an application process, this implies that no housekeeping parameter report from that application process is stored in that packet store.

For a packet store, if no event definition blocking control conditions are defined for an application process, this implies that all event reports from that application process are stored in that packet store.

The packet selection subservice can be implemented by the application processes that generate the reports, by any application processes that route these reports, or by the application process that implements the storage and retrieval subservice.

6.15.2 Service layout

6.15.2.1 Subservice

6.15.2.1.1 Storage and retrieval subservice

a. Each on-board storage and retrieval service shall contain exactly one storage and retrieval subservice.

6.15.2.1.2 Packet selection subservice

a. Each on-board storage and retrieval service shall contain at least one packet selection subservice.

6.15.2.2 Application process

6.15.2.2.1 Storage and retrieval subservice

a. Each application process shall host at most one storage and retrieval subservice provider.

6.15.2.2.2 Packet selection subservice

a. Each application process shall host at most one packet selection subservice provider.

6.15.3 Storage and retrieval subservice

6.15.3.1 Storage and retrieval configuration

a. The configuration policy to apply when starting and restarting the storage and retrieval subservice shall be declared when specifying the subservice.

NOTE in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

b. The storage and retrieval subservice functionality and related configuration shall remain operational during the transitions between spacecraft modes.

NOTE for example, the packet storage, routing configurations and position of retrieval pointers remain unaffected after a transition to safe mode or after a processor module reboot

6.15.3.2 Packet store

a. The maximum number of packet stores that the storage and retrieval subservice can contemporaneously maintain at any time shall be declared when specifying that subservice.

b. The list of pre-defined packet stores maintained by the storage and retrieval subservice shall be declared when specifying that subservice.

c. Each packet store shall be managed by exactly one storage and retrieval subservice.

NOTE Within a subservice, each packet store is uniquely identified by a packet store identifier. The meaning and internal structure of that packet store identifier are beyond the scope of this Standard. A packet store identifier can for example be the name of an object in memory.

d. Whether the storage and retrieval subservice supports the capability to manage the packet stores of circular type shall be declared when specifying that subservice.

e. Whether the storage and retrieval subservice supports the capability to manage the packet stores of bounded type shall be declared when specifying that subservice.

f. For each packet store, the circular or bounded type of that packet store shall be declared when specifying that packet store.

NOTE The packet store type is either circular or bounded. A circular packet store implies that when a packet store is full, the new packets are stored by overwriting the oldest packets. A bounded packet store implies that when the packet store is full, the new packets are discarded.

g. The list of virtual channels that can be used by the storage and retrieval subservice shall be declared when specifying that subservice.

h. For each packet store, the virtual channel used to transmit the packets retrieved from that packet store shall be declared when specifying that packet store.

NOTE Refer to clause 6.15.3.10.4 for changing the virtual channel used by a packet store.

i. Whether the storage and retrieval subservice supports concurrent retrieval requests executing in parallel shall be declared when specifying that subservice.

j. If the storage and retrieval subservice supports concurrent retrieval requests executing in parallel, the maximum number of concurrent retrieval requests supported shall be declared when specifying that subservice.

NOTE 1 If the subservice provides both the open retrieval capability and the by-time-range retrieval capability, that maximum number of concurrent retrieval requests covers both the open retrieval requests and the by-time-range retrieval requests.

NOTE 2 For a given packet store, there can only be at most one open retrieval and one by-time-range retrieval executing in parallel.

k. The storage and retrieval subservice provide the capability to queue the retrieval requests pending their execution.

l. The maximum number of retrieval requests that the storage and retrieval subservice can queue pending their execution shall be declared when specifying that subservice.

m. The retrieval requests queuing policy used by the storage and retrieval subservice shall be declared when specifying that subservice.

n. Whether the storage and retrieval subservice supports prioritizing the packet retrievals from packet stores shall be declared when specifying that subservice.

NOTE If prioritizing packet retrievals is supported, a default retrieval priority is associated to each packet store. When using open retrieval or by-time-range retrieval, either the default retrieval priority or a specific one can be used.

o. If the storage and retrieval subservice supports prioritizing the packet retrievals from packet stores, the priority policy shall be declared when specifying that subservice.

NOTE In particular, how to proceed if the same retrieval priority is used for more than one packet store

p. If the storage and retrieval subservice supports prioritizing the packet retrievals from packet stores, that subservice shall not constrain the number of retrievals that can be given the same priority level.

q. For each packet store, the open retrieval policy shall be declared when specifying that packet store.

NOTE 1 the open retrieval policy indicates if the retrieval stays open when the last packet stored before the start of execution of the related request has been retrieved, i.e. it can be only suspended by telecommand.

NOTE 2 A default open retrieval policy is associated to each packet store, either the default open retrieval policy or a specific one can be used.

6.15.3.3 Time-stamping

a. For each storage and retrieval subservice, the storage time-stamping method used by that subservice shall be declared when specifying that subservice.

NOTE The storage time-stamping method can, for example, be:

- storage based, meaning that each received telemetry packet is time-stamped with the storage time of the telemetry packet;
- packet based, meaning that each received telemetry packet is time-stamped with the on-board time already present in the telemetry packet.

6.15.3.4 Controlling the packet store storage function

6.15.3.4.1 Storage process

a. For each packet store managed by the storage and retrieval subservice, the subservice shall maintain a status indicating whether the storage into that packet store is enabled or disabled.

NOTE This status is named "packet store storage status".

b. For each packet with ground destination that it receives, the storage and retrieval subservice shall:

1. time stamp that packet;
2. store that time-stamped packet into the enabled stores that match the storage selection criteria for those stores.

NOTE The time stamps referred by this requirement are subservice related. They represent the storage time of the telemetry packets. They do not have to be confused with the time contained in the header of the telemetry packets. These subservice related time stamps are used by the subservice to retrieve the telemetry packets from the stores.

6.15.3.4.2 Enable the storage function of packet stores

a. The storage and retrieval subservice shall provide the capability to enable the storage function of packet stores.

NOTE 1 The corresponding requests are of message type "TC[15,1] enable the storage function of packet stores".

NOTE 2 For the capability to disable the storage function of packet stores, refer to clause 6.15.3.4.3.

b. Each request to enable the storage function of packet stores shall contain exactly one of:

1. an ordered list of one or more instructions to enable the storage function of a packet store;
2. a single instruction to enable the storage function of all packet stores.

NOTE The instructions to enable the storage function of all packet stores contain no argument.

c. Each instruction to enable the storage function of a packet store shall contain:

1. the packet store identifier of the packet store to enable for storage.

d. The storage and retrieval subservice shall reject any request to enable the storage function of packet stores if:

1. that request contains an instruction that refers to a packet store that does not exist.

e. For each request to enable the storage function of packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to enable the storage function of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable the storage function of a packet store, the storage and retrieval subservice shall:

1. set the packet store storage status of the packet store specified in that instruction to "enabled".

h. For each valid instruction to enable the storage function of all packet stores, the storage and retrieval subservice shall:

1. for each packet store maintained by that subservice, set its packet store storage status to "enabled".

6.15.3.4.3 Disable the storage function of packet stores

a. The storage and retrieval subservice shall provide the capability to disable the storage function of packet stores.

NOTE 1 The corresponding requests are of message type "TC[15,2] disable the storage function of packet stores".

NOTE 2 For the capability to enable the storage function of packet stores, refer to clause 6.15.3.4.2.

b. Each request to disable the storage function of packet stores shall contain exactly one of:

1. an ordered list of one or more instructions to disable the storage function of a packet store;
2. a single instruction to disable the storage function of all packet stores.

NOTE The instructions to disable the storage function of all packet stores contain no argument.

c. Each instruction to disable the storage function of a packet store shall contain:

1. the packet store identifier of the packet store to disable for storage.

d. The storage and retrieval subservice shall reject any request to disable the storage function of packet stores if:

1. that request contains an instruction that refers to a packet store that does not exist.

e. For each request to disable the storage function of packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to disable the storage function of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable the storage function of a packet store, the storage and retrieval subservice shall:

1. set the packet store storage status of the packet store specified in that instruction to "disabled".

h. For each valid instruction to disable the storage function of all packet stores, the storage and retrieval subservice shall:

1. for each packet store maintained by that subservice, set its packet store storage status to "disabled".

6.15.3.5 Controlling the open retrieval function

6.15.3.5.1 Open retrieval process

a. For each packet store managed by the storage and retrieval subservice, that subservice shall maintain a status indicating whether the open retrieval function of that packet store is in-progress or suspended.

NOTE This status is named "packet store open retrieval status".

b. For each packet store whose packet store open retrieval status is "in-progress", the storage and retrieval subservice shall:

1. retrieve the stored packets chronologically according to their storage time tags starting from the open retrieval start time tag of that packet store;
2. route these packets to the virtual channel associated with that packet store.

NOTE For item 2, if the subservice supports prioritizing the packet retrievals from packet stores, the routing is done according to the retrieval policy, refer to requirements 6.15.3.2n and 6.15.3.2o.

c. The open retrieval function shall ensure that consecutive suspend and resume open retrieval operations do not cause any gap or overlap in the packet retrieval process.

NOTE Suspending the open retrieval process can result from a request to suspend the open retrieval of packet stores (refer to clause 6.15.3.5.4). This Standard does not elaborate on any other autonomous mechanism that can exist on-board.

6.15.3.5.2 Change the open retrieval start time tag of packet stores

a. The storage and retrieval subservice shall provide the capability to change the open retrieval start time tag of packet stores.

NOTE The corresponding requests are of message type "TC[15,14] change the open retrieval start time tag of packet stores".

b. Each request to change the open retrieval start time tag of packet stores shall contain:

1. an open retrieval start time tag,
2. exactly one of:
   (a) an ordered list of one or more instructions to change the open retrieval start time tag of a packet store,
   (b) a single instruction to change the open retrieval start time tag of all packet stores.

NOTE The instructions to change the open retrieval start time tag of all packet stores contain no argument.

c. Each instruction to change the open retrieval start time tag of a packet store shall contain:

1. the packet store identifier of the packet store to modify.

d. The storage and retrieval subservice shall reject any request to change the open retrieval start time tag of packet stores if any of the following conditions occurs:

1. that request refers to an open retrieval start time tag that is in the future;
2. that request contains an instruction that refers to a packet store that does not exist;
3. the packet store open retrieval status of that packet store is in-progress.

e. For each request to change the open retrieval start time tag of a packet store that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to change the open retrieval start time tag of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to change the open retrieval start time tag of a packet store, the storage and retrieval subservice shall:

1. set the open retrieval start time tag of the specified packet store to the value specified in the request.

h. For each valid instruction to change the open retrieval start time tag of all packet stores, the storage and retrieval subservice shall:

1. for each packet store maintained by that subservice, if its open retrieval status is "suspended", set its open retrieval start time tag to the value specified in that request.

6.15.3.5.3 Resume the open retrieval of packet stores

a. The storage and retrieval subservice shall provide the capability to resume the open retrieval of packet stores.

NOTE 1 The corresponding requests are of message type "TC[15,15] resume the open retrieval of packet stores".

NOTE 2 For the capability to suspend the open retrieval of packet stores, refer to clause 6.15.3.5.4.

b. Each request to resume the open retrieval of packet stores shall contain exactly one of:

1. an ordered list of one or more instructions to resume the open retrieval of a packet store;
2. a single instruction to resume the open retrieval of all packet stores.

NOTE The instructions to resume the open retrieval of all packet stores contain no argument.

c. Each instruction to resume the open retrieval of a packet store shall contain:

1. the identifier of the packet store;
2. the open retrieval policy;
3. if the storage and retrieval subservice supports prioritizing the packet retrievals from packet stores, the retrieval priority.

NOTE 1 Item 2, the open retrieval policy, refer to 6.15.3.2q.

NOTE 2 The default open retrieval policy is applied when resuming the open retrieval of all packet stores, see 6.15.3.5.3h

NOTE 3 For item 3, the retrieval priority, refer to requirements 6.15.3.2n and 6.15.3.2o.

NOTE 4 The default retrieval priority is applied when resuming the open retrieval of all packet stores, see 6.15.3.5.3h

d. The storage and retrieval subservice shall reject any request to resume the open retrieval of packet stores if any of the following conditions occurs:

1. that request contains an instruction that refers to a packet store that does not exist;
2. that subservice does not support concurrent retrieval requests executing in parallel and the packet store by-time-range retrieval status of that packet store is enabled.

NOTE For item 2, refer to requirement 6.15.3.2i.

e. For each request to resume the open retrieval of packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to resume the open retrieval of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to resume the open retrieval of a packet store, the storage and retrieval subservice shall:

1. set the packet store open retrieval status of that packet store to "in progress";
2. if that subservice supports prioritizing the packet retrievals from packet stores, start the retrieval process according to the priority policy;
3. when the last packet stored before the start of execution of the related request has been retrieved and the open retrieval policy does not indicate to stay open, set the packet store open retrieval status of that packet store to "suspended".

h. For each valid instruction to resume the open retrieval of all packet stores, the storage and retrieval subservice shall:

1. for each packet store maintained by that subservice, set the packet store open retrieval status of that packet store to "in progress";
2. for each packet store maintained by that subservice, if that subservice supports prioritizing the packet retrievals from packet stores, start the retrieval process according to the default priority policy;
3. for each packet store maintained by that subservice, when the last packet stored before the start of execution of the related request has been retrieved and the default open retrieval policy does not indicate to stay open, set the packet store open retrieval status of that packet store to "suspended".

6.15.3.5.4 Suspend the open retrieval of packet stores

a. The storage and retrieval subservice shall provide the capability to suspend the open retrieval of packet stores.

NOTE 1 The corresponding requests are of message type "TC[15,16] suspend the open retrieval of packet stores".

NOTE 2 For the capability to resume the open retrieval of packet stores, refer to clause 6.15.3.5.3.

b. Each request to suspend the open retrieval of packet stores shall contain exactly one of:
   1. an ordered list of one or more instructions to suspend the open retrieval of a packet store;
   2. a single instruction to suspend the open retrieval of all packet stores.

      NOTE The instructions to suspend the open retrieval of all packet stores contain no argument.

c. Each instruction to suspend the open retrieval of a packet store shall contain:
   1. the identifier of the packet store.

d. The storage and retrieval subservice shall reject any request to suspend the open retrieval of a packet store if:
   1. that request contains an instruction that refers to packet store that is unknown.

e. For each request to suspend the open retrieval of packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to suspend the open retrieval of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to suspend the open retrieval of a packet store, the storage and retrieval subservice shall:
   1. finalize the on-going packet retrieval transaction, if any,
   2. set the packet store open retrieval status of that packet store to "suspended".

NOTE a packet retrieval transaction implies the retrieval of a single packet from the store and its routing to ground

h. For each valid instruction to suspend the open retrieval of all packet stores, the storage and retrieval subservice shall:

1. for each packet store maintained by that subservice, finalize the on-going packet retrieval transaction, if any,
2. for each packet store maintained by that subservice, set the packet store open retrieval status of that packet store to "suspended".

NOTE a packet retrieval transaction implies the retrieval of a single packet from the store and its routing to ground

6.15.3.6 Controlling the by-time-range retrieval function

6.15.3.6.1 By-time-range retrieval process

a. Whether the storage and retrieval subservice supports the by-time-range retrieval function shall be declared when specifying that subservice.

b. For each packet store managed by the storage and retrieval subservice, that subservice shall maintain a status indicating whether the by-time-range retrieval function of that packet store is enabled or disabled.

NOTE This status is named "packet store by-time-range retrieval status".

c. For each packet store whose packet store by-time-range retrieval status is "enabled", the storage and retrieval subservice shall:

1. retrieve the stored packets chronologically according to their storage time tag, starting from the start retrieval time up to the end retrieval time;
2. route these packets to the virtual channel associated with that packet store;
3. when the end retrieval is reached, set the packet store by-time-range retrieval status to "disabled".

NOTE For item 2, if the subservice supports prioritizing the packet retrievals from packet stores, the routing is done according to the retrieval policy, refer to requirement 6.15.3.2o.

d. The maximum time required by the storage and retrieval subservice to process any request to start the by-time-range retrieval of packet stores and retrieve the first related stored packet shall be declared when specifying that subservice.

6.15.3.6.2 Start the by-time-range retrieval of packet stores

a. The storage and retrieval subservice shall provide the capability to start the by-time-range retrieval of packet stores if that subservice supports the by-time-range retrieval function.

NOTE 1 The corresponding requests are of message type "TC[15,9] start the by-time-range retrieval of packet stores".

NOTE 2 For the by-time-range retrieval function support, refer to requirement 6.15.3.6.1a.

NOTE 3 For the capability to abort the by-time-range retrieval of packet stores, refer to clause 6.15.3.6.3.

b. Each request to start the by-time-range retrieval of packet stores shall contain an ordered list of one or more instructions to start the by-time-range retrieval of a packet store.

c. Each instruction to start the by-time-range retrieval of a packet store shall contain:

1. the packet store identifier of the packet store;
2. if the storage and retrieval subservice supports prioritizing the packet retrievals from packet stores, the retrieval priority to use;
3. the retrieval time window, expressed as:
   (a) a retrieval start time, and
   (b) a retrieval end time.

NOTE For item 2, refer to requirements 6.15.3.2n and 6.15.3.2o.

d. The storage and retrieval subservice shall reject any request to start the by-time-range retrieval of packet stores if any of the following conditions occurs:

1. that request contains an instruction that refers to a packet store that does not exist;
2. that subservice does not support concurrent retrieval requests executing in parallel and the packet store open retrieval status of that packet store is in-progress;
3. the retrieval start time in that instruction is later than the retrieval end time;
4. the storage time period of that instruction is fully in the past and the packet store contains no packet with a time stamp in that time period;
5. that request contains an instruction that refers to a packet store whose by-time-range retrieval status is "enabled".

NOTE For item 2, refer to requirement 6.15.3.2i.

e. For each request to start the by-time-range retrieval of packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to start the by-time-range retrieval of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to start the by-time-range retrieval of a packet store, the storage and retrieval subservice shall:

1. set the packet store by-time-range retrieval status of that packet store to "enabled";
2. set the retrieval start time to the start time specified in the request;
3. set the retrieval end time to the end time specified in the request;
4. start the by-time-range retrieval process.

NOTE For item 4, if that subservice supports prioritizing the packet retrievals from packet stores, the retrieval process is performed according the priority policy, refer to requirement 6.15.3.2o.

6.15.3.6.3 Abort the by-time-range retrieval of packet stores

a. The storage and retrieval subservice shall provide the capability to abort the by-time-range retrieval of packet stores if that subservice supports the by-time-range retrieval function.

NOTE 1 The corresponding requests are of message type "TC[15,17] abort the by-time-range retrieval of packet stores".

NOTE 2 For the by-time-range retrieval function support, refer to requirement 6.15.3.6.1a.

NOTE 3 For the capability to start the by-time-range retrieval of packet stores, refer to clause 6.15.3.6.2.

b. Each request to abort the by-time-range retrieval of packet stores shall contain exactly one of:

1. an ordered list of one or more instructions to abort the by-time-range retrieval of a packet store;
2. a single instruction to abort the by-time-range retrieval of all packet stores.

NOTE The instructions to abort the by-time-range retrieval of all packet stores contain no argument.

c. Each instruction to abort the by-time-range retrieval of a packet store shall contain:

1. the identifier of a packet store.

d. The storage and retrieval subservice shall reject any request to abort the by-time-range retrieval of a packet store if:

1. that request contains an instruction that refers to a packet store that does not exist.

e. For each request to abort the by-time-range retrieval of packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to abort the by-time-range retrieval of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to abort the by-time-range retrieval of a packet store, the storage and retrieval subservice shall:

1. finalize the retrieval of the on-going packet, if any;

2. set the packet store by-time-range retrieval status of that packet store to "disabled".

NOTE a packet retrieval transaction implies the retrieval of a single packet from the store and its routing to ground

h. For each valid instruction to abort the by-time-range retrieval of all packet stores, the storage and retrieval subservice shall:

1. for each packet store maintained by that subservice:

(a) finalize the retrieval of the on-going packet, if any;

(b) set its packet store by-time-ranged retrieval status to "disabled".

NOTE a packet retrieval transaction implies the retrieval of a single packet from the store and its routing to ground

6.15.3.7 Report the status of each packet store

a. The storage and retrieval subservice capability to report the status of each packet store shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[15,18] report the status of each packet store". The responses are data reports of message type "TM[15,19] packet store status report".

b. Each request to report the status of each packet store shall contain exactly one instruction to report the status of each packet store.

NOTE The instructions to report the status of each packet store contain no argument.

c. For each valid instruction to report the status of each packet store, the storage and retrieval subservice shall:

1. generate, for each packet store maintained by that subservice, a single packet store status notification that includes:

(a) the packet store identifier;

(b) its packet store storage status;

(c) its packet store open retrieval status;

(d) its packet store by-time-range retrieval status, if the by-time-range retrieval function is supported by that subservice.

NOTE For item 1(d), refer to requirement 6.15.3.6.1a.

d. For each valid request to report the status of each packet store, the storage and retrieval subservice shall generate a single packet store status report that includes the corresponding packet store status notifications.

6.15.3.8 Deleting the packet store contents

6.15.3.8.1 Delete the content of packet stores up to the specified time

a. The storage and retrieval subservice capability to delete the content of packet stores up to the specified time shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[15,11] delete the content of packet stores up to the specified time".

b. Each request to delete the content of packet stores up to the specified time shall contain:

1. the storage time that is the limit of the deletion;
2. exactly one of:
   (a) an ordered list of one or more instructions to delete the content of a packet store up to the specified time, or
   (b) a single instruction to delete the content of all packet stores up to the specified time.

NOTE The instructions to delete the content of all packet stores up to the specified time contain no argument.

c. Each instruction to delete the content of a packet store up to the specified time shall contain:

1. the identifier of the packet store.

d. The storage and retrieval subservice shall reject any request to delete the content of packet stores up to the specified time if any of the following conditions occurs:

1. that request contains an instruction that refers to a packet store that does not exist;
2. that request contains an instruction that refers to a packet store whose packet store by-time-range retrieval status is "enabled";
3. that request contains an instruction that refers to a packet store whose packet store open retrieval status is "in-progress".

e. For each request to delete the content of packet stores up to the specified time that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to delete the content of packet stores up to the specified time that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete the content of a packet store up to the specified time, the storage and retrieval subservice shall:

1. delete the contents of the packet store specified in that instruction, from the earliest packet in that store up to and including the last packet with a time stamp less than or equal to the time specified in that request.

h. For each valid instruction to delete the content of all packet stores up to the specified time, the storage and retrieval subservice shall:

1. for each packet store maintained by that subservice, delete the contents of that packet store from the earliest packet in that store up to and including the last packet with a storage time less than or equal to the time specified in that request.

6.15.3.9 Managing the packet stores

6.15.3.9.1 Create packet stores

a. The storage and retrieval subservice capability to create packet stores shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[15,20] create packet stores".

NOTE 2 For the capability to delete packet stores, refer to clause 6.15.3.9.2.

b. Each request to create packet stores shall contain an ordered list of one or more instructions to create a packet store.

c. Each instruction to create a packet store shall contain:

1. the packet store identifier;
2. the packet store size in bytes;
3. the default open retrieval policy;
4. if more than one packet store type is supported, the packet store type;
5. if more than one packet store virtual channel is supported, the packet store virtual channel.
6. if prioritizing the packet retrievals is supported, the default retrieval priority

NOTE 1 For item 3, refer to requirement 6.15.3.2f.

NOTE 2 For item 4, refer to requirement 6.15.3.2g.

d. The storage and retrieval subservice shall reject any request to create packet stores if any of the following conditions occurs:

1. that request contains an instruction that refers to an already existing packet store;
2. that request contains more than one instruction that refers to the same packet store;
3. the maximum number of packet stores that the subservice supports is already reached;
4. that request contains an instruction that specifies a packet store size that is not compatible with the current memory availability;
5. that request contains an instruction that specifies an invalid default open retrieval policy;
6. that request contains an instruction that specifies an invalid default retrieval priority;

7. that request contains an instruction that specifies an invalid virtual channel.

NOTE 3 For item 1, refer to requirement 6.15.3.2a.

NOTE 4 For item 2, the criteria used to establish whether memory availability is sufficient to allocate the new packet store are not specified in this Standard.

e. For each request to create packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification for that instruction.

f. For each request to create packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to create a packet store, the storage and retrieval subservice shall:

1. create a new packet store with the properties specified in that instruction;
2. set the packet store storage status of the new packet store to "disabled".
3. set the packet store by-time-range retrieval status of the new packet store to "disabled";
4. set the packet store open retrieval status of the new packet store to "suspended"
5. set the default retieval priority with the default retrieval priority specified in that instruction;
6. set the default open retrieval policy with the default open retrieval policy specified in that instruction.

6.15.3.9.2 Delete packet stores

a. The storage and retrieval subservice shall provide the capability to delete packet stores if the capability to create packet stores is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[15,21] delete packet stores".

NOTE 2 For the capability to create packet stores, refer to clause 6.15.3.9.1.

b. Each request to delete packet stores shall contain exactly one of:

1. an ordered list of one or more instructions to delete a packet store;
2. a single instruction to delete all packet stores.

NOTE The instructions to delete all packet stores contain no argument.

c. Each instruction to delete a packet store shall contain:

1. the packet store identifier of the packet store to delete.

d. The storage and retrieval subservice shall reject any request to delete packet stores if any of the following conditions occurs:

1. that request contains an instruction that refers to a packet store that does not exist;
2. that request contains an instruction that refers to a packet store whose packet store storage status is "enabled";
3. that request contains an instruction that refers to a packet store whose packet store by-time-range retrieval status is "enabled";
4. that request contains an instruction that refers to a packet store whose packet store open retrieval status is "in-progress".

e. For each request to delete packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to delete packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a packet store, the storage and retrieval subservice shall:

1. delete the packet store specified in that instruction.

h. For each valid instruction to delete all packet stores, the storage and retrieval subservice shall, for each packet store maintained by that subservice:

1. delete that packet store if it satisfies all of the following conditions:
   (a) its packet store storage status is "disabled";
   (b) its packet store by-time-range retrieval status is "disabled";
   (c) its packet store open retrieval status is "suspended".

6.15.3.9.3 Report the configuration of each packet store

a. The storage and retrieval subservice capability to report the configuration of each packet store shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[15,22] report the configuration of each packet store". The responses are data reports of message type "TM[15,23] packet store configuration report".

NOTE 2 That capability requires the capability for that subservice to create packet stores, refer to clause 6.15.3.9.1.

b. Each request to report the configuration of each packet store shall contain exactly one instruction to report the configuration of each packet store.

NOTE The instructions to report the configuration of each packet store contain no argument.

c. For each valid instruction to report the configuration of each packet store, the storage and retrieval subservice shall:

1. generate, for each managed packet store, a single packet store configuration notification that includes:
   (a) the packet store identifier;
   (b) the packet store size in bytes;
   (c) the default open retrieval policy;

(d) if more than one packet store type is supported, the packet store type (bounded or circular);

(e) if more than one packet store virtual channel is supported, the virtual channel identifier;

(f) if prioritizing packet retrievals is supported, the default retrieval priority.

NOTE 1 For item 1(d), refer to requirements 6.15.3.2f.

NOTE 2 For item 1(e), refer to requirement 6.15.3.2g.

NOTE 3 For item 1(f), refer to requirement 6.15.3.2g.

d. For each valid request to report the configuration of each packet store, the storage and retrieval subservice shall generate a single packet store configuration report that includes all related packet store configuration notifications.

6.15.3.9.4 Copy the packets contained in a packet store selected by time window

a. The storage and retrieval subservice capability to copy the packets contained in a packet store selected by time window shall be declared when specifying that subservice.

NOTE 4 The corresponding requests are of message type "TC[15,24] copy the packets contained in a packet store selected by time window".

NOTE 5 That capability requires the capability for that subservice to create packet stores, refer to clause 6.15.3.9.1.

b. Each request to copy the packets contained in a packet store selected by time window shall contain exactly one instruction to copy the packets contained in a packet store selected by time window.

c. Each instruction to copy the packets contained in a packet store selected by time window shall contain:

1. the time window;
2. the source packet store identifier;
3. the destination packet store identifier.

d. The time window filtering function shall support the following mechanisms:

1. "select all packets stored from time tag to time tag";
2. "select all packets stored after time tag";
3. "select all packets stored before time tag".

e. The set of packets identified by the "select all packets stored from time tag to time tag" filtering mechanism shall be all packets that are stored between and including the specified "from time tag" and "to time tag".

f. The set of packets identified by the "select all packets stored after time tag" filtering mechanism shall be all packets that are stored at and after that specified "from time tag".

g. The set of packets identified by the "select all packets stored before time tag" filtering mechanism shall be all packets that are scheduled before and at that specified "to time tag".

h. The storage and retrieval subservice shall queue any request to copy the packets contained in a packet store selected by time window if another request to copy packets in that packet store is already executing.

i. The storage and retrieval subservice shall reject any request to copy the packets contained in a packet store selected by time window if any of the following conditions occurs:

1. that request contains an instruction that refers to an unknown source packet store;
2. that request contains an instruction that refers to an unknown destination packet store;
3. that request contains an instruction that contains an invalid time window;
4. that request contains an instruction that refers to a destination packet store whose packet store storage status is "enabled".

NOTE For bullet 4. this Standard does not require the destination packet store to be empty. It also does not specify how to merge the packets, those already present in the packet store and those that are copied. For merging, refer also to the storage time-stamping method used in clause 6.15.3.3.

j. For each request to copy the packets contained in a packet store selected by time window that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

k. For each valid instruction to copy the packets contained in a packet store selected by time window, the storage and retrieval subservice shall:

1. copy all packets from the source packet store that are in the specified time window to the destination packet store.

6.15.3.9.5 Abort all requests to copy the packets contained in a packet store selected by time window

a. The storage and retrieval subservice capability to abort all requests to copy the packets contained in a packet store selected by time window shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[15,10] abort all requests to copy the packets contained in a packet store selected by time window".

b. Each request to abort all requests to copy the packets contained in a packet store selected by time window shall contain exactly one instruction to abort all requests to copy the packets contained in a packet store selected by time window.

NOTE The instructions to abort all requests to copy the packets contained in a packet store selected by time window contain no argument.

c. The storage and retrieval subservice shall reject any request to abort all requests to copy the packets contained in a packet store selected by time window in case of no on-going copy operation

d. For each request to abort all requests to copy the packets contained in a packet store selected by time window that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

e. For each valid instruction to abort all requests to copy the packets contained in a packet store selected by time window, the storage and retrieval subservice shall:

1. abort all copy operations of packets contained in a packet store selected by time window
2. ensure that each copy operation abort respects packet boundaries

NOTE a copy operation will stop once the on-going packet copy is completed

6.15.3.10 Changing packet store properties

6.15.3.10.1 Resize packet stores

a. The storage and retrieval subservice capability to resize packet stores shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[15,25] resize packet stores".

b. Each request to resize packet stores shall contain an ordered list of one or more instructions to resize a packet store.

c. Each instruction to resize a packet store shall contain:

1. the packet store identifier of the packet store to resize;
2. the new packet store size in bytes.

d. The storage and retrieval subservice shall reject any request to resize packet stores if any of the following conditions occurs:

1. that request contains an instruction that refers to a packet store that does not exist;
2. that request contains an instruction that specifies a packet store size that is not compatible with the current memory availability;
3. that request contains an instruction that refers to a packet store whose packet store storage status is "enabled";
4. that request contains an instruction that refers to a packet store whose packet store open retrieval status is "in-progress";
5. that request contains an instruction that refers to a packet store that packet store by-time-range retrieval status is "enabled".

e. For each request to resize packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to resize packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to resize a packet store, the storage and retrieval subservice shall:

1. change the size of the packet store to the size specified in that instruction.

NOTE The time and conditions needed for this request to take effect is implementation-dependent.

6.15.3.10.2 Change a packet store type to circular

a. The storage and retrieval subservice shall provide the capability to change the type of a packet store to circular if the capability to manage circular packet stores is provided by that subservice.

NOTE The corresponding requests are of message type "TC[15,26] change a packet store type to circular".

b. Each request to change a packet store type to circular shall contain exactly one instruction to change a packet store type to circular.

c. Each instruction to change a packet store type to circular shall contain:

1. the packet store identifier of the packet store whose type is changed.

d. The storage and retrieval subservice shall reject any request to change a packet store type to circular if any of the following conditions occurs:

1. that request contains an instruction that refers to a packet store that does not exist;
2. that request contains an instruction that refers to a packet store whose packet store storage status is "enabled";
3. that request contains an instruction that refers to a packet store whose packet store by-time-range retrieval status is "enabled";
4. that request contains an instruction that refers to a packet store whose packet store open retrieval status is "in-progress".

e. For each request to change a packet store type to circular that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each valid instruction to change a packet store type to circular, the storage and retrieval subservice shall:

1. change the packet store type of the packet store specified in that instruction to "circular".

NOTE This Standard does not elaborate on how the content of the packet store is managed when its type is changed.

6.15.3.10.3 Change a packet store type to bounded

a. The storage and retrieval subservice shall provide the capability to change the type of a packet store to bounded if the capability to manage bounded packet stores is provided by that subservice.

NOTE The corresponding requests are of message type "TC[15,27] change a packet store type to bounded".

b. Each request to change a packet store type to bounded shall contain exactly one instruction to change a packet store type to bounded.

c. Each instruction to change a packet store type to bounded shall contain:
   1. the packet store identifier of the packet store whose type is to change.

d. The storage and retrieval subservice shall reject any request to change a packet store type to bounded if any of the following conditions occurs:
   1. that request contains an instruction that refers to a packet store that does not exist;
   2. that request contains an instruction that refers to a packet store whose packet store storage status is "enabled";
   3. that request contains an instruction that refers to a packet store whose packet store by-time-range retrieval status is "enabled";
   4. that request contains an instruction that refers to a packet store whose packet store open retrieval status is "in-progress".

e. For each request to change a packet store type to bounded that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each valid instruction to change a packet store type to bounded, the storage and retrieval subservice shall:
   1. change the packet store type of the packet store specified in that instruction to "bounded".

      NOTE This Standard does not elaborate on how the content of the packet store is managed when its type is changed.

6.15.3.10.4 Change the virtual channel used by a packet store

a. The storage and retrieval subservice shall provide the capability to change the virtual channel of a packet store if the list of virtual channels that can be used by the subservice contains more than one virtual channel.

      NOTE The corresponding requests are of message type "TC[15,28] change the virtual channel used by a packet store".

b. Each request to change the virtual channel used by a packet store shall contain exactly one instruction to change the virtual channel used by a packet store.

c. Each instruction to change the virtual channel used by a packet store shall contain:
   1. the packet store identifier of the packet store whose virtual channel is to change.
   2. the identifier of the new virtual channel for that packet store.

d. The storage and retrieval subservice shall reject any request to change the virtual channel used by a packet store if any of the following conditions occurs:
   1. that request contains an instruction that refers to a packet store that does not exist;
   2. that request contains an instruction that refers to a virtual channel that is not valid;

3. that request contains an instruction that refers to a packet store whose packet store by-time-range retrieval status is "enabled";
4. that request contains an instruction that refers to a packet store whose packet store open retrieval status is "in-progress".

e. For each request to change the virtual channel used by a packet store that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each valid instruction to change the virtual channel used by a packet store, the storage and retrieval subservice shall:

1. change the virtual channel of the packet store specified in that instruction to the specified virtual channel.

6.15.3.10.5 Change the default retrieval priority of a packet store

a. The storage and retrieval subservice shall provide the capability to change the default retrieval priority of a packet store if the capability to prioritize packet store retrievals is provided by that subservice.

NOTE The corresponding requests are of message type "TC[15,41] change the default retrieval priority of a packet store ".

b. Each request to change the default retrieval priority of a packet store shall contain exactly one instruction to change the default retrieval priority of a packet store.

c. Each instruction to change the default retrieval priority of a packet store shall contain:

1. the identifier of the packet store whose default retrieval priority is to be changed.
2. the new default retrieval priority.

d. The storage and retrieval subservice shall reject any request to change the default retrieval priority if any of the following conditions occurs:

1. that request contains an instruction that refers to a packet store that does not exist;
2. that request contains an instruction that refers to a default retrieval priority that is not valid;
3. that request contains an instruction that refers to a packet store whose packet store by-time-range retrieval status is "enabled";
4. that request contains an instruction that refers to a packet store whose packet store open retrieval status is "in-progress".

e. For each request to change the default retrieval priority of a packet store that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each valid instruction to change the default retrieval priority of a packet store, the storage and retrieval subservice shall:

1. change the default retrieval priority of the packet store specified in that instruction with the specified default retrieval priority.

6.15.3.11 Reporting the content of the packet stores

6.15.3.11.1 Summary-report the content of packet stores

a. The storage and retrieval subservice capability to summary-report the content of packet stores shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[15,12] summary-report the content of packet stores". The responses are data reports of message type "TM[15,13] packet store content summary report".

b. Each request to summary-report the content of packet stores shall contain exactly one of:

1. an ordered list of one or more instructions to summary-report the content of a packet store;
2. a single instruction to summary-report the content of each packet store.

NOTE The instructions to summary-report the content of each packet store contain no argument.

c. Each instruction to summary-report the content of a packet store shall contain:

1. the packet store identifier of the packet store to report.

d. The storage and retrieval subservice shall reject any request to summary-report the content of a packet store if:

1. that request contains an instruction that refers to a packet store that does not exist.

e. For each request to summary-report the content of packet stores that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each request to summary-report the content of packet stores that contains only valid instructions, the storage and retrieval subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to summary-report the content of a packet store, the storage and retrieval subservice shall generate a single packet store content summary notification that includes:

1. the packet store identifier;
2. the storage time tag of the oldest packet in the packet store:
3. the storage time tag of the packet information for the newest packet in the packet store;
4. the current start time for open retrieval;
5. the packet store filling percentage;
6. the packet store filling percentage for packets between the current open retrieval start time tag and the newest packet in that store.

NOTE For item 6, this value gives the amount of data still to transfer to ground.

h. For each valid instruction to summary-report the content of each packet store, the storage and retrieval subservice shall generate, for each packet store maintained by that subservice, a single packet store content summary notification.

NOTE The content of each packet store content summary notification is as defined in requirement 6.15.3.11.1g.

i. For each valid request to summary-report the content of packet stores, the storage and retrieval subservice shall generate a single packet store content summary report that contains all related packet store content summary notifications.

6.15.3.12 Subservice observables

a. The following observables shall be defined for the storage and retrieval subservice:

1. the number of retrieval requests queued pending their execution;
2. for each packet store,
   (a) the packet store open retrieval status;
   (b) the packet store by-time-range retrieval status, if the by-time-range retrieval function is supported by that subservice;
   (c) the current packet store open retrieval start time tag;
   (d) the packet store filling percentage;
   (e) the storage time of the last packet stored;
   (f) the packet store filling percentage for packets between the packet store open retrieval start time tag and the newest packet in that store;
   (g) the storage time of the oldest packet present in the packet store;
   (h) the packet store storage status;
3. if the subservice provides the capability to copy the packets contained in a packet store selected by time window,
   (a) the source packet store identifier and the destination packet store identifier of the on-going, if any, copy operation;
   (b) the number of on-going and queued copy requests.

NOTE 1 For item 1, refer to requirement 6.15.3.2k.

NOTE 2 For item 3, refer to requirement 6.15.3.9.4.

6.15.4 Packet selection subservice

6.15.4.1 Accessibility

6.15.4.1.1 Application process

a. The list of application processes that are controlled by the packet selection subservice shall be declared when specifying that subservice.

NOTE The packet selection subservice always controls the storage of reports generated by the application process that hosts that subservice.

b. The packet selection subservice shall be able to handle, at any time, all reports that are generated by each application process that is controlled by that subservice.

6.15.4.1.2 Packet store

a. The packet selection subservice shall, at any time, have access to the packet stores maintained by the storage and retrieval subservice of the parent on-board storage and retrieval service.

6.15.4.2 Storage-control definitions

6.15.4.2.1 Capability

a. Whether the packet selection subservice provides the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports shall be declared when specifying that subservice.

NOTE 1 See clause 6.15.4.2.3.

NOTE 2 For the housekeeping parameter reports, refer to clause 6.3.3.4.

b. Whether the packet selection subservice provides the capability to control, per event definition, the storage of event reports shall be declared when specifying that subservice.

NOTE 1 See clause 6.15.4.2.4.

NOTE 2 For the event reports, refer to clause 6.5.4.

c. If the packet selection subservice provides the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports, the subservice capability to subsample the storage of the parameter reports shall be declared when specifying that subservice.

NOTE Refer to requirement 6.15.4.2.1a.

6.15.4.2.2 Application process storage-control configuration

a. For each packet store, the maximum number of application process storage-control definitions that the packet selection subservice can contemporaneously control shall, at any time, correspond to the number of application processes that are controlled by that subservice.

NOTE 1 See requirement 6.15.4.1.1a.

NOTE 2 The application process storage control configuration for a packet store contains the application process storage control definitions that the packet selection subservice maintains for that packet store.

b. Each application process storage-control definition shall contain:

1. the packet store identifier;
2. the identifier of the application process to control;
3. a list of zero or more application process related "service type storage-control definitions", each one containing:
   (a) the identifier of the service type to control;
   (b) a list of zero or more application process and service type related "report type storage-control definitions", each one containing the message subtype identifier of a report type.

NOTE 1 The packet selection subservice has knowledge about the application processes that it controls but no knowledge about the service types and report types that they can generate. This lack of knowledge results in the possibility for the subservice to handle on-board, service type storage-control definitions or report type storage-control definitions that can be meaningless. It is of ground operations responsibility to ensure consistency in this respect.

NOTE 2 An empty application process storage-control configuration (i.e. no application process storage-control definition is defined) implies that the subservice blocks all reports. Blocking means that these reports are not stored in the corresponding packet store.

NOTE 3 If the subservice provides none of the capabilities specified in requirements 6.15.4.2.1a, 1.1.1.1.1a and 6.15.4.2.1b, a report is stored only if it fulfils one of the following conditions:

- an application process storage-control definition with no service type storage-control definition is defined for the application process identifier of that report;
- an application process storage-control definition with a service type storage-control definition that has no report type storage-control definition is defined for the application process identifier and the service type of that report;
- an application process storage-control definition with a service type storage-control definition is defined that has a report type storage-control definition for the application process identifier and the service type and the message subtype identifier of that report.

c. The maximum number of service type storage control definitions that can be contained within an application process storage control definition shall be declared when specifying the packet selection subservice.

d. The maximum number of report type storage control definitions that can be contained within a service type storage control definition shall be declared when specifying the packet selection subservice.

6.15.4.2.3 Housekeeping parameter report storage-control configuration

a. For each packet store, the maximum number of housekeeping parameter report storage-control definitions that the packet selection subservice can contemporaneously control shall, at any time, correspond to the number of application processes that are controlled by that subservice and that provide the capability for generating housekeeping parameter reports.

NOTE 1 For the number of application processes, see requirement 6.15.4.1.1a.

NOTE 2 The housekeeping parameter report storage configuration for a packet store contains the housekeeping parameter report storage definitions that the packet selection subservice maintains for that packet store.

NOTE 3 The housekeeping parameter report storage-control configuration contains the housekeeping parameter report storage-control definitions of the packet selection subservice.

b. Each housekeeping parameter report storage-control definition shall contain:

1. the packet store identifier;
2. the identifier of the application process;
3. a list of zero or more related housekeeping parameter report structure identifiers.

NOTE 1 An empty housekeeping parameter report storage-control configuration (i.e. no housekeeping parameter report storage-control definition is defined) implies that the subservice blocks all housekeeping parameter reports. Blocking means that these reports are not stored in the corresponding packet store.

NOTE 2 A housekeeping parameter report is stored in the corresponding packet store only if the application process storage-control configuration does not block that report and one of the following conditions occurs:

- a housekeeping parameter report storage-control definition with no housekeeping parameter report structure identifiers is defined for the application process identifier of that report;

- a housekeeping parameter report storage-control definition with a housekeeping parameter report structure identifier is defined for the application process identifier and the housekeeping parameter report structure identifier of that report.

c. The maximum number of housekeeping parameter report structure identifiers that can be contained within a housekeeping parameter report storage-control definition shall be declared when specifying the packet selection subservice.

6.15.4.2.4 Event report blocking storage-control configuration

a. For each packet store, the maximum number of event report blocking storage-control definitions that the packet selection subservice can contemporaneously control shall, at any time, correspond to the number of application processes that are controlled by that subservice and that provide the capability for generating event reports.

NOTE 1 For the number of application processes, see requirement 6.15.4.1.1a.

NOTE 2 The event report blocking storage-control configuration contains the event report blocking storage-control definitions of the packet selection subservice.

b. Each event report blocking storage-control definition shall contain:

1. the packet store identifier;
2. the identifier of the application process;
3. a list of zero or more related event definition identifiers.

NOTE 1 An empty event report blocking storage-control configuration (i.e. no event report blocking storage-control definition is defined) implies that an event report is stored in the corresponding packet store if the application process storage-control configuration does not block that report.

NOTE 2 The packet selection subservice blocks the storage of an event report in the corresponding packet store if any of the following conditions occurs:

- the application process storage-control configuration blocks that report;
- the application process storage-control configuration does not block that report and an event report blocking storage-control definition with no event definition identifiers is defined for the application process identifier of that report;
- the application process storage-control configuration does not block that report and an event report blocking storage-control definition with an event definition identifier is defined for

the application process identifier and the event definition identifier of that report.

c. The maximum number of event definition identifiers that can be contained within an event report blocking storage-control definition shall be declared when specifying the packet selection subservice.

6.15.4.3 Storage control processing logic

a. The packet selection subservice shall block the storage of a report to a packet store if the application process identifier of that report is not contained within an application process storage definition for that packet store.

b. The packet selection subservice shall block the storage of a report to a packet store if that report fulfils all of the following conditions:
   1. the application process identifier of that report is contained within an application process storage definition for that packet store, and
   2. that application process storage definition contains at least one service type storage definition, and
   3. that application process storage definition does not contain a service type storage definition for the service type of that report.

c. The packet selection subservice shall block the storage of a report to a packet store if that report fulfils all of the following conditions:
   1. the application process identifier of that report is contained within an application process storage definition for that packet store, and
   2. that application process storage definition contains a service type storage definition for the service type of that report, and
   3. that service type storage definition contains at least one report type storage definition, and
   4. that service type storage definition does not contain a report type storage definition for the report type of that report.

d. If the packet selection subservice provides the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports, the subservice shall block the storage of a housekeeping parameter report to a packet store if the application process identifier of that report is not contained within a housekeeping parameter report storage definition for that packet store,

e. If the packet selection subservice provides the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports, the subservice shall block the storage of a housekeeping parameter report to a packet store if that report fulfils all of the following conditions:
   1. the application process identifier of that report is contained within a housekeeping parameter report storage definition for that packet store, and
   2. that housekeeping parameter report storage definition contains at least one housekeeping parameter report structure identifier, and

3. that housekeeping parameter report storage definition does not contain the housekeeping parameter report structure identifier of that report.

NOTE The storage of a housekeeping parameter report structure of an application process is enabled if it is blocked neither by the application process storage control configuration nor by the housekeeping parameter report storage configuration.

f. If the packet selection subservice provides the capability to control, per event definition, the storage of event reports, the subservice shall block the storage of an event report to a packet store if that report fulfils all of the following conditions:

1. the application process identifier of that report is contained within an event report blocking storage-control definition for that packet store, and
2. that event report blocking storage-control definition has no event definition identifier.

g. If the packet selection subservice provides the capability to control, per event definition, the storage of event reports, the subservice shall block the storage of an event report to a packet store if that report fulfils all of the following conditions:

1. the application process identifier of that report is contained within an event report blocking storage-control definition for that packet store, and
2. that event report blocking storage-control definition contains the event definition identifier of that report.

NOTE The storage of an event definition of an application process is enabled if it is blocked neither by the application process storage control configuration nor by the event report blocking control configuration.

6.15.4.4 Managing the application process storage-control configuration

6.15.4.4.1 Add report types to the application process storage-control configuration

a. The packet selection subservice shall provide the capability to add report types to the application process storage-control configuration of a packet store.

NOTE 1 The corresponding requests are of message type "TC[15,3] add report types to the application process storage-control configuration".

NOTE 2 For the capability to delete report types from the application process storage-control configuration, refer to clause 6.15.4.4.2.

b. Each request to add report types to the application process storage-control configuration shall contain:

1. the packet store identifier of the packet store whose application process storage-control configuration is to change;
2. exactly one of:
   (a) an ordered list of one or more instructions to add a report type to the application process storage-control configuration,
   (b) an ordered list of one or more instructions to add all report types of a service type to the application process storage-control configuration,
   (c) an ordered list of one or more instructions to add all report types of an application process to the application process storage-control configuration.

c. Each instruction to add a report type to the application process storage-control configuration shall contain:
   1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction;
   2. the report type identifier consisting of:
      (a) the service type identifier;
      (b) the message subtype identifier.

d. Each instruction to add all report types of a service type to the application process storage-control configuration shall contain:
   1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction;
   2. the service type identifier.

e. Each instruction to add all report types of an application process to the application process storage-control configuration shall contain:
   1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction.

f. The packet selection subservice shall reject any request to add report types to the application process storage-control configuration if any of the following conditions occurs:
   1. that request refers to a packet store that does not exist;
   2. that request contains an instruction that refers to an application process that is not controlled by that subservice;
   3. that request contains an instruction that implies the addition of a service type storage definition and the maximum number of service type storage definitions for the corresponding application process storage definition is already reached;
   4. the maximum number of report type storage-control definitions that can be contained within the corresponding service type storage-control definition is already reached;
   5. the corresponding service type storage-control definition has no report type storage-control definition already defined;

6. the corresponding application process storage-control definition has no service type storage-control definition already defined;
7. that request contains an instruction that refers to an application process that is not controlled by that subservice;
8. that request contains an instruction that implies the addition of a service type storage definition and the maximum number of service type storage definitions for the corresponding application process storage definition is already reached;
9. the corresponding application process storage-control definition has no service type storage-control definition already defined;
10. that request contains an instruction that refers to an application process that is not controlled by that subservice.

g. For each request to add report types to the application process storage-control configuration that is rejected, the packet selection subservice shall generate a failed start of execution notification.

h. The packet selection subservice shall reject any instruction contained within a request to add report types to the application process storage-control configuration if:
1. that instruction refers to an application process that is not controlled by that subservice.

i. For each valid instruction to add a report type to the application process storage-control configuration, the packet selection subservice shall:
1. add, for the specified application process identifier, an application process storage-control definition if not already existing;
2. add, for the related application process storage-control definition and the specified service type identifier, a service type storage-control definition, if not already existing;
3. add, for the related service type storage-control definition and the specified message subtype identifier, a report type storage-control definition, if not already existing.

j. For each valid instruction to add all report types of a service type to the application process storage-control configuration, the packet selection subservice shall:
1. add, for the specified application process identifier, an application process storage-control definition if not already existing;
2. add, for the related application process storage-control definition and the specified service type identifier, a service type storage-control definition to the related application process storage-control definition, if not already existing;
3. delete, if any, all report type storage-control definitions of the related service type storage-control definition.

k. For each valid instruction to add all report types of an application process to the application process storage-control configuration, the packet selection subservice shall:
1. add, for the specified application process identifier, an application process storage-control definition if not already existing;

2. delete, if any, all service type storage-control definitions of the related application process storage-control definition.

6.15.4.4.2 Delete report types from the application process storage-control configuration

a. The packet selection subservice shall provide the capability to delete report types from the application process storage-control configuration of a packet store.

NOTE 1 The corresponding requests are of message type "TC[15,4] delete report types from the application process storage-control configuration of a packet store".

NOTE 2 For the capability to add report types to the application process storage-control configuration, refer to clause 6.15.4.4.1.

b. Each request to delete report types from the application process storage-control configuration shall contain the packet store identifier of the packet store whose application process storage-control configuration is to change and exactly one of:

1. at least one of:
   (a) one or more instructions to delete a report type from the application process storage-control configuration,
   (b) one or more instructions to delete a service type from the application process storage-control configuration,
   (c) if the packet selection subservice controls more than one application process, one or more instructions to delete an application process from the application process storage-control configuration,
2. a single instruction to empty the application process storage-control configuration.

NOTE The instructions to empty the application process storage-control configuration contain no argument.

c. Each instruction to delete a report type from the application process storage-control configuration shall contain:

1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction,
2. the report type identifier consisting of:
   (a) the service type identifier;
   (b) the message subtype identifier.

NOTE For item 1, refer to requirement 6.15.4.1.1a.

d. Each instruction to delete a service type from the application process storage-control configuration shall contain:

1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction;

2. the service type identifier.

e. Each instruction to delete an application process from the application process storage-control configuration shall contain:

1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction.

f. The packet selection subservice shall reject any request to delete report types from the application process storage-control configuration if any of the folowing conditions occurs:

1. that request refers to a packet store than does not exist;

2. that request contains an instruction that refers to a report type identifier that is not in the application process storage-control configuration;

3. that request contains an instruction that refers to a service type identifier that is not in the application process storage-control configuration;

4. that request contains an instruction that refers to an application process identifier that is not in the application process storage-control configuration

g. For each request to delete report types from the application process storage-control configuration of a packet store that is rejected, the packet selection subservice shall generate a failed start of execution notification.

h. For each valid instruction to delete a report type from the application process storage-control configuration, the packet selection subservice shall:

1. delete the report type storage-control definition related to that specified application process identifier, service type identifier and message subtype identifier;

2. if that report type storage-control definition deletion results in an emptied service type storage-control definition, delete that service type storage-control definition;

3. if that service type storage-control definition deletion results in an emptied application process storage-control definition, delete that application process storage-control definition.

i. For each valid instruction to delete a service type from the application process storage-control configuration, the packet selection subservice shall, for the related packet store:

1. delete the service type storage-control definitions related to that specified application process identifier and service type identifier;

2. if that service type storage-control definition deletion results in an emptied application process storage-control definition, delete that application process storage-control definition.

j. For each valid instruction to delete an application process from the application process storage-control configuration, the packet selection subservice shall, for the related packet store:

1. delete the application process storage-control definition related to that specified application process identifier.

k. For each valid instruction to empty the application process storage-control configuration, the packet selection subservice shall, for the related packet store:

1. delete, if any, all application process storage-control definitions.

6.15.4.4.3 Report the content of the application process storage-control configuration

a. The packet selection subservice capability to report the content of the application process storage-control configuration of a packet store shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[15,5] report the content of the application process storage-control configuration". The responses are data reports of message type "TM[15,6] application process storage-control configuration content report".

NOTE 2 That capability requires the capability for that subservice to add report types to the application process storage-control configuration, refer to clause 6.15.4.4.1.

b. Each request to report the content of the application process storage-control configuration shall contain exactly one instruction to report the content of the application process storage-control configuration.

c. Each instruction to report the content of the application process storage-control configuration shall contain:

1. the packet store identifier of the packet store.

d. The packet selection subservice shall reject any request to report the content of the application process storage-control configuration if:

1. that request contains an instruction that refers to a packet store that does not exist.

e. For each instruction to report the content of the application process storage-control configuration that is rejected, the packet selection subservice shall generate a failed start of execution notification.

f. For each valid instruction to report the content of the application process storage-control configuration, the packet selection subservice shall generate, for each existing application process storage-control definition of the related packet store, a single application process storage-control definition notification that includes:

1. if the packet selection subservice controls more than one application process, the related application process identifier;
2. for each related service type storage-control definition, if any:
   (a) the related service type identifier;
   (b) for each related report type storage-control definition, if any, the related message subtype identifier.

NOTE For item 1, refer to requirement 6.15.4.1.1a.

g. For each valid request to report the content of the application process storage-control configuration, the packet selection subservice shall generate a single application process storage-control configuration content report that includes:

1. the packet store identifier of the related packet store;
2. all related application process storage-control definition notifications.

6.15.4.5 Managing the housekeeping parameter report storage-control configuration

6.15.4.5.1 Add structure identifiers to the housekeeping parameter report storage-control configuration

a. The packet selection subservice shall provide the capability to add structure identifiers to the housekeeping parameter report storage-control configuration of a packet store if that subservice provides the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports.

NOTE 1 The corresponding requests are of message type "TC[15,29] add structure identifiers to the housekeeping parameter report storage-control configuration".

NOTE 2 For the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports, refer to requirement 6.15.4.2.1a.

NOTE 3 For the capability to delete structure identifiers from the housekeeping parameter report storage-control configuration, refer to clause 6.15.4.5.2.

b. Each request to add structure identifiers to the housekeeping parameter report storage-control configuration shall contain:

1. the packet store identifier of the packet store whose housekeeping parameter report storage-control configuration is to change;
2. exactly one of:
   (a) an ordered list of one or more instructions to add a structure identifier to the housekeeping parameter report storage-control configuration,
   (b) exactly one instruction to add all structure identifiers to the housekeeping parameter report storage-control configuration.

NOTE The instruction to add all structure identifiers to the housekeeping parameter report storage-control configuration contain no argument

c. Each instruction to add a structure identifier to the housekeeping parameter report storage-control configuration shall contain:

1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction;
2. the housekeeping parameter report structure identifier;
3. if subsampling is supported, the subsampling rate.

d. Each instruction to add all structure identifiers to the housekeeping parameter report storage-control configuration shall contain:
   1. the application process identifier addressed by that instruction.

e. The packet selection subservice shall reject any request to add structure identifiers to the housekeeping parameter report storage-control configuration if any of the following conditions occurs:
   1. that request refers to a packet store that does not exist;
   2. that request contains an instruction that refers to an application process that is not controlled by that subservice;
   3. the maximum number of housekeeping parameter report structure identifiers that can be contained within a housekeeping parameter report storage-control definition is already reached;
   4. the corresponding housekeeping parameter report storage-control definition has no structure identifier already defined;
   5. that request contains an instruction that refers to an application process that is not controlled by that subservice.

f. For each request to add structure identifiers to the housekeeping parameter report storage-control configuration that is rejected, the packet selection subservice shall generate a failed start of execution notification.

g. The packet selection subservice shall reject any instruction contained within a request to add structure identifiers to the housekeeping parameter report storage-control configuration if:
   1. that instruction refers to an application process that is not controlled by that subservice.

h. For each valid instruction to add a structure identifier to the housekeeping parameter report storage-control configuration, the packet selection subservice shall:
   1. add, for the specified application process identifier, a housekeeping parameter report storage-control definition if not already existing;
   2. add, to the related housekeeping parameter report storage-control definition, the specified housekeeping parameter report structure identifier, if not already existing;
   3. if subsampling is supported, set, to the related housekeeping parameter report storage-control definition

and the specified housekeeping parameter report structure identifier, the specified subsampling rate.

i. For each valid instruction to add all structure identifiers to the housekeeping parameter report storage-control configuration, the packet selection subservice shall:

1. add, for the specified application process identifier, a housekeeping parameter report storage-control definition if not already existing;
2. delete, if any, all housekeeping parameter report structure identifiers of the related housekeeping parameter report storage-control definition.

6.15.4.5.2 Delete structure identifiers from the housekeeping parameter report storage-control configuration

a. The packet selection subservice shall provide the capability to delete structure identifiers from the housekeeping parameter report storage-control configuration of a packet store if that subservice provides the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports.

NOTE 1 The corresponding requests are of message type "TC[15,30] delete structure identifiers from the housekeeping parameter report storage-control configuration".

NOTE 2 For the capability to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports, refer to requirement 6.15.4.2.1a.

NOTE 3 For the capability to add structure identifiers to the housekeeping parameter report storage-control configuration, refer to clause 6.15.4.5.1.

b. Each request to delete structure identifiers from the housekeeping parameter report storage-control configuration shall contain the packet store identifier of the packet store whose housekeeping parameter report storage-control configuration is to change and exactly one of:

1. any combination of ordered lists of one or more instructions:
   (a) to delete a structure identifier from the housekeeping parameter report storage-control configuration,
   (b) to delete an application process from the housekeeping parameter report storage-control configuration;
2. exactly one instruction to empty the housekeeping parameter report storage-control configuration.

NOTE The instructions to empty the housekeeping parameter report storage-control configuration contain no argument.

c. Each instruction to delete a structure identifier from the housekeeping parameter report storage-control configuration shall contain:

1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction;
2. the housekeeping parameter report structure identifier.

NOTE For item 1, refer to requirement 6.15.4.1.1a.

d. Each instruction to delete an application process from the housekeeping parameter report storage-control configuration shall contain:
   1. the application process identifier addressed by that instruction.

e. The packet selection subservice shall reject any request to delete structure identifiers from the housekeeping parameter report storage-control configuration if any of the following conditions occurs:
   1. that request contains an instruction that refers to an application process identifier that is not in the housekeeping parameter report storage configuration of the related packet store;
   2. that request contains an instruction that refers to a housekeeping parameter report structure identifier that is not in the housekeeping parameter report storage definition for the specified application process identifier;
   3. that request contains an instruction that refers to an application process identifier that is not in the housekeeping parameter report storage configuration of the related packet store.

f. For each request to delete structure identifiers from the housekeeping parameter report storage-control configuration that is rejected, the packet selection subservice shall generate a failed start of execution notification.

g. For each request to delete structure identifiers from the housekeeping parameter report storage-control configuration that contains only valid instructions, the packet selection subservice shall execute those instructions in the order of their appearance in that request

h. For each valid instruction to delete a structure identifier from the housekeeping parameter report storage-control configuration , the packet selection subservice shall, for the related packet store:
   1. delete the housekeeping parameter report structure identifier related to the specified application process identifier;
   2. if that housekeeping parameter report structure identifier deletion results in an emptied housekeeping parameter report storage-control definition, delete that housekeeping parameter report storage-control definition.

      NOTE Deleting a housekeeping parameter report structure identifier implies deleting the corresponding subsampling rate if any (see also requirement 6.15.4.2.1c).

i. For each valid instruction to delete an application process from the housekeeping parameter report storage-control configuration, the packet selection subservice shall, for the related packet store:
   1. delete the housekeeping parameter report storage definition that is defined for that specified application process identifier.

j. For each valid instruction to empty the housekeeping parameter report storage-control configuration, the packet selection subservice shall, for the related packet store:

1. delete all housekeeping parameter report storage definitions.

6.15.4.5.3 Report the content of the housekeeping parameter report storage-control configuration

a. The packet selection subservice capability to report the content of the housekeeping parameter report storage-control configuration of a packet store shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[15,35] report the content of the housekeeping parameter report storage-control configuration". The responses are data reports of message type "TM[15,36] housekeeping parameter report storage-control configuration content report".

NOTE 2 That capability requires the capability for that subservice to control, per housekeeping parameter report structure, the storage of housekeeping parameter reports (refer to requirement 6.15.4.2.1a).

b. Each request to report the content of the housekeeping parameter report storage-control configuration shall contain exactly one instruction to report the content of the housekeeping parameter report storage-control configuration.

c. Each instruction to report the content of the housekeeping parameter report storage configuration shall include:

1. the packet store identifier of the packet store.

d. The packet selection subservice shall reject any request to report the content of the housekeeping parameter report storage configuration if:

1. that request contains an instruction that refers to a packet store that does not exist.

e. For each valid instruction to report the content of the housekeeping parameter report storage-control configuration, the packet selection subservice shall generate, for each existing housekeeping parameter report storage-control definition of the related packet store, a single housekeeping parameter report storage-control definition notification that includes:

1. if the packet selection subservice controls more than one application process, the related application process identifier;
2. for each housekeeping parameter report structure identifier entry:
   (a) the housekeeping parameter report structure identifier;
   (b) if subsampling is supported, the subsampling rate.

NOTE 1 For item 1, refer to requirement 6.15.4.1.1a.

NOTE 2 For item 2(b), refer to requirement 6.15.4.2.1c.

f. For each valid request to report the content of the housekeeping parameter report storage-control configuration, the packet selection subservice shall

generate a single housekeeping parameter report storage-control configuration content report that includes:

1. the packet store identifier of the related packet store;
2. all related housekeeping parameter report storage-control definition notifications.

6.15.4.6 Managing the event report blocking storage-control configuration

6.15.4.6.1 Add event definition identifiers to the event report blocking storage-control configuration

a. The packet selection subservice shall provide the capability to add event definition identifiers to the event report blocking storage-control configuration of a packet store if that subservice provides the capability to control, per event definition, the storage of event reports.

NOTE 1 The corresponding requests are of message type "TC[15,34] add event definition identifiers to the event report blocking storage-control configuration".

NOTE 2 For the capability to control, per event definition, the storage of event reports, refer to requirement 6.15.4.2.1b.

NOTE 3 For the capability to delete event definition identifiers from the event report blocking storage-control configuration, refer to clause 6.15.4.6.2.

b. Each request to add event definition identifiers to the event report blocking storage-control configuration shall contain:

1. the packet store identifier of the packet store whose event report blocking storage-control configuration is to change;
2. exactly one of:
   (a) an ordered list of one or more instructions to add an event definition identifier to the event report blocking storage-control configuration,
   (b) a single instruction to add all event definition identifiers to the event report blocking storage-control configuration.

NOTE The instruction to add all event definition identifiers to the event report blocking storage-control configuration contain no argument

c. Each instruction to add an event definition identifier to the event report blocking storage-control configuration shall contain:

1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction;
2. the event definition identifier.

d. Each instruction to add all event definition identifiers to the event report blocking storage-control configuration shall contain:

1. the application process identifier addressed by that instruction.

e. The packet selection subservice shall reject any request to add event definition identifiers to the event report blocking storage-control configuration if any of the following conditions occurs:
    1. that request refers to a packet store that does not exist;
    2. that request contains an instruction that refers to an application process that is not controlled by that subservice;
    3. the maximum number of event definition identifiers that can be contained within an event report blocking storage-control definition is already reached;
    4. the corresponding event report blocking storage-control definition has no event definition identifier already defined;
    5. that request contains an instruction that refers to an application process that is not controlled by that subservice.

f. For each request to add event definition identifiers to the event report blocking storage-control configuration that is rejected, the packet selection subservice shall generate a failed start of execution notification.

g. The packet selection subservice shall reject any instruction contained within a request to add event definition identifiers to the event report blocking storage-control configuration if:
    1. that instruction refers to an application process that is not controlled by that subservice.

h. For each valid instruction to add an event definition identifier to the event report blocking storage-control configuration, the packet selection subservice shall, for the related packet store:
    1. add, for the specified application process identifier, an event report blocking storage-control definition if not already existing;
    2. add, to the related event report blocking storage-control definition, the specified event definition identifier, if not already existing.

i. For each valid instruction to add all event definition identifiers to the event report blocking storage-control configuration, the packet selection subservice shall, for the related packet store:
    1. add, for the specified application process identifier, an event report blocking storage-control definition if not already existing;
    2. delete, if any, all event definition identifiers of the related event report blocking storage-control definition.

6.15.4.6.2 Delete event definition identifiers from the event report blocking storage-control configuration

a. The packet selection subservice shall provide the capability to delete event definition identifiers from the event report blocking storage-control configuration of a packet store if that subservice provides the capability to control, per event definition, the storage of event reports.

NOTE 2 The corresponding requests are of message type "TC[15,33] delete event definition identifiers from the event report blocking storage-control configuration".

NOTE 3 For the capability to control, per event definition, the storage of event reports, refer to requirement 6.15.4.2.1b.

NOTE 4 For the capability to add event definition identifiers to the event report blocking storage-control configuration, refer to clause 6.15.4.6.1.

b. Each request to delete event definition identifiers from the event report blocking storage-control configuration shall contain the packet store identifier of the packet store whose event report blocking storage-control configuration is to change and exactly one of:

1. any combination of ordered lists of one or more instructions:
   (a) to delete an event definition identifier from the event report blocking storage-control configuration,
   (b) to delete an application process from the event report blocking storage-control configuration;
2. a single instruction to empty the event report blocking storage-control configuration.

NOTE The instructions to empty the event report blocking storage-control configuration contain no argument.

c. Each instruction to delete an event definition identifier from the event report blocking storage-control configuration shall contain:

1. if the packet selection subservice controls more than one application process, the application process identifier addressed by that instruction;
2. the event definition identifier.

NOTE For item 1, refer to requirement 6.15.4.1.1a.

d. Each instruction to delete an application process from the event report blocking storage-control configuration shall contain:

1. the application process identifier addressed by that instruction.

e. The packet selection subservice shall reject any request to delete event definition identifiers from the event report blocking storage-control configuration if any of the following occurs:

1. that request contains an instruction that refers to an application process identifier that is not in the event report blocking control configuration of the related packet store;

2. that request contains an instruction that refers to an event definition identifier that is not in the event report blocking control definition for the specified application process identifier;
3. that request contains an instruction that refers to an application process identifier that is not in the event report blocking control configuration of the related packet store.

f. For each request to delete event definition identifiers from the event report blocking storage-control configuration that is rejected, the packet selection subservice shall generate a failed start of execution notification

g. For each request to delete event definition identifiers from the event report blocking storage-control configuration that contains only valid instructions, the packet selection subservice shall execute those instructions in the order of their appearance in that request

h. For each valid instruction to delete an event definition identifier from the event report blocking storage-control configuration, the packet selection subservice shall, for the related packet store:
1. delete the event definition identifier related to the specified application process identifier;
2. if that event definition identifier deletion results in an emptied event report blocking storage-control definition, delete that event report blocking storage-control definition.

i. For each valid instruction to delete an application process from the event report blocking storage-control configuration, the packet selection subservice shall:
1. for the related packet store, delete the event report blocking control definition for the specified application process identifier.

j. For each valid instruction to empty the event report blocking storage-control configuration, the packet selection subservice shall:
1. for the related packet store, delete all event report blocking storage-control definitions.

6.15.4.6.3 Report the content of the event report blocking storage-control configuration

a. The packet selection subservice capability to report the content of the event report blocking storage-control configuration of a packet store shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[15,39] report the content of the event report blocking storage-control configuration". The responses are data reports of message type "TM[15,40] event report blocking storage-control configuration content report".

NOTE 2 That capability requires the capability for that subservice to control, per event definition, the storage of event reports, refer to requirement 6.15.4.2.1b.

b. Each request to report the content of the event report blocking storage-control configuration shall contain exactly one instruction to report the content of the event report blocking storage-control configuration.

c. Each instruction to report the content of the event report blocking control configuration shall include:

1. the packet store identifier of the packet store.

d. The packet selection subservice shall reject any request to report the content of the event report blocking control configuration if:

1. that request contains an instruction that refers to a packet store that does not exist.

e. For each request to report the content of the event report blocking storage-control configuration that is rejected, the packet selection subservice shall generate a failed start of execution notification.

f. For each valid instruction to report the content of the event report blocking storage-control configuration, the packet selection subservice shall generate, for each existing event report blocking storage-control definition of the related packet store, a single event report blocking storage-control definition notification that includes:

1. if the packet selection subservice controls more than one application process, the related application process identifier;
2. for each event definition identifier entry:

(a) the event definition identifier.

NOTE For item 1, refer to requirement 6.15.4.1.1a.

g. For each valid request to report the content of the event report blocking storage-control configuration, the packet selection subservice shall generate a single event report blocking storage-control configuration content report that includes:

1. the packet store identifier of the related packet store;
2. all related event report blocking storage-control definition notifications.

6.15.4.7 Subservice observables

This Standard does not define any observables for the packet selection subservice.

6.16 ST[16] (reserved)

6.17 ST[17] test

6.17.1 Scope

6.17.1.1 General

The test service type provides the capability to activate test functions implemented on-board and to report the results of such tests.

The test service type defines a single standardized subservice type, i.e. the test subservice type.

6.17.1.2 Test subservice

The test subservice type provides the capability to perform a set of end-to-end test functions that can be exercised under ground control. These include, for example, an are-you-alive function.

6.17.2 Service layout

6.17.2.1 Subservice

6.17.2.1.1 Test subservice

a. Each test service shall contain at least one test subservice.

6.17.2.2 Application process

a. Each application process shall host at most one test subservice provider.

6.17.3 Perform an are-you-alive connection test

a. The test subservice shall provide the capability to perform an are-you-alive connection test.

NOTE 1 The corresponding requests are of message type "TC[17,1] perform an are-you-alive connection test". The responses are data reports of message type "TM[17,2] are-you-alive connection test report".

NOTE 2 The end-to-end connection is achieved when the application process is alive and the communication to the application process is active.

b. Each request to perform an are-you-alive connection test shall contain exactly one instruction to perform an are-you-alive connection test.

NOTE The instructions to perform an are-you-alive connection test contain no argument.

c. For each valid instruction to perform an are-you-alive connection test, the test subservice shall generate a single are-you-alive connection test notification that notifies that the application process that hosts the test subservice is alive and has successfully received the request.

NOTE The are-you-alive connection test notifications contain no parameter.

d. For each valid request to perform an are-you-alive connection test, the test subservice shall generate a single are-you-alive connection test report that includes the related are-you-alive connection test notification.

NOTE The reception on the ground of the report confirms that the communication routes (uplink and downlink) between the ground and the application process are operational and that the application process itself is performing a minimum set of functions.

6.17.4 End-to-end is-application-process-alive connection testing

6.17.4.1 Application process accessibility

a. The list of application processes for which the test subservice can perform an on-board connection testing shall be declared when specifying that subservice.

NOTE The application process that hosts the test subservice is not included in this list.

b. For each application process for which the test subservice can perform an on-board connection testing, the criteria for a successful on-board connection test between that application process and that service shall be declared when specifying that subservice.

6.17.4.2 Perform an on-board connection test

a. The test subservice capability to perform an on-board connection test shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[17,3] perform an on-board connection test". The responses are data reports of message type "TM[17,4] on-board connection test report".

NOTE 2 The on-board connection test is between two on-board application processes, i.e. the one executing the request and the one addressed by the argument of the related instruction.

b. Each request to perform an on-board connection test shall contain exactly one instruction to perform an on-board connection test.

c. Each instruction to perform an on-board connection test shall contain:

1. the identifier of the application process that connection test is requested.

d. The test subservice shall reject any request to perform an on-board connection test if:

1. that request contains an instruction that refers to an application process that is not in the list of application processes for which the test subservice can perform an on-board connection testing.

e. For each request to perform an on-board connection test that is rejected, the test subservice shall generate a failed start of execution notification.

f. For each valid instruction to perform an on-board connection test, the test subservice shall:

1. perform a connection test with the application process referred to by that instruction;
2. if the criteria for a successful on-board connection test with that application process are satisfied, generate a single on-board connection test notification that includes the identifier of the application process that connection has been tested.
3. if the criteria for a successful on-board connection test with that application process are not satisfied, generate a failed completion of execution verification report.

g. For each valid request to perform an on-board connection test, the test subservice shall generate a single on-board connection test report that includes the related on-board connection test notification.

6.17.5 Perform a variable size are-you-alive connection test

a. The test subservice capability to perform a variable size are-you-alive connection test shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[17,5] perform a variable size are-you-alive connection test". The responses are data reports of message type "TM[17,6] are-you-alive variable size connection test report".

NOTE 2 The end-to-end connection is achieved when the application process is alive and the communication to the application process is active.

NOTE 3 This test subservice request allows a test command of arbitrary length and in particular of a max telecommand length for testing purposes.

b. Each request to perform a variable size are-you-alive connection test shall contain exactly one instruction to perform a variable size are-you-alive connection test.

c. Each instruction to perform a variable size are-you-alive connection test shall contain:

1. An arbitrary connection test data string.

NOTE This arbitrary data string is of variable length up to the maximum telecommand length allocated.

d. For each valid instruction to perform a variable size are-you-alive connection test, the test subservice shall generate a single variable size are-you-alive connection test notification containing:

1. The connection test data string provided in that request.

e. For each valid request to perform a variable size are-you-alive connection test, the test subservice shall generate a single variable size are-you-alive connection test report that includes the related are-you-alive connection test notification.

NOTE The reception on the ground of the report confirms that the communication routes (uplink and downlink) between the ground and the application process are operational and that the application process itself is performing a minimum set of functions.

6.17.6 Subservice observables

This Standard does not define any observables for the test subservice.

6.18 ST[18] on-board control procedure

6.18.1 Scope

6.18.1.1 General

The on-board control procedure service type is compliant with and complements the spacecraft on-board control procedures standard (refer to ECSS-E-ST-70-01).

The on-board control procedure service type defines two standardized subservice types, i.e.:

- the OBCP management subservice type;
- the OBCP engine management subservice type.

6.18.1.2 OBCP management subservice

The OBCP management subservice type provides an interface to the OBCP engine that executes OBCPs. The subservice type therefore provides the capability to control, from ground, the on-board execution of OBCPs.

The OBCP code represents the form of the procedure that can be loaded within the OBCP engine for subsequent execution.

A list of OBCP arguments can be associated to an OBCP, corresponding to the values that it expects to receive at execution initiation time. A list of OBCP parameters can also be associated to an OBCP, corresponding to the values that it expects to receive during execution. Refer to ECSS-E-ST-70-31 for OBCP arguments and parameters. The validity of arguments and parameters supplied to an OBCP is checked by the OBCP itself, not by the OBCP management subservice.

ECSS-E-ST-70-01 specifies that the procedures can contain steps that are sequences of OBCP source code statements constituting the smallest operational units within an OBCP. The OBCP management subservice type supports the use of steps in accordance with ECSS-E-ST-70-01. Within the OBCP code, each ECSS-E-ST-70-01 step is represented by exactly one step identifier.

6.18.1.3 OBCP engine management subservice

The OBCP engine management subservice type provides the capability to control the OBCP engine that is responsible for executing the OBCPs.

6.18.2 Service layout

6.18.2.1 Subservice

6.18.2.1.1 OBCP management subservice

a. Each on-board control procedure service shall contain exactly one OBCP management subservice.

6.18.2.1.2 OBCP engine management subservice

a. Each on-board control procedure service shall contain at most one OBCP engine management subservice.

6.18.2.2 Application process

a. For each on-board control procedure service that contains both, an OBCP management subservice and an OBCP engine management subservice, the two subservice providers shall be hosted by the same application process.

b. Each application process shall host at most one OBCP subservice provider.

c. Each application process shall host at most one OBCP engine management subservice provider.

6.18.2.3 OBCP engine

a. Each on-board control procedure service shall be associated to exactly one OBCP engine.

b. The on-board control procedure service shall maintain a status that reflects whether the OBCP engine is running or not.

NOTE 1 This status is called the "OBCP engine status".

NOTE 2 This status exists regardless of the presence of an OBCP engine management subservice to start or stop the OBCP engine.

c. The on-board control procedure service shall automatically start the OBCP engine at initialization time.

NOTE For the OBCP engine initialization procedure, refer to requirement 6.18.5.2.1f.

6.18.3 Accessibility

6.18.3.1 Event reporting

a. The list of event reporting subservices that generate the event reports that can be caught by the on-board control procedure service shall be declared when specifying that on-board control procedure service.

NOTE The event reporting subservice is specified in clause 6.5.

6.18.3.2 Application process

a. The list of application processes that can be addressed by the on-board control procedure service shall be declared when specifying that service.

NOTE 1 The application process that hosts the on-board control procedure service is always part of that list.

NOTE 2 This Standard assumes that all requests of addressable application processes can be used by the on-board control procedure service.

NOTE 3 When the on-board control procedure service releases a request, the request is processed by an executing service, indicated by the service type and the application process identifier within the request. The generation of verification reports for the request is the responsibility of the executing service. The destination of the generated verification reports is the application process that hosts that on-board control procedure service.

6.18.3.3 Parameter

a. The on-board control procedure service shall be able to collect the values of each on-board parameter that is accessible to the application processes that can be addressed by the on-board control procedure service.

NOTE The accessible application processes are those specified by requirement 6.18.3.2a.

6.18.4 OBCP management subservice

6.18.4.1 OBCP definition

6.18.4.1.1 Resources

a. The maximum number of OBCPs that the OBCP management subservice can contemporaneously process at any time shall be declared when specifying that subservice

b. The total resources available to the OBCP engine for storage of OBCPs shall be declared when specifying the OBCP management subservice.

NOTE The resources required for storing the OBCPs include some margin to accommodate additional OBCP developments during in-flight operations.

6.18.4.1.2 OBCP checksum

a. Whether the OBCP management subservice verifies the checksum of the OBCP code when loading an OBCPs into the engine shall be declared when specifying that subservice.

NOTE 1 For the checksum algorithm, refer to clause 5.4.4.

NOTE 2 In a request to direct-load an OBCP, the OBCP checksum is contained directly within the request, see clause 6.18.4.4.2.

NOTE 3 In a request to load an OBCP by reference (see clause 6.18.4.4.3) or a request to load by reference and activate an OBCP (see clause 6.18.4.4.6), the OBCP checksum is contained within the file or as a file attribute.

6.18.4.1.3 OBCP identifier

a. Each OBCP shall have a unique OBCP identifier.

NOTE If the OBCP is loaded from a file, the OBCP identifier can be used by the loading policy as described in clause 6.18.4.4.3. See also E-ST-70-01, requirement 5.1a.

6.18.4.1.4 Scheduling policy

a. The OBCP scheduling policy used by the OBCP management subservice for the OBCP execution shall be declared when specifying that subservice.

NOTE The OBCP scheduling policy ensures that the critical OBCPs are executed with an high priority.

6.18.4.2 OBCP execution observability level

6.18.4.2.1 General

a. For each of the following OBCP execution observability levels, whether the OBCP management subservice supports that observability level shall be declared when specifying that subservice:
   1. at-procedure-level observability;
   2. at-step-level observability;
   3. at-detailed-level observability;
   4. no-observability.

b. If the OBCP management subservice does not support the capability for configuring the OBCP execution observability level, the observability level implemented for that subservice shall be declared when specifying that subservice.

c. If the at-procedure-level OBCP execution observability is selected, the OBCP management subservice shall raise an OBCP execution observability event for each OBCP whose execution status changes to:
   1. "active and running" due to:
      (a) a request to activate that OBCP;
      (b) a request to resume that OBCP;
   2. "active and held" due to:
      (a) a request to suspend that OBCP;
      (b) the end of the step execution demanded by a request to activate and execute one OBCP step;
      (c) the end of the step execution demanded by a request to resume and execute one OBCP step;
   3. "inactive" due to:
      (a) the successful or failed completion of execution of that OBCP;
      (b) a request to abort the execution of that OBCP;
      (c) a request to stop the execution of that OBCP.

      NOTE 1 The activation, suspending, resuming, stopping and aborting of OBCP execution initiated from ground can also be reported as verification reports of the requests provided by the OBCP management subservice.

NOTE 2 This observability level is especially useful to report the execution of OBCPs autonomously initiated from within an OBCP.

NOTE 3 Refer to clause 6.5.3 for additional requirements related to these events. The auxiliary data provided by the event include the OBCP identifier and the conditions that caused the event to occur.

d. If the at-step-level OBCP execution observability is selected, in addition to the "at-procedure-level" OBCP execution events, the OBCP management subservice shall raise an OBCP execution observability event:

1. for each step of the OBCP that has been reached.

NOTE Refer to clause 6.5.3 for additional requirements related to these events.

e. If the at-detailed-level OBCP execution observability is selected, in addition to the "at-procedure-level" and "at-step-level" OBCP execution events, the list of OBCP execution observability events used for that observability level together with their raising conditions shall be declared when specifying the OBCP management subservice.

NOTE 1 For example, an at-detailed-level event can be associated to the initiation of an activity, the execution of a branch (e.g. an IF statement, a loop statement), the execution of a statement.

NOTE 2 Refer to clause 6.5.3 for additional requirements related to these events.

6.18.4.2.2 Accessibility

a. If the OBCP management subservice provides the capability to raise OBCP execution observability related events, the associated event reporting subservice shall be declared when specifying that OBCP management subservice.

NOTE 1 This event reporting subservice is responsible for catching the events generated by the OBCP management subservice and issuing the corresponding event reports.

NOTE 2 The event reporting subservice is specified in clause 6.5.

6.18.4.3 Execution status

a. For each OBCP that is loaded within the OBCP engine, the OBCP management subservice shall maintain the OBCP execution status indicating whether that OBCP is:

1. inactive,
2. active and running;
3. active and held.

NOTE 1 The "active and held" execution status means that the OBCP execution is suspended.

NOTE 2 If an OBCP is waiting for an event, the OBCP execution status is "active and running".

NOTE 3 An OBCP is described as active if it has execution status "active and running" or "active and held". It is described as running if it has execution status "active and running". It is described as held if it has execution status "active and held".

b. The configuration policy to apply when starting and restarting the OBCP management subservice shall be declared when specifying the subservice.

NOTE in case of any specific initialization needed, it will be stated in a dedicated requirement

6.18.4.4 Loading, activating and deleting

6.18.4.4.1 Capability

a. The OBCP management subservice shall provide at least one of the following capabilities to load an OBCP into the OBCP engine:

1. the capability to direct-load an OBCP specified in clause 6.18.4.4.2;
2. the capability to load an OBCP by reference specified in clause 6.18.4.4.3;
3. the capability to load by reference and activate an OBCP specified in clause 6.18.4.4.6.

NOTE 1 Direct loading an OBCP means the corresponding request contains the OBCP code.

NOTE 2 Loading an OBCP by reference means that the OBCP code is already defined on-board within a file. The request to load that OBCP refers to that file and is in accordance with the loading policy defined in E-ST-70-01 clauses 5.4.4.4a, 5.4.4.4b and 5.4.4.4c.

b. If the capability to load an OBCP by reference is provided, the OBCP loading policy that the OBCP management subservice supports shall be declared when specifying that subservice.

NOTE For the OBCP loading policy, refer to ECSS-E-ST-70-01.

6.18.4.4.2 Direct-load an OBCP

a. The OBCP management subservice capability to direct-load an OBCP shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,1] direct-load an OBCP".

NOTE 2 For that declaration, refer to requirement 6.18.4.4.1a.

NOTE 3 For the capability to unload an OBCP, refer to clause 6.18.4.4.4.

b. Each request to direct-load an OBCP shall contain exactly one instruction to direct-load an OBCP.

c. Each instruction to direct-load an OBCP shall contain:

1. the identifier of the OBCP;
2. the OBCP code to load into the OBCP engine;
3. if the OBCP management subservice verifies the checksum of the OBCP code, the checksum of the OBCP code.

NOTE For item 3, refer to requirement 6.18.4.1.2a.

d. If the OBCP management subservice verifies the checksum of the OBCP code contained within the requests to direct-load an OBCP, that subservice shall checksum the OBCP code prior to loading the OBCP code into the OBCP engine.

e. The OBCP management subservice shall reject any request to direct-load an OBCP if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is already in the OBCP engine;
3. the OBCP code in the instruction fails the checksum verification;
4. the OBCP cannot be loaded due to the lack of OBCP engine available resources.

f. For each request to direct-load an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

g. For each valid instruction to direct-load an OBCP, the OBCP management subservice shall:

1. load the OBCP code in the OBCP engine;
2. set the execution status of the OBCP to "inactive".

NOTE The OBCP identifier and the OBCP checksum (if used) are also stored in the OBCP engine.

6.18.4.4.3 Load an OBCP by reference

a. The OBCP management subservice capability to load an OBCP by reference shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,13] load an OBCP by reference".

NOTE 2 For that declaration, refer to requirement 6.18.4.4.1a.

NOTE 3 For the capability to unload an OBCP, refer to clause 6.18.4.4.4.

b. Each request to load an OBCP by reference shall contain exactly one instruction to load an OBCP by reference.

c. Each instruction to load an OBCP by reference shall contain:

1. the identifier of the OBCP;

2. if the OBCP is not to be loaded according to the loading policy, the file path of the on-board file that contains the OBCP code to load into the OBCP engine.

NOTE When the loading policy is used, the policy determines which on-board file contains the OBCP code to load into the OBCP engine, refer to requirement 6.18.4.4.1b.

d. If the OBCP management subservice verifies the checksum of OBCP code, the subservice shall checksum the OBCP code in the on-board file prior to loading the OBCP code into the OBCP engine.

e. The OBCP management subservice shall reject any request to load an OBCP by reference if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is already in the OBCP engine;
3. that request contains an instruction that refers to a file that does not exist;
4. that request contains an instruction that refers to a file that is not recognized as an OBCP file;
5. the on-board file determined by the loading policy does not exist;
6. the OBCP code in the file fails the checksum verification;
7. the OBCP cannot be loaded due to the lack of OBCP engine available resources.

f. For each request to load an OBCP by reference that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

g. For each valid instruction to load an OBCP by reference, the OBCP management subservice shall:

1. load the OBCP code contained in the file into the OBCP engine;
2. set the execution status of the OBCP to "inactive".

NOTE The OBCP identifier and the OBCP checksum (if used) are also stored in the OBCP engine.

6.18.4.4.4 Unload OBCPs

a. The OBCP management subservice shall provide the capability to unload OBCPs if the capability to direct-load an OBCP or the capability to load an OBCP by reference is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,2] unload OBCPs".

NOTE 2 For the capability to direct-load an OBCP, refer to clause 6.18.4.4.2.

NOTE 3 For the capability to load an OBCP by reference, refer to clause 6.18.4.4.3.

b. Each request to unload OBCPs shall contain exactly one of:

1. an ordered list of one or more instructions to unload an OBCP;

2. a single instruction to unload all OBCPs

c. Each instruction to unload an OBCP shall contain:

1. the identifier of the OBCP.

d. The OBCP management subservice shall reject any request to unload OBCPs if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine;
3. that request contains an instruction that refers to an OBCP that is active.

NOTE The unload request can only be used for an OBCP with execution status "inactive".

e. For each request to unload OBCPs that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each request to unload OBCPs that contains only valid instructions, the OBCP management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to unload an OBCP, the OBCP management subservice shall:

1. unload the OBCP from the engine;
2. clean the engine from any information related to that OBCP.

NOTE Item 2 implies that, after removal of the OBCP from the engine, the identifier of that OBCP can be reused.

h. For each valid instruction to unload all OBCPs, the OBCP management subservice shall:

1. unload all OBCPs from the engine;
2. clean the engine from any information related to all OBCPs.

NOTE Item 2 implies that, after removal of all OBCPs from the engine, the identifier of those OBCPs can be reused.

6.18.4.4.5 Activate an OBCP

a. The OBCP management subservice shall provide the capability to activate an OBCP.

NOTE 1 The corresponding requests are of message type "TC[18,3] activate an OBCP".

NOTE 2 For the capability to stop an OBCP, refer to clause 6.18.4.4.7.

b. Each request to activate an OBCP shall contain exactly one instruction to activate an OBCP.

c. Each instruction to activate an OBCP shall contain:

1. the identifier of the OBCP;

2. if selecting the OBCP execution observability level is supported, the observability level to use during the execution of the OBCP;
3. if the OBCP uses arguments, the argument values.

NOTE For item 2, refer to requirement 6.18.4.2.1a.

d. The OBCP management subservice shall reject any request to activate an OBCP if any of the following conditions occurs:
1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine;
3. that request contains an instruction that refers to an observability level that is invalid;
4. that request contains an instruction that refers to an OBCP that is active;
5. that OBCP cannot be activated due to the lack of OBCP engine availability resources.

e. For each request to activate an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each valid instruction to activate an OBCP, the OBCP management subservice shall:
1. remove the execution trace of the previous execution of that OBCP, if any;
2. enable the raising of OBCP execution observability events according to the OBCP execution observability level of that OBCP;
3. set the execution status of the OBCP to "active and running";
4. initiate the execution of the OBCP with the related argument values.

NOTE At the end of execution of the OBCP, the OBCP status is "inactive" and remains loaded in the OBCP engine.

6.18.4.4.6 Load by reference and activate an OBCP

a. The OBCP management subservice capability to load by reference and activate an OBCP shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,19] load by reference and activate an OBCP".

NOTE 2 For that declaration, refer to requirement 6.18.4.4.1a.

NOTE 3 For the capability to stop an OBCP, refer to clause 6.18.4.4.7.

NOTE 4 For the capability to stop and unload an OBCP, refer to clause 6.18.4.4.8.

b. Each request to load by reference and activate an OBCP shall contain exactly one instruction to load by reference and activate an OBCP.

c. Each instruction to load by reference and activate an OBCP shall contain:
1. the identifier of the OBCP;

2. if the OBCP is not loaded according to the loading policy, the file path of the on-board file that contains the OBCP code to load into the OBCP engine;
3. if selecting the OBCP execution observability level is supported, the observability level to use during the execution of the OBCP;
4. if the OBCP uses arguments, the argument values.

NOTE 1 For item 2, refer to requirement 6.18.4.4.1b. When the loading policy is used, the policy determines which on-board file contains the OBCP code to load into the OBCP engine.

NOTE 2 For item 3, refer to requirement 6.18.4.2.1a.

d. If the OBCP management subservice verifies the checksum of OBCP code, the subservice shall checksum the OBCP code in the on-board file prior to loading the OBCP code into the OBCP engine.

e. The OBCP management subservice shall reject any request to load by reference and activate an OBCP if any of the following conditions occurs:
1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is already in the OBCP engine;
3. that request contains an instruction that refers to a file that does not exist;
4. that request contains an instruction that refers to a file that is not recognized as an OBCP file;
5. the on-board file determined by the loading policy does not exist;
6. the OBCP code in the file fails the checksum verification;
7. that request contains an instruction that refers to an observability level that is invalid;
8. that OBCP cannot be loaded and activated due to the lack of OBCP engine available resources.

NOTE Item 8 implies that insufficient resources to activate the OBCP prevents the loading of the OBCP.

f. For each request to load by reference and activate an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

g. For each valid instruction to load by reference and activate an OBCP, the OBCP management subservice shall:
1. load the OBCP code contained in the file into the OBCP engine;
2. enable the raising of OBCP execution observability events according to the OBCP execution observability level of that OBCP;
3. set the execution status of the OBCP to "active and running";
4. initiate the execution of the OBCP with the related argument values.
5. at the end of execution of the OBCP:
   (a) remove the OBCP from the engine;
   (b) clean the engine from any information related to that OBCP.

NOTE 1 The OBCP identifier and the OBCP checksum (if used) are also stored in the OBCP engine.

NOTE 2 Item 5 implies that, after removal of the OBCP from the engine, the identifier of that OBCP can be reused.

6.18.4.4.7 Stop an OBCP

a. The OBCP management subservice shall provide the capability to stop an OBCP.

NOTE 1 The corresponding requests are of message type "TC [18,4] stop an OBCP".

NOTE 2 If several requests to stop an OBCP are received, the OBCP execution stops at the first step reached.

b. Each request to stop an OBCP shall contain exactly one of:
   1. a single instruction to stop an OBCP at the end of current step;
   2. a single instruction to stop an OBCP at the end of a step.

c. Each instruction to stop an OBCP at the end of current step shall contain:
   1. the identifier of the OBCP.

d. Each instruction to stop an OBCP at the end of a step shall contain:
   1. the identifier of the OBCP;
   2. the identifier of that step.

e. The OBCP management subservice shall reject any request to stop an OBCP if any of the following conditions occurs:
   1. the OBCP engine is not running;
   2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine.

f. For each request to stop an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

g. For each valid instruction to stop an OBCP at the end of current step, the OBCP management subservice shall:
   1. if the OBCP is running, wait until the OBCP execution ends the execution of the running step;
   2. freeze the execution of any remaining OBCP statements;
   3. remove the "stop at step" configuration properties resulting from the received requests to stop that OBCP;
   4. set the execution status of the OBCP to "inactive".

h. For each valid instruction to stop an OBCP at the end of a step, the OBCP management subservice shall:
   1. if the OBCP is running, wait until the OBCP execution reaches the execution step referred to in the instruction;
   2. freeze the execution of any remaining OBCP statements;
   3. remove the "stop at step" configuration properties resulting from the received requests to stop that OBCP;
   4. set the execution status of the OBCP to "inactive".

6.18.4.4.8 Stop and unload an OBCP

a. The OBCP management subservice capability to stop and unload an OBCP shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,20] stop and unload an OBCP".

NOTE 2 If several requests to stop and unload an OBCP are received, the OBCP execution stops at the first step reached.

b. Each request to stop and unload an OBCP shall contain exactly one of:

1. a single instruction to stop and unload an OBCP at the end of current step;
2. a single instruction to stop and unload an OBCP at the end of a step.

c. Each instruction to stop and unload an OBCP at the end of current step shall contain:

1. the identifier of the OBCP.

d. Each instruction to stop and unload an OBCP at the end of a step shall contain:

1. the identifier of the OBCP;
2. the identifier of that step.

e. The OBCP management subservice shall reject any request to stop and unload an OBCP if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine.

f. For each request to stop and unload an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

g. For each valid instruction to stop and unload an OBCP at the end of current step, the OBCP management subservice shall:

1. if the OBCP is active:
   (a) if the OBCP is running, wait until the OBCP execution ends the execution of the running step;
   (b) freeze the execution of any remaining OBCP statements;
2. unload the OBCP from the OBCP engine;
3. clean the engine from any remaining information related to that OBCP.

h. For each valid instruction to stop and unload an OBCP at the end of a step, the OBCP management subservice shall:

1. if the OBCP is active:
   (a) if the OBCP is running, wait until the OBCP execution reaches the execution step referred to in the instruction;
   (b) freeze the execution of any remaining OBCP statements;
2. unload the OBCP from the OBCP engine;

3. clean the engine from any remaining information related to that OBCP.

6.18.4.4.9 Abort an OBCP

a. The OBCP management subservice shall provide the capability to abort an OBCP.

NOTE The corresponding requests are of message type "TC[18,12] abort an OBCP".

b. Each request to abort an OBCP shall contain exactly one instruction to abort an OBCP.

c. Each instruction to abort an OBCP shall contain:

1. the identifier of the OBCP.

d. The OBCP management subservice shall reject any request to abort an OBCP if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine.

e. For each request to abort an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each valid instruction to abort an OBCP, the OBCP management subservice shall:

1. if the OBCP is active, freeze the execution of any remaining OBCP statements;
2. set the status of the OBCP to "inactive".

6.18.4.4.10 Abort all OBCPs and report

a. The OBCP management subservice capability to abort all OBCPs and report shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[18,17] abort all OBCPs and report". The responses are data reports of message type "TM[18,18] aborted OBCP report".

b. Each request to abort all OBCPs and report shall contain exactly one instruction to abort all OBCPs and report.

NOTE The instructions to abort all OBCPs and report contain no argument.

c. The OBCP management subservice shall reject any request to abort all OBCPs and report if:

1. the OBCP engine is not running.

d. For each request to abort all OBCPs and report that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

e. For each valid instruction to abort all OBCPs and report, the OBCP management subservice shall:

1. freeze the execution of all OBCP statements;
2. for each active OBCP, set the execution status of that OBCP to "inactive".
3. generate, for each aborted OBCP, a single aborted OBCP notification that includes:
   (a) the identifier of that aborted OBCP.

f. For each valid request to abort all OBCPs and report, the OBCP management subservice shall generate a single aborted OBCP report that includes all related aborted OBCP notifications.

6.18.4.5 Execution status reporting

6.18.4.5.1 Report the execution status of each OBCP

a. The OBCP management subservice capability to report the execution status of each OBCP shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[18,8] report the execution status of each OBCP". The responses are data reports of message type "TM[18,9] OBCP execution status report".

b. Each request to report the execution status of each OBCP shall contain exactly one instruction to report the execution status of each OBCP.

NOTE The instructions to report the execution status of each OBCP contain no argument.

c. The OBCP management subservice shall reject any request to report the execution status of each OBCP if:
1. the OBCP engine is not running.

d. For each request to report the execution status of each OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

e. For each valid instruction to report the execution status of each OBCP, the OBCP management subservice shall:
1. generate, for each OBCP that is loaded within the engine, a single OBCP execution status notification that includes:
   (a) the identifier of that OBCP;
   (b) if the OBCP management subservice verifies the checksum of the OBCP code, the OBCP checksum;
   (c) the execution status of that OBCP;
   (d) if the execution status is "active and running", the identification of the step being executed;
   (e) if the execution status is "active and held", the identification of the next step to execute.

f. For each valid request to report the execution status of each OBCP, the OBCP management subservice shall generate a single OBCP execution status report that includes all related OBCP execution status notifications.

6.18.4.6 Suspending and resuming

6.18.4.6.1 Suspend an OBCP

a. The OBCP management subservice capability to suspend an OBCP shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,5] suspend an OBCP".

NOTE 2 If several requests to suspend an OBCP are received, the OBCP execution suspends at the first step reached.

NOTE 3 For the capability to resume an OBCP, refer to clause 6.18.4.6.2.

b. Each request to suspend an OBCP shall contain exactly one of:
   1. a single instruction to suspend an OBCP at the end of current step;
   2. a single instruction to suspend an OBCP at the end of a step.

c. Each instruction to suspend an OBCP at the end of current step shall contain:
   1. the identifier of the OBCP.

d. Each instruction to suspend an OBCP at the end of a step shall contain:
   1. the identifier of the OBCP;
   2. the identifier of that step.

e. The OBCP management subservice shall reject any request to suspend an OBCP if any of the following conditions occurs:
   1. the OBCP engine is not running;
   2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine;
   3. that request contains an instruction that refers to an OBCP that is not active.

f. For each request to suspend an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

g. For each valid instruction to suspend an OBCP at the end of current step, the OBCP management subservice shall:
   1. if the OBCP is running, wait until the OBCP execution ends the execution of the running step;
   2. freeze the execution of any remaining OBCP statements;
   3. set the execution status of the OBCP to "active and held ".

h. For each valid instruction to suspend an OBCP at the end of a step, the OBCP management subservice shall:
   1. if the OBCP is running, wait until the OBCP execution reaches the execution step referred to in the instruction;
   2. freeze the execution of any remaining OBCP statements;
   3. set the execution status of the OBCP to "active and held ".

6.18.4.6.2 Resume an OBCP

a. The OBCP management subservice shall provide the capability to resume an OBCP if the capability to suspend an OBCP is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,6] resume an OBCP".

NOTE 2 For the capability to suspend an OBCP, refer to clause 6.18.4.6.1.

b. Each request to resume an OBCP shall contain exactly one instruction to resume an OBCP.

c. Each instruction to resume an OBCP shall contain:

1. the identifier of the OBCP.

d. The OBCP management subservice shall reject any request to resume an OBCP if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine;
3. that request contains an instruction that refers to an OBCP that is not active.

e. For each request to resume an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each valid instruction to resume an OBCP, the OBCP management subservice shall:

1. if the execution status of the OBCP is "active and held", unfreeze the execution of the OBCP at the position where it was frozen;
2. set the execution status of the OBCP to "active and running".

6.18.4.6.3 Activate and execute one OBCP step

a. The OBCP management subservice capability to activate and execute one OBCP step shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[18,14] activate and execute one OBCP step".

NOTE 2 For the capability to resume and execute one OBCP step, refer to clause 6.18.4.6.4.

b. Each request to activate and execute one OBCP step shall contain exactly one instruction to activate and execute one OBCP step.

c. Each instruction to activate and execute one OBCP step shall contain:

1. the identifier of the OBCP;
2. if selecting the OBCP execution observability level is supported, the observability level to use during the execution of the OBCP;
3. if the OBCP uses arguments, the argument values.

NOTE For item 2, refer to requirement 6.18.4.2.1a.

d. The OBCP management subservice shall reject any request to activate and execute one OBCP step if any of the following conditions occurs:
   1. the OBCP engine is not running;
   2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine;
   3. that request contains an instruction that refers to an observability level that is invalid;
   4. that request contains an instruction that refers to an OBCP that is active.

e. For each request to activate and execute one OBCP step that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each valid instruction to activate and execute one OBCP step, the OBCP management subservice shall:
   1. remove the execution trace of the previous execution of that OBCP, if any;
   2. enable the raising of OBCP execution observability events according to the OBCP execution observability level of that OBCP;
   3. set the execution status of the OBCP to "active and running";
   4. initiate the execution of the OBCP with the related argument values;
   5. wait until the raising of the first step identifier event;
   6. freeze the execution of any remaining statements;
   7. set the execution status of the OBCP to "active and held".

6.18.4.6.4 Resume and execute one OBCP step

a. The OBCP management subservice shall provide the capability to resume and execute one OBCP step if the capability to activate and execute one OBCP step is provided by that subservice.

   NOTE 1 The corresponding requests are of message type "TC[18,15] resume and execute one OBCP step".

   NOTE 2 For the capability to activate and execute one OBCP step, refer to clause 6.18.4.6.3.

b. Each request to resume and execute one OBCP step shall contain exactly one instruction to resume and execute one OBCP step.

c. Each instruction to resume and execute one OBCP step shall contain:
   1. the identifier of the OBCP.

d. The OBCP management subservice shall reject any request to resume and execute one OBCP step if any of the following conditions occurs:
   1. the OBCP engine is not running;
   2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine;
   3. that request contains an instruction that refers to an OBCP that is not held.

e. For each request to resume and execute one OBCP step that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each valid instruction to resume and execute one OBCP step, the OBCP management subservice shall:

1. set the execution status of the OBCP to "active and running";
2. unfreeze the execution of the OBCP at the position where it was frozen when the OBCP was previously held;
3. wait until the raising of the next step identifier event;
4. freeze the execution of any remaining statements;
5. set the execution status of the OBCP to "active and held".

6.18.4.7 Communicating parameters

6.18.4.7.1 Communicate parameters to an OBCP

a. The OBCP management subservice capability to communicate parameters to an OBCP shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[18,7] communicate parameters to an OBCP".

b. Each request to communicate parameters to an OBCP shall contain exactly one instruction to communicate parameters to an OBCP.

c. Each instruction to communicate parameters to an OBCP shall contain:

1. the identifier of the OBCP;
2. the parameter values.

d. The OBCP management subservice shall reject any request to communicate parameters to an OBCP if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine;
3. that request contains an instruction that refers to an OBCP identifier that is not active.

e. For each request to communicate parameters to an OBCP that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each valid instruction to communicate parameters to an OBCP, the OBCP management subservice shall:

1. provide the parameter values to the OBCP.

6.18.4.8 Tracing

6.18.4.8.1 Set the observability level of OBCPs

a. The OBCP management subservice capability to set the observability level of OBCPs shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[18,16] set the observability level of OBCPs".

b. Each request to set the observability level of OBCPs shall contain an ordered list of one or more instructions to set the observability level of an OBCP.

c. Each instruction to set the observability level of an OBCP shall contain:

1. the identifier of an OBCP;
2. the observability level to set for that OBCP.

d. The OBCP management subservice shall reject any request to set the observability level of OBCPs if any of the following conditions occurs:

1. the OBCP engine is not running;
2. that request contains an instruction that refers to an OBCP identifier that is not loaded in the OBCP engine
3. that instruction refers to an observability level that is invalid;
4. that instruction refers to an OBCP that is not active

e. For each request to set the observability level of OBCPs that is rejected, the OBCP management subservice shall generate a failed start of execution notification.

f. For each request to set the observability level of an OBCP that that contains only valid instructions, the OBCP management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to set the observability level of an OBCP, the OBCP management subservice shall:

1. immediately enable the raising of OBCP execution observability events associated to the new observability level;
2. disable the raising of OBCP execution observability events associated to the previous observability level.

6.18.4.9 Subservice observables

a. The following observables shall be defined for the OBCP management subservice:

1. the OBCP engine running status;
2. For each OBCP loaded in the OBCP engine:
   (a) its identifier;
   (b) its execution status;
   (c) if the execution status is "active" and "running", the identifier of the step being executed;
   (d) if the execution status is "active" and "held", the identifier of the next step to be executed.

6.18.5 OBCP engine management subservice

6.18.5.1 OBCP engine configuration

a. The configuration policy to apply when starting and restarting the OBCP engine management subservice shall be declared when specifying the subservice.

NOTE in case of any specific initialization needed, it will be stated in a dedicated requirement

6.18.5.2 Controlling the OBCP engine

6.18.5.2.1 Start the OBCP engine

a. The OBCP engine management subservice shall provide the capability to start the OBCP engine.

NOTE 1 The corresponding requests are of message type "TC[18,21] start the OBCP engine".

NOTE 2 For the capability to stop the OBCP engine, refer to clause 6.18.5.2.2.

b. Each request to start the OBCP engine shall contain exactly one instruction to start the OBCP engine.

NOTE The instructions to start the OBCP engine contain no argument.

c. The OBCP engine management subservice shall reject any request to start the OBCP engine if:

1. the OBCP engine status is "running".

d. For each request to start the OBCP engine that is rejected, the OBCP engine management subservice shall generate a failed start of execution notification.

e. For each valid instruction to start the OBCP engine, the OBCP engine management subservice shall:

1. run the OBCP engine initialization procedure.

f. The OBCP engine initialization procedure shall be declared when specifying the OBCP engine management subservice.

6.18.5.2.2 Stop the OBCP engine

a. The OBCP engine management subservice shall provide the capability to stop an OBCP engine.

NOTE 1 The corresponding requests are of message type "TC[18,22] stop the OBCP engine".

NOTE 2 For the capability to start the OBCP engine, refer to clause 6.18.5.2.1.

b. Each request to stop the OBCP engine shall contain exactly one instruction to stop an OBCP engine.

NOTE The instructions to stop the OBCP engine contain no argument.

c. The OBCP engine management subservice shall reject any request to stop the OBCP engine if:

1. the OBCP engine is not running.

d. For each request to stop the OBCP engine that is rejected, the OBCP engine management subservice shall generate a failed start of execution notification.

e. For each valid instruction to stop the OBCP engine, the OBCP engine management subservice shall:

1. abort the execution of all OBCPs;
2. unload all OBCPs from the engine;
3. set the OBCP engine status to "not running".

6.18.5.3 Subservice observables

a. The observables used by the OBCP engine management subservice to monitor the status of the OBCP engine shall be declared when specifying this subservice.

NOTE For the observables, refer also to ECSS-E-ST-70-01.

6.19 ST[19] event-action

6.19.1 Scope

6.19.1.1 General

The event-action service type provides the capability to define on-board actions that can be autonomously executed when specific on-board events occur.

This service is associated to one or more event reporting subservices and has the visibility of all event reports generated by these services.

The event-action service type defines a single standardized subservice type, i.e. the event-action subservice type.

6.19.1.2 Event-action subservice

The event-action subservice type includes the capability to maintain a list of event-action definitions. Each event-action definition relates to an event (by means of the corresponding event definition identifier) and the corresponding request (i.e. the action). The subservice reacts to any event occurrence by initiating the execution of the associated request. Such requests can, for example, directly reconfigure hardware, start an on-board control procedure or start a request sequence.

The event-action subservice is an extension of the ground monitoring and control. As such, the application process that executes a request released by the subservice directly sends the request verification reports, if any, to the source identified by the source identifier specified in the request.

6.19.2 Service layout

6.19.2.1 Subservice

6.19.2.1.1 Event-action subservice

a. Each event-action service shall contain at least one event-action subservice.

6.19.2.2 Application process

a. Each application process shall host at most one event-action subservice provider.

6.19.3 Accessibility

6.19.3.1 Event reporting

a. The list of event reporting subservices that generate the event reports used by the event-action subservice shall be declared when specifying that event-action subservice.

NOTE The event reporting subservice is specified in clause 6.5.

b. The event-action subservice shall be associated to at least one event reporting subservice.

c. The event-action subservice shall be able to detect and react to all event reports generated by the associated event reporting subservices.

6.19.3.2 Application process

a. The list of application processes that can be addressed by the event-action subservice when releasing requests shall be declared when specifying that subservice.

NOTE 1 The application process that hosts the event-action subservice is always part of that list.

NOTE 2 This Standard assumes that all requests of addressable application processes can be used by the event-action subservice.

NOTE 3 When the event-action subservice releases a request, the request is processed by an executing service, indicated by the service type and the application process identifier within the request. The generation of the execution verification reports for that request is the responsibility of the executing service.

NOTE 4 Requests released by the event-action subservice are not generated by that subservice but by the source that initiated the add event-action definition request, i.e. the original source.

6.19.4 Event-action definition

a. The maximum number of event-action definitions that the event-action subservice can contemporaneously evaluate at any time shall be declared when specifying that subservice.

b. Each event-action definition shall contain:

1. the system identifier of the event definition associated to an event, that is the combination of:
   (a) if the event-action subservice is associated to more than one event reporting subservice, the identifier of the application process that hosts the event reporting subservice;
   (b) the event definition identifier;
2. the action consisting of the request to release when the event report is detected.

NOTE For item 1(a), refer to requirement 6.19.3.2a.

6.19.5 Processing logic

6.19.5.1 Statuses

a. The event-action subservice shall maintain a status indicating whether the overall event-action function is enabled or disabled.

NOTE This status is named "event-action function status".

b. For each event-action definition, the event-action subservice shall maintain a status indicating whether the event-action definition is enabled or disabled.

NOTE This status is named "event-action status".

c. The configuration policy to apply when starting and restarting the event-action subservice shall be declared when specifying the subservice.

NOTE 1 this includes the overall event-action function status, "event-action function status"

NOTE 2 this includes the event-action status for each event-action definition, "event-action status"

NOTE 3 in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

d. When starting the event-action subservice, the overall event-action function status shall be set to "enabled".

6.19.5.2 Action initiation

a. If the event-action function is disabled, the event-action subservice shall not trigger the action for any event-action definition.

NOTE When the event-action function is disabled, the service does not react to any event reports.

b. When the enabled event-action function detects the occurrence of an event that is used by an enabled event-action definition, the event-action subservice shall immediately trigger the related action.

NOTE 1 Triggering an action implies releasing the associated request.

NOTE 2 Once the action has been triggered and the request released, no change is made to the event-action status of that event-action definition, i.e. it remains enabled.

c. The maximum action generation time that the event-action subservice requires to generate the action resulting from the detection of any event shall be declared when specifying that subservice.

6.19.6 Controlling the event-action function

6.19.6.1 Enable the event-action function

a. The event-action subservice shall provide the capability to enable the event-action function.

NOTE 1 The corresponding requests are of message type "TC[19,8] enable the event-action function".

NOTE 2 For the capability to disable the event-action function, refer to clause 6.19.6.2.

b. Each request to enable the event-action function shall contain exactly one instruction to enable the event-action function.

NOTE The instructions to enable the event-action function contain no argument.

c. For each valid instruction to enable the event-action function, the event-action subservice shall:

1. set the event-action function status to "enabled".

NOTE 1 When the event-action function status is "enabled", the event-action subservice reacts to event reports as specified in requirement 6.19.5.2b.

NOTE 2 Enabling the event-action function has no impact on the event-action status of the event-action definitions.

6.19.6.2 Disable the event-action function

a. The event-action subservice shall provide the capability to disable the event-action function.

NOTE 1 The corresponding requests are of message type "TC[19,9] disable the event-action function".

NOTE 2 For the capability to enable the event-action function, refer to clause 6.19.6.1.

b. Each request to disable the event-action function shall contain exactly one instruction to disable the event-action function.

NOTE The instructions to disable the event-action function contain no argument.

c. For each valid instruction to disable the event-action function, the event-action subservice shall:

1. set the event-action function status to "disabled".

NOTE 1 As specified in requirement 6.19.5.2a, the event-action subservice does not react to event reports when the event-action function status is "disabled".

NOTE 2 Disabling the event-action function has no impact on the event-action status of the event-action definitions.

6.19.7 Controlling the event-action definitions

6.19.7.1 Enable event-action definitions

a. The event-action subservice shall provide the capability to enable event-action definitions.

NOTE 1 The corresponding requests are of message type "TC[19,4] enable event-action definitions".

NOTE 2 For the capability to disable event-action definitions, refer to clause 6.19.6.2.

b. Each request to enable event-action definitions shall contain exactly one of:

1. An ordered list of one or more instructions to enable an event-action definition;
2. a single instruction to enable all event-action definitions.

NOTE The instructions to enable all event-action definitions contain no argument.

c. Each instruction to enable an event-action definition shall contain:

1. the system identifier of the event definition.

NOTE For the system identifier of the event definition, refer to requirement 6.19.4b.1.

d. The event-action subservice shall reject any request to enable event-action definitions if:

1. that instruction refers to an unknown event-action definition.

e. For each request to enable event-action definitions that is rejected, the event-action subservice shall generate a failed start of execution notification.

f. For each request to enable event-action definitions that contains only valid instructions, the event-action subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable an event-action definition, the event-action subservice shall:

1. set the event-action status of that event-action definition to "enabled".

h. For each valid instruction to enable all event-action definitions, the event-action subservice shall:

1. for each event-action definition maintained by that subservice, set its event-action status to "enabled".

6.19.7.2 Disable event-action definitions

a. The event-action subservice shall provide the capability to disable event-action definitions.

NOTE 1 The corresponding requests are of message type "TC[19,5] disable event-action definitions".

NOTE 2 For the capability to enable event-action definitions, refer to clause 6.19.7.1.

b. Each request to disable event-action definitions shall contain exactly one of:

1. an ordered list of one or more instructions to disable an event-action definition;
2. a single instruction to disable all event-action definitions.

NOTE The instructions to disable all event-action definitions contain no argument.

c. Each instruction to disable an event-action definition shall contain:

1. the system identifier of the event definition.

NOTE For the system identifier of the event definition, refer to requirement 6.19.4b.1.

d. The event-action subservice shall reject any request to disable event-action definitions if:

1. that instruction refers to an unknown event-action definition.

e. For each request to disable event-action definitions that is rejected, the event-action subservice shall generate a failed start of execution notification.

f. For each request to disable event-action definitions that contains only valid instructions, the event-action subservice shall execute those instructions in the order of their appearance in that request..

g. For each valid instruction to disable an event-action definition, the event-action subservice shall:

1. set the event-action status of that event-action definition to "disabled".

h. For each valid instruction to disable all event-action definitions, the event-action subservice shall:

1. for each event-action definition maintained by that subservice, set its event-action status to "disabled".

6.19.8 Maintaining event-action definitions

6.19.8.1 Add event-action definitions

a. The event-action subservice shall provide the capability to add event-action definitions.

NOTE 1 The corresponding requests are of message type "TC[19,1] add event-action definitions".

NOTE 2 For the capability to delete event-action definitions, refer to clause 6.19.8.3.

NOTE 3 For the capability to delete all event-action definitions, refer to clause 6.19.8.4.

b. Each request to add event-action definitions shall contain an ordered list of one or more instructions to add an event-action definition.

c. Each instruction to add an event-action definition shall contain:

1. the system identifier of the event definition;
2. the action consisting of the request to release when the event report is detected.

NOTE For the system identifier of the event definition, refer to requirement 6.19.4b.1.

d. The list of verification checks that the event-action subservice shall perform on the request contained in the action of an instruction to add an event-action definition shall be declared when specifying that subservice.

e. The event-action subservice shall reject any request to add an event-action definitions if any of the following conditions occurs:

1. that instruction refers to an event-action definition that is enabled;
2. the maximum number of event-action definitions that the service can contemporaneously evaluate is already reached;
3. the request contained in the action of that instruction fails any of the specified verification checks.

f. For each request to add event-action definitions that is rejected, the event-action subservice shall generate a failed start of execution notification.

g. For each request to add event-action definitions that contains only valid instructions, the event-action subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to add an event-action definition, the event-action subservice shall:

1. if the identifier of the event definition in that instruction does not refer to an existing event-action definition:
   (a) create a new event-action definition with the identifier of the event definition and the action specified in that instruction;
   (b) set the event-action status of the new event-action definition to "disabled".
2. if the identifier of the event definition in that instruction refers to an existing event-action definition:
   (a) replace the previously specified action of the existing event-action definition by the action specified in that instruction.

6.19.8.2 Capability

a. The event-action subservice shall provide at least one of the following capabilities:

1. the capability to delete event-action definitions specified in clause 6.19.8.3;
2. the capability to delete all event-action definitions specified in clause 6.19.8.4.

6.19.8.3 Delete event-action definitions

a. The event-action subservice capability to delete event-action definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[19,2] delete event-action definitions".

NOTE 2 For that declaration, refer to requirement 6.19.8.2a.

NOTE 3 For the capability to add event-action definitions, refer to clause 6.19.8.1.

b. Each request to delete event-action definitions shall contain an ordered list of one or more instructions to delete an event-action definition.

c. Each instruction to delete an event-action definition shall contain:

1. the system identifier of the event definition.

NOTE For the identifier of the event definition, refer to requirement 6.19.4b.1.

d. The event-action subservice shall reject any request to delete event-action definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to an event-action definition that is enabled;

2. that request contains an instruction that refers to an unknown event-action definition.

e. For each request to delete event-action definitions that is rejected, the event-action subservice shall generate a failed start of execution notification.

f. For each request to delete event-action definitions that contains only valid instructions, the event-action subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete an event-action definition, the event-action subservice shall:

1. delete the event-action definition specified by that instruction.

6.19.8.4 Delete all event-action definitions

a. The event-action subservice capability to delete all event-action definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[19,3] delete all event-action definitions".

NOTE 2 For that declaration, refer to requirement 6.19.8.2a.

NOTE 3 For the capability to add event-action definitions, refer to clause 6.19.8.1.

b. Each request to delete all event-action definitions shall contain exactly one instruction to delete all event-action definitions.

NOTE The instructions to delete all event-action definitions contain no argument.

c. For each valid instruction to delete all event-action definitions, the event-action subservice shall:

1. set the event-action function status to "disabled";

2. delete all event-action definitions.

NOTE Each event-action definition is deleted without regard to its enabled or disabled event-action status.

6.19.8.5 Report the status of each event-action definition

a. The event-action subservice capability to report the status of each event-action definition shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[19,6] report the status of each event-action definition". The responses are data reports of message type "TM[19,7] event-action status report".

b. Each request to report the status of each event-action definition shall contain exactly one instruction to report the status of each event-action definition.

NOTE The instructions to report the status of each event-action definition contain no argument.

c. For each valid instruction to report the status of each event-action definition, the event-action subservice shall generate, for each event-action definition, a single event-action status notification that includes:

1. the system identifier of the event definition;
2. the event-action status.

NOTE For the identifier of the event definition, see requirement 6.19.4b.1.

d. For each valid request to report the status of each event-action definition, the event-action subservice shall generate a single event-action status report that includes all related event-action status notifications.

6.19.8.6 Report event-action definitions

a. The event-action subservice capability to report event-action definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[19,10] report event-action definitions". The responses are data reports of message type "TM[19,11] event-action definition report".

NOTE 2 That capability requires the capability for that subservice to add event-action definitions, refer to clause 6.19.8.1.

b. Each request to report event-action definitions shall contain exactly one of:

1. an ordered list of one or more instructions to report an event-action definition;
2. a single instruction to report all event-action definitions.

NOTE The instructions to report all event-action definitions contain no argument.

c. Each instruction to report an event-action definition shall contain:

1. the system identifier of the event definition.

NOTE For the system identifier of the event definition, refer to requirement 6.19.4b.1.

d. The event-action subservice shall reject any request to report event-action definitions if:

1. that instruction refers to an unknown event-action definition.

e. For each request to report event-action definitions that is rejected, the event-action subservice shall generate the failed start of execution notification.

f. For each request to report event-action definitions that contains only valid instructions, the event-action subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report an event-action definition, the event-action subservice shall generate a single event-action definition notification that includes:

1. the system identifier of the event definition;
2. the event-action status;
3. the action consisting of the request to release when the event report is detected.

NOTE For the system identifier of the event definition, refer to requirement 6.19.4b.1.

h. For each valid instruction to report all event-action definitions, the event-action subservice shall generate a single event-action definition notification for each event-action definition.

NOTE For the content of the event-action definition notification, see 6.19.8.6g.

i. For each valid request to report event-action definitions, the event-action subservice shall generate a single event-action definition report that includes all related event-action definition notifications.

6.19.9 Subservice observables

a. The following observables shall be defined for the event-action subservice:

1. the event-action function status;
2. the number of event-action definitions;
3. the number of actions that have been triggered due to any event occurrence detection;
4. the number of event-action definitions that are enabled;
5. the number of event-action definitions that are disabled.

6.20 ST[20] parameter management

6.20.1 Scope

6.20.1.1 General

The parameter management service type provides capabilities for managing on-board parameters, including reading current values, setting new values and redefining parameter locations and properties.

The parameter management service type defines a single standardized subservice type, i.e. the parameter management subservice type.

6.20.1.2 Parameter management subservice

The parameter management subservice type includes the capability to maintain a list of parameter definitions, where each definition consists of the parameter identifier, the mapped on-board memory address and the packet field code. The parameter identifiers are predefined and unique within the context of the spacecraft. The packet field code contains a packet field type code (PTC) and format code (PFC) as specified in 7.3.

The parameter management subservice type includes optional capability to create new parameters by associating a new parameter memory location or a new field code to a predefined parameter identifier. For example, a new parameter can be used for an OBCP or for exporting an existing internal variable as a parameter for housekeeping or monitoring.

6.20.2 Service layout

6.20.2.1 Subservice

6.20.2.1.1 Parameter management subservice

a. Each parameter management service shall contain at least one parameter management subservice.

6.20.2.2 Application process

a. Each application process shall host at most one parameter management subservice provider.

6.20.3 Parameter definition

a. The list of parameter identifiers for which the parameter management subservice manages their definition shall be declared when specifying that subservice.

b. Each parameter definition shall consist of:
   1. an on-board parameter identifier that is unique within the context of the spacecraft;

2. if the parameter management subservice manages more than one memory, a memory ID;
3. an address that is either:
   (a) a base plus offset, if that memory ID refers to a memory that uses a base plus offset addressing scheme;
   (b) a byte offset from the start of the memory if that memory ID refers to a memory that uses an absolute addressing scheme;
4. the packet field code of the memory field that is used to read and/or write the values of the parameter.

NOTE 1 For item 2, refer to requirement 6.20.5.1b.

NOTE 2 For item 4, refer to clause7.3.

6.20.4 Managing parameter values

6.20.4.1 Report parameter values

a. The parameter management subservice shall provide the capability to report parameter values.

NOTE The corresponding requests are of message type "TC[20,1] report parameter values". The responses are data reports of message type "TM[20,2] parameter value report".

b. Each request to report parameter values shall contain an ordered list of one or more instructions to report a parameter value.

c. Each instruction to report a parameter value shall contain:
   1. the identifier of the parameter.

d. The parameter management subservice shall reject any request to report parameter values if:
   1. that request contains an instruction that refers to an unknown parameter.

e. For each request to report parameter values that is rejected, the parameter management subservice shall generate a failed start of execution notification.

f. For each request to report parameter values that contains only valid instructions, the parameter management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report a parameter value, the parameter management subservice shall generate a single parameter value notification that includes:
   1. the parameter identifier;
   2. its value.

h. For each valid request to report parameter values, the parameter management subservice shall generate a single parameter value report that contains all related parameter value notifications.

6.20.4.2 Set parameter values

a. The parameter management subservice capability to set parameter values shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[20,3] set parameter values".

b. Each request to set parameter values shall contain an ordered list of one or more instructions to set a parameter value.

c. Each instruction to set a parameter value shall contain:

1. the identifier of the parameter;
2. the new value for the parameter.

d. The parameter management subservice shall reject any request to set parameter values if:

1. that request contains an instrcution that refers to an unknown parameter.

e. For each request to set parameter values that is rejected, the parameter management subservice shall generate a failed start of execution notification.

f. For each request to set parameter values that contains only valid instructions, the parameter management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to set a parameter value, the parameter management subservice shall:

1. set the value of the parameter identified in that instruction to the new value specified in that instruction.

6.20.5 Managing parameter definitions

6.20.5.1 Accessibility

a. The list of accessible parameters for which the parameter management subservice can change the parameter definition shall be declared when specifying that subservice.

NOTE 1 For the accessible parameters, see requirement 6.20.3a.

NOTE 2 Changing the definition of a parameter affects any service that makes use of that parameter.

b. The list of memories that the parameter management subservice uses for managing parameter definitions shall be declared when specifying that subservice.

NOTE This allows restricting the memories on which new parameters can be mapped.

6.20.5.2 Change raw memory parameter definitions

a. The parameter management subservice capability to change raw memory parameter definitions shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[20,4] change raw memory parameter definitions".

b. Each request to change raw memory parameter definitions shall contain an ordered list of one or more instructions to change a raw memory parameter definition.

c. Each instruction to change a raw memory parameter definition shall contain:

1. the identifier of the parameter definition that corresponds to the parameter identifier;
2. if the parameter management subservice manages more than one memory, the memory identifier of the new parameter;
3. the start address of the new parameter specified as a byte offset;
4. the packet field code of the new parameter made of:
   (a) the packet field type code;
   (b) the packet field format code.

NOTE 1 For item 2, refer to requirement 6.20.5.1b.

NOTE 2 For item 4, refer to clause 7.3.

d. The parameter management subservice shall reject any request to change raw memory parameter definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a parameter definition identifier that is unknown;
2. that request contains an instruction that refers to a memory identifier that is not allowed for parameter definition;
3. that request contains an instruction that refers to a memory address that is invalid;
4. that request contains an instruction that refers to a packet field code is not compatible with the memory alignment access constraint;
5. that request contains an instruction that refers to a packet field code that is invalid.

NOTE For item 4, refer to requirement 7.3.1.a.

e. For each request to change raw memory parameter definitions that is rejected, the parameter management subservice shall generate a failed start of execution notification.

f. For each request to change raw memory parameter definitions that contains only valid instructions, the parameter management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to change a raw memory parameter definition, the parameter management subservice shall:

1. set the new parameter definition as required.

6.20.5.3 Change object memory parameter definitions

a. The parameter management subservice capability to change object memory parameter definitions shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[20,5] change object memory parameter definitions".

b. Each request to change object memory parameter definitions shall contain an ordered list of one or more instructions to change an object memory parameter definition.

c. Each instruction to change an object memory parameter definition shall contain:

1. the identifier of the parameter definition that corresponds to the parameter identifier;
2. if the parameter management subservice manages more than one memory, the memory identifier of the new object;
3. the memory address of the new object specified as a base plus an offset;
4. the packet field code of the new object made of:
   (a) the packet field type code;
   (b) the packet field format code.

NOTE 1 For item 2, refer to requirement 6.20.5.1b.

NOTE 2 For item 3, refer to requirement 5.4.3.3.2.c..

NOTE 3 For item 4, refer to clause 7.3.

d. The parameter management subservice shall reject any request to change object memory parameter definitions if any of the following conditions occurs:

1. that request contains an instruction that refers to a parameter definition identifier that is unknown;
2. that request contains an instruction that instruction refers to a memory identifier that is not allowed for parameter definition;
3. that request contains an instruction that instruction refers to a memory address that is invalid;
4. that request contains an instruction that instruction refers to a packet field code is not compatible with the memory alignment access constraint;
5. that request contains an instruction that instruction refers to a packet field code that is invalid.

NOTE For item 4, refer to requirement 7.3.1.a.

e. For each request to change object memory parameter definitions that is rejected, the parameter management subservice shall generate a failed start of execution notification.

f. For each request to change object memory parameter definitions that contains only valid instructions, the parameter management subservice

shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to change an object memory parameter definition, the parameter management subservice shall:

1. set the new parameter definition as required.

6.20.5.4 Report parameter definitions

a. The parameter management subservice capability to report parameter definitions shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[20,6] report parameter definitions". The responses are data reports of message type "TM[20,7] parameter definition report".

NOTE 2 That capability requires the capability for that subservice to provide at least one of:

- the capability to change raw memory parameter definitions (refer to clause 6.20.5.2);
- the capability to change object memory parameter definitions (refer to clause 6.20.5.3).

b. Each request to report parameter definitions shall contain an ordered list of one or more instructions to report a parameter definition.

c. Each instruction to report a parameter definition shall contain:

1. the identifier of the parameter.

d. The parameter management subservice shall reject any request to report parameter definitions if:

1. that request contains an instruction that refers to an unknown parameter.

e. For each request to report parameter definitions that is rejected, the parameter management subservice shall generate a failed start of execution notification.

f. For each request to report parameter definitions that contains only valid instructions, the parameter management subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report a parameter definition, the parameter management subservice shall generate a single parameter definition notification that includes:

1. the parameter identifier;
2. if the parameter management subservice manages more than one memory, the memory identifier;
3. if a base plus offset addressing scheme is used for accessing any memory managed by the parameter management subservice, the memory related addressing scheme;
4. if the addressing scheme is absolute address, the absolute address;
5. if the addressing scheme is base plus offset, the base plus offset;

6. the packet field code of the parameter.

NOTE 1 For item 2, refer to requirement 6.20.5.1b.

NOTE 2 For item 3, refer to requirement 5.4.3.3.2.c.

6.20.6 Subservice observables

This Standard does not define any observables for the parameter management subservice.

6.21 ST[21] request sequencing

6.21.1 Scope

6.21.1.1 General

The request sequencing service type provides the capability to manage the release of an on-board sequence of requests. It also provides capabilities for the loading, control and reporting of on-board sequences.

The request sequencing service type defines a single standardized subservice type, i.e. the request sequencing subservice type.

6.21.1.2 Request sequencing subservice

The request sequencing subservice type provides the capability to release, one by one, the requests contained in an on-board sequence of requests. Within a request sequence, the delay between the release of a request and the release of the next request can be specified. Several request sequences can be running in parallel.

This provides an extension of the ground monitoring and control. As such, the application process that executes a request released by the request sequencing subservice directly sends the request verification reports, if any, to the source identified by the source identifier specified in the request. The release of a request by the subservice is not conditional on the successful or unsuccessful execution of earlier requests released by the subservice.

The subservice type provides the capability to load a request sequence from a file stored on-board or directly from ground. When loading directly from ground, the requests that constitute the request sequence are inside the load request sequence request.

6.21.2 Service layout

6.21.2.1 Subservice

6.21.2.1.1 Request sequencing subservice

a. Each request sequencing service shall contain at least one request sequencing subservice.

6.21.2.2 Application process

a. Each application process shall host at most one request sequencing subservice provider.

6.21.3 Accessibility

6.21.3.1 Application process

a. The list of application processes that are addressed by the request sequencing subservice when releasing requests shall be declared when specifying that subservice.

NOTE 1 The application process that hosts the request sequencing subservice is by nature, an addressable application process.

NOTE 2 This Standard assumes that all requests of addressable application processes can be used by the request sequencing subservice.

NOTE 3 When the request sequencing subservice releases a request, the request is processed by the service, which is indicated by the service type and hosted by the application process identified within the request.

NOTE 4 Requests released by the request sequencing subservice are not generated by that service but by the subservice that initiated the request to load the request sequence, i.e. the original source.

6.21.4 Request sequence

a. The maximum number of request sequences that the request sequencing subservice can contemporaneously process at any time shall be declared when specifying that subservice.

b. The total resources available to the request sequencing subservice for storage of request sequences shall be declared when specifying that subservice.

NOTE This includes the directories to be used as folders of request sequence files, see 6.21.5.a

c. The list of verification checks that the request sequencing subservice shall perform on the requests contained within the request sequences shall be declared when specifying that subservice.

d. Each request sequence shall have a unique request sequence identifier.

NOTE The request sequence identifier is unique within the context of the request sequencing service. If the sequence is loaded from a file, the request sequence identifier can be used by the loading policy as described in clause 6.21.6.3.

e. For each loaded request sequence, the request sequencing subservice shall maintain a status indicating whether that request sequence is inactive or under execution.

NOTE This status is named "request sequence execution status".

f. Each request sequence shall contain

1. an ordered list of request entries.

NOTE 1 This Standard does not constrain the maximum number of request entries that a request sequence can contain.

NOTE 2 This Standard does not constrain the maximum number of request entries that the service can handle.

g. Each request entry shall contain:

1. a single request;
2. a time interval that is the delay between the release of this request and the release of the next request in the request sequence.

NOTE The time interval for the last entry of a request sequence is the delay between the release of the last request and the completion of the execution of the request sequence.

6.21.5 Usage of directories

a. The maximum number of directories that the request sequencing subservice can maintain at any time shall be declared when specifying that subservice.

b. Each directory used for request sequence files shall support the following attributes:

1. immediate execution of request sequence files after creation;
2. immediate deletion of request sequence files after execution.

NOTE the request sequence file will be created in the directory if it is uploaded or moved/copied from another on-board directory

c. when the “immediate execution of request sequence files after creation” attribute is set for a directory, the request sequencing subservice shall process the execution of the request sequence files immediately after creation as specified in 6.21.6.6.

d. when the “immediate deletion of request sequence files after execution” attribute is set for a directory, the request sequencing subservice shall process the deletion of the request sequence files immediately after execution as specified in 6.23.4.1.2.

e. For each request sequence directory, the default value of its attributes shall be declared when specifying that subservice.

6.21.6 Loading, activating and unloading a request sequence

6.21.6.1 Capability

a. The request sequencing subservice shall provide at least one of the following capabilities:
   1. the capability to directly load a request sequence specified in clause 6.21.6.2;
   2. the capability to load a request sequence by reference specified in clause 6.21.6.3;
   3. the capability to load by reference and activate a request sequence specified in clause 6.21.6.6.

      NOTE 1 Directly loading a request sequence means the corresponding load sequence request contains the requests that constitute the sequence.

      NOTE 2 Loading a request sequence by reference means that the request sequence is already defined on-board within a file. The request to load that request sequence refers to that file.

b. If the capability to load a request sequence by reference is provided, whether the request sequencing subservice supports a loading policy shall be declared when specifying that subservice.

c. If the capability to load by reference and activate a request sequence is provided by the request sequencing subservice, the delta time required on-board to activate any request sequence and initiate the execution of the first request of that request sequence shall be declared when specifying that subservice.

6.21.6.2 Direct-load a request sequence

a. The request sequencing subservice capability to direct-load a request sequence shall be declared when specifying that subservice.

   NOTE 1 The corresponding requests are of message type "TC[21,1] direct-load a request sequence".

   NOTE 2 For that declaration, refer to requirement 6.21.6.1a.

   NOTE 3 For the capability to unload a request sequence, refer to clause 6.21.6.4.

b. Each request to direct-load a request sequence shall contain exactly one instruction to direct-load a request sequence.

c. Each instruction to direct-load a request sequence shall contain:
   1. the identifier of the request sequence;
   2. the ordered list of request entries for the request sequence.

      NOTE The contents of a request entry are defined in requirement 6.21.4g.

d. The request sequencing subservice shall reject any request to direct-load a request sequence if any of the following conditions occurs:

1. that request contains an instruction with a request sequence identifier that refers to a request sequence that is already loaded;
2. the request sequence cannot be loaded due to the lack of resources available to the request sequencing subservice;
3. any request contained in that request sequence fails any of the verification checks.

NOTE For the verification checks, see requirement 6.21.4c.

e. For each request to direct-load a request sequence that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to direct-load a request sequence, the request sequencing subservice shall:
1. load the request sequence;
2. set the execution status of the request sequence to "inactive".

6.21.6.3 Load a request sequence by reference

a. The request sequencing subservice capability to load a request sequence by reference shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[21,2] load a request sequence by reference".

NOTE 2 For that declaration, refer to requirement 6.21.6.1a.

NOTE 3 For the capability to unload a request sequence, refer to clause 6.21.6.4.

b. Each request to load a request sequence by reference shall contain exactly one instruction to load a request sequence by reference.

c. Each instruction to load a request sequence by reference shall contain:
1. the identifier of the request sequence;
2. if the request sequence is not to be loaded according to the loading policy, the file path of the on-board file that contains the request sequence to load.

NOTE When the loading policy is used, the policy determines which on-board file contains the request sequence to load, refer to requirement 6.21.6.1b.

d. The request sequencing subservice shall reject any request to load a request sequence by reference if any of the following conditions occurs:
1. that request contains an instruction with a request sequence identifier that refers to a request sequence that is already loaded;
2. the request sequence cannot be loaded due to the lack of resources available to the request sequencing subservice;
3. that request contains an instruction that refers to a file that does not exist;
4. that request contains an instruction that refers to a file that is not recognized as a request sequence file;
5. any request contained in that request sequence fails any of the verification checks.

NOTE For the verification checks, see requirement 6.21.4c.

e. For each request to load a request sequence by reference that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to load a request sequence by reference, the request sequencing subservice shall:

1. load the request sequence;
2. set the execution status of the request sequence to "inactive".

6.21.6.4 Unload a request sequence

a. The request sequencing subservice shall provide the capability to unload a request sequence if the capability to direct-load a request sequence or the capability to load a request sequence by reference is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[21,3] unload a request sequence".

NOTE 2 For the capability to direct-load a request sequence, refer to clause 6.21.6.2.

NOTE 3 For the capability to load a request sequence by reference, refer to clause 6.21.6.3.

b. Each request to unload a request sequence shall contain exactly one instruction to unload a request sequence.

c. Each instruction to unload a request sequence shall contain:

1. the identifier of the request sequence to unload.

d. The request sequencing subservice shall reject any request to unload a request sequence if any of the following conditions occurs:

1. that request contains an instruction with a request sequence identifier that refers to a request sequence that is not loaded;
2. that request contains an instruction that refers to a request sequence whose execution status is "under execution".

e. For each request to unload a request sequence that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to unload a request sequence, the request sequencing subservice shall:

1. unload the request sequence.

6.21.6.5 Activate a request sequence

a. The request sequencing subservice shall provide the capability to activate a request sequence.

NOTE The corresponding requests are of message type "TC[21,4] activate a request sequence".

b. Each request to activate a request sequence shall contain exactly one instruction to activate a request sequence.

c. Each instruction to activate a request sequence shall contain:

1. the identifier of the request sequence to activate.

d. The request sequencing subservice shall reject any request to activate a request sequence if any of the following conditions occurs:

1. that request contains an instruction with a request sequence identifier that refers to a request sequence that is not loaded;
2. the request sequence cannot be activated due to the lack of resources available to the request sequencing subservice;
3. that request contains an instruction that refers to a request sequence whose execution status is "under execution".

e. For each request to activate a request sequence that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to activate a request sequence, the request sequencing subservice shall:

1. set the execution status of the request sequence to "under execution";
2. start releasing the requests in the request sequence;
3. upon release of the last request and the elapse of its associated time interval, set the execution status of the request sequence to "inactive".

NOTE Request sequences are persistent. To unload the request sequence at the end of execution, the last request in the sequence can be the request to unload the request sequence.

6.21.6.6 Load by reference and activate a request sequence

a. The request sequencing subservice capability to load by reference and activate a request sequence shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[21,8] load by reference and activate a request sequence".

NOTE 2 For that declaration, refer to requirement 6.21.6.1a.

b. Each request to load by reference and activate a request sequence shall contain exactly one instruction to load by reference and activate a request sequence.

c. Each instruction to load by reference and activate a request sequence shall contain:

1. the identifier of the request sequence;
2. if the request sequence is not to be loaded according to the loading policy, the file path of the on-board file that contains the request sequence to load and activate.

NOTE 1 a specific value for the request sequence identifier is reserved to indicate that the next free identifier will be used, refer to requirement 8.21.2.8.c

NOTE 1 When the loading policy is used, the policy determines which on-board file contains the request sequence to load, refer to requirement 6.21.6.1b.

d. The request sequencing subservice shall reject any request to load by reference and activate a request sequence if any of the following conditions occurs:

1. that request refers to a request sequence identifier that is already used;
2. the request sequence cannot be loaded and activated due to the lack of resources available to the request sequencing subservice;
3. that request contains an instruction that refers to a file that does not exist;
4. that request contains an instruction that refers to a file that is not recognized as a request sequence file;
5. any request contained in that request sequence fails any of the verification checks.

NOTE For the verification checks, see requirement 6.21.4c.

e. For each request to load by reference and activate a request sequence that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to load by reference and activate a request sequence, the request sequencing subservice shall:

1. load the request sequence;
2. set the execution status of the request sequence to "under execution";
3. start releasing the requests in the request sequence;
4. upon release of the last request and the elapse of its associated time interval:
   (a) set the execution status of the request sequence to "inactive";
   (b) unload the request sequence.

6.21.6.7 Abort a request sequence

a. The request sequencing subservice shall provide the capability to abort a request sequence.

NOTE The corresponding requests are of message type "TC[21,5] abort a request sequence".

b. Each request to abort a request sequence shall contain exactly one instruction to abort a request sequence.

c. Each instruction to abort a request sequence shall contain:

1. the identifier of the request sequence to abort.

d. The request sequencing subservice shall reject any request to abort a request sequence if any of the following conditions occurs:

1. that request contains an instruction with a request sequence identifier that refers to a request sequence that is not loaded;

2. that request contains an instruction that refers to a request sequence whose execution status is "inactive".

e. For each request to abort a request sequence that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to abort a request sequence, the request sequencing subservice shall:

1. stop releasing the requests in the request sequence;
2. set the execution status of the request sequence to "inactive".

NOTE an instruction to abort a request sequence does not unload it

6.21.6.8 Abort all request sequences and report

a. The request sequencing subservice capability to abort all request sequences and report shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[21,13] abort all request sequences and report". The responses are data reports of message type "TM[21,14] aborted request sequence report".

b. Each request to abort all request sequences and report shall contain exactly one instruction to abort all request sequences and report.

NOTE The instructions to abort all request sequences and report contain no argument.

c. For each valid instruction to abort all request sequences and report, the request sequencing subservice shall:

1. for each request sequence that is under execution:
   (a) stop releasing the requests in that request sequence;
   (b) set the execution status of that request sequence to "inactive";
   (c) generate a single aborted request sequence notification that includes the identifier of that request sequence.

NOTE an instruction to abort all request sequences and report does not unload the request sequences

d. For each valid request to abort all request sequences and report, the request sequencing subservice shall generate a single aborted request sequence report that includes all related aborted request sequence notifications.

6.21.7 Report the execution status of each request sequence

a. The request sequencing subservice capability to report the execution status of each request sequence shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[21,6] report the execution status of each request

sequence". The responses are data reports of message type "TM[21,7] request sequence execution status report".

b. Each request to report the execution status of each request sequence shall contain exactly one instruction to report the execution status of each request sequence.

NOTE The instructions to report the execution status of each request sequence contain no argument.

c. For each valid instruction to report the execution status of each request sequence, the request sequencing subservice shall:

1. generate, for each request sequence that is currently loaded, a single request sequence execution status notification that includes:
   (a) the request sequence identifier;
   (b) the request sequence execution status.

d. For each valid request to report the execution status of each request sequence, the request sequencing subservice shall generate a single request sequence execution status report that includes all related request sequence execution status notifications.

6.21.8 Checksum a request sequence

a. The request sequencing subservice capability to checksum a request sequence shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[21,9] checksum a request sequence". The responses are data reports of message type "TM[21,10] request sequence checksum report".

NOTE 2 For the checksum algorithm, refer to clause 5.4.4..

b. Each request to checksum a request sequence shall contain exactly one instruction to checksum a request sequence.

c. Each instruction to checksum a request sequence shall contain:

1. the identifier of the request sequence to checksum.

d. The request sequencing subservice shall reject any request to checksum a request sequence if:

1. that request contains an instruction with a request sequence identifier that refers to a request sequence that is not loaded.

e. For each request to checksum a request sequence that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to checksum a request sequence, the request sequencing subservice shall generate a single request sequence checksum notification that includes:
   (a) the request sequence identifier;
   (b) the calculated checksum value.

g. For each valid request to checksum a request sequence, the request sequencing subservice shall generate a single request sequence checksum report that includes the related request sequence checksum notification.

6.21.9 Report the content of a request sequence

a. The request sequencing subservice capability to report the content of a request sequence shall be declared when specifying that subservice

NOTE The corresponding requests are of message type "TC[21,11] report the content of a request sequence". The responses are data reports of message type "TM[21,12] request sequence content report".

b. Each request to report the content of a request sequence shall contain exactly one instruction to report the content of a request sequence.

c. Each instruction to report the content of a request sequence shall contain:

1. the identifier of the request sequence to report.

d. The request sequencing subservice shall reject any request to report the content of a request sequence if:

1. that request contains an instruction with a request sequence identifier that refers to a request sequence that is not loaded.

e. For each request to report the content of a request sequence that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to report the content of a request sequence, the request sequencing subservice shall generate a single request sequence content notification that includes:

(a) the request sequence identifier;

(b) the ordered list of request entries.

g. For each valid request to report the content of a request sequence, the request sequencing subservice shall generate a single request sequence content report that includes the related request sequence content notification.

6.21.10 Change request sequence directory attributes

a. The request sequencing subservice capability to change the attributes of a request sequence directory shall be declared when specifying that subservice

NOTE The corresponding requests are of message type "TC[21,15] change request sequence directory attributes ".

b. Each request to change the attributes of a request sequence directory shall contain exactly one instruction to change the attributes of a request sequence directory.

c. Each instruction to change the attributes of a request sequence directory shall contain:

1. the request sequence directory path;
2. the new value for the immediate execution after creation attribute;
3. the new value for the immediate deletion after execution attribute.

d. The storage and retrieval subservice shall reject any request to change the attributes of a request sequence directory if any of the following conditions occurs:

1. that request contains an instruction with an invalid directory path;
2. that request contains an instruction with an invalid value for the immediate execution after creation attribute;
3. that request contains an instruction with an invalid value for the immediate deletion after execution attribute.

e. For each request to the attributes of a request sequence directory that is rejected, the storage and retrieval subservice shall generate a failed start of execution notification.

f. For each valid instruction to change the attributes of a request sequence directory, the storage and retrieval subservice shall:

1. change the immediate execution after creation attribute of the directory specified in that instruction with the value specified in that instruction;
2. change the immediate deletion after execution attribute of the directory specified in that instruction with the value specified in that instruction.

6.21.11 Report request sequence directory attributes

a. The request sequencing subservice capability to report the attributes of a request sequence directory shall be declared when specifying that subservice

NOTE The corresponding requests are of message type "TC[21,16] report the attributes of a request sequence directory ". The responses are data reports of message type "TM[21,17] request sequence directory attributes report".

b. Each request to report the attributes of a request sequence directory shall contain exactly one instruction to report the attributes of a request sequence directory.

c. Each instruction to report the attributes of a request sequence directory shall contain:

1. the request sequence directory path.

d. The request sequencing subservice shall reject any request to report the attributes of a request sequence directory if:

1. that request contains an instruction with a directory path that is invalid.

e. For each request to report the attributes of a request sequence directory that is rejected, the request sequencing subservice shall generate a failed start of execution notification.

f. For each valid instruction to report the attributes of a request sequence directory, the request sequencing subservice shall generate a single request sequence directory attributes notification that includes:

(a) the request sequence directory path;

(b) the value of the immediate execution after creation attribute;

(c) the value of the immediate deletion after execution attribute.

g. For each valid request to report the attributes of a request sequence directory, the request sequencing subservice shall generate a single request sequence directory attributes report that includes the related notification.

6.21.12 Subservice observables

a. The following observables shall be defined for the request sequencing subservice:

1. the list of request sequence identifiers, reference of the request if activated and associated execution status, in an array of size corresponding to the maximum number of request sequences that can be contemporaneously loaded at any time;

6.22 ST[22] position-based scheduling

6.22.1 Scope

6.22.1.1 General

The (orbit) position-based scheduling service type provides the capability to command on-board application processes using requests pre-loaded on-board the spacecraft and released when the spacecraft reaches the associated position on the orbit. The service type does not specify how the orbit positions are determined; this is done when tailoring the service to the mission.

The position-based scheduling service type defines a single standardized subservice type, i.e. the position-based scheduling subservice type.

6.22.1.2 Position-based scheduling subservice

The position-based scheduling subservice type includes the capability to maintain an on-board position-based schedule of requests and to ensure the release of those requests at the associated orbit positions.

This provides an extension of the ground monitoring and control. As such, the application process that executes a request released by the position-based scheduling subservice directly sends the request verification reports, if any, to the source identified by the source identifier specified in the request. The release of a request by the subservice is not conditional on the successful or unsuccessful execution of earlier requests released by the subservice.

An entry in the position-based schedule is usually deleted once the related request is released. However, the position-based scheduling subservice type provides the optional concept of persistent scheduling, which can be used to retain an entry in the schedule so that it can be reused on a later orbit.

The position-based scheduling subservice type provides the optional concept of sub-schedules. If the position-based scheduling subservice supports sub-schedules, each request in the position-based schedule is associated to a sub-schedule. Each sub-schedule consists of a sequence of position-tagged requests that perform a coherent on-board operation. If a sub-schedule has no requests with persistent scheduling status, then once the operation is completed, the sub-schedule has no further reason to exist. Therefore, sub-schedules are automatically created when used and deleted when empty. The position-based scheduling subservice type includes the capability for enabling and disabling the execution of each sub-schedule.

The position-based scheduling subservice type also provides the optional concept of groups. If the position-based scheduling subservice supports groups, each request in the position-based schedule is associated to a group. The position-based scheduling subservice type includes the capability for enabling and disabling the execution of grouped requests, independently of the application processes they are released to and of the sub-schedules they belong to. Groups are typically related to spacecraft entities (e.g. hardware or software). Groups can be created and deleted by request and can exist even if empty. They can be used, for example, to group all requests associated to a specific instrument and disable their release when the conditions for their execution are not fulfilled, while other

requests for the same application process are associated to a different group and enabled for release.

The term "scheduled activity" is used in this service to refer to each entry of the position-based schedule. A scheduled activity consists of:

- scheduling data, e.g. the identifier of the sub-schedule, the identifier of the group, the release position;
- the request that is scheduled for later release.

Each scheduled activity is identified by the identifier of the request that is scheduled for later release.

6.22.2 Service layout

6.22.2.1 Subservice

6.22.2.1.1 Position-based scheduling subservice

a. Each position-based scheduling service shall contain at least one position-based scheduling subservice.

6.22.2.2 Application process

a. Each application process shall host at most one position-based scheduling subservice provider.

6.22.3 Accessibility

6.22.3.1 Application process

a. The list of application processes that can be addressed by the position-based scheduling subservice when releasing requests shall be declared when specifying that subservice.

NOTE 1 This Standard assumes that all requests of addressable application processes can be used by the position-based scheduling subservice. The application process that hosts the position-based scheduling subservice is, by nature, an addressable application process.

NOTE 2 When the position-based scheduling subservice releases a request, the request is processed by an executing service, indicated by the service type and the application process identifier within the request.

NOTE 3 Requests released by the position-based scheduling subservice are not generated by that subservice but by the source that initiated the insert activities into schedule request, i.e. the original source.

6.22.4 Determining orbit positions

a. Each position tag used to specify a position shall consist of an orbit number and the position on that orbit.

NOTE 1 The orbit number never wraps around during a mission, while the orbit position is cyclic.

NOTE 2 The orbit number increments autonomously on-board and can be set to a specific value using the request to set the orbit number, refer to clause 6.22.6.4.

b. The position within the orbit shall be specified using the angle measured in the plane of the osculating inertial orbit starting from the intersection with the Earth Fixed Equatorial Plane.

NOTE The angle is defined in the inertial and instantaneous orbital plane between the Earth equator and the desired location on orbit for the command execution. That angle is positive in the flight direction and measured from the ascending intersection of this orbital plane and the Earth equator. The angle is a expressed in degrees between 0 and 360 degrees.

c. The delta position to apply to the position tag to determine a new position tag shall consist of:

1. the shift of the orbit number
2. the shift of the position in that orbit

NOTE 1 the orbit number shift is expressed as a signed integer.

NOTE 2 the orbit position shift is expressed in degrees, from 0 to 360.

d. The angle accuracy that the position-based scheduling subservice uses to determine the orbit position shall be declared when specifying that subservice.

NOTE This Standard does not further elaborate on the algorithm used to compute this angle, e.g. the accuracy to use remains mission-specific.

6.22.5 Persistent scheduling

a. Whether the position-based scheduling subservice provides the capability for persistent scheduling shall be declared when specifying that subservice.

b. If the position-based scheduling subservice provides the capability for persistent scheduling, the subservice shall maintain, for each scheduled activity, a status indicating whether that scheduled activity is persistent or non-persistent.

NOTE 1 This status is named "activity persistency status".

NOTE 2 If the activity persistency status of a scheduled activity is non-persistent, then once the request contained in that activity is released, the scheduled activity is deleted from the schedule. If the capability for persistent scheduling is not provided, all scheduled activities are handled in this way.

NOTE 3 If the activity persistency status of a scheduled activity is persistent, then after the request associated with that activity is released, the scheduled activity remains in the schedule. It can subsequently be released again or deleted.

6.22.6 Managing the position-based schedule

6.22.6.1 Capability

a. Whether the position-based scheduling subservice supports the capability for managing sub-schedules shall be declared when specifying that subservice.

NOTE See clause 6.22.7.

b. Whether the position-based scheduling subservice supports the capability for managing groups specified shall be declared when specifying that subservice.

NOTE See clause 6.22.8.

6.22.6.2 General

a. Each scheduled activity definition shall consist of:

1. the request;
2. a position tag containing the release position for the request;
3. if the position-based scheduling subservice provides the capability for persistent scheduling:
   (a) the activity persistency status that is either "persistent" or "non-persistent";
   (b) the persistent activity periodicity expressed as an integer number of orbits;
4. if sub-schedules are supported, the identifier of the sub-schedule to which that scheduled activity is associated;
5. if groups are supported, the identifier of the group to which that scheduled activity is associated.

NOTE 1 For item 2, refer to clause 6.22.4

NOTE 2 For item 3, refer to clause 6.22.5.

NOTE 3 For item 4, refer to requirement 6.22.6.1a.

NOTE 4 For item 5, refer requirement 6.22.6.1b.

b. Each scheduled activity definition shall be identified by a scheduled activity identifier that corresponds to the identifier of the request contained in that definition.

NOTE For the request identifier, refer to requirement 5.4.11.2.1.c.

c. The maximum number of scheduled activity definitions that the position-based scheduling subservice can insert within the position-based schedule and contemporaneously process at any time shall be declared when specifying that subservice.

NOTE 1 This Standard assumes that the resources allocated to the position-based scheduling subservice are sufficient to support this maximum number of scheduled activities independently of the size of the requests they contain.

NOTE 2 The subservice does not constrain the number of scheduled activities having the same release position.

d. The position margin that the position-based scheduling subservice uses shall be declared when specifying that subservice.

NOTE The position margin is present in order to ensure the consistency and operability of the schedule at any time. Inserting activities or position-shifting them can only be performed if the release position of these activities is greater than or equal to the current position plus a position margin.

e. The maximum activity release positions to release a request contained within a scheduled activity definition once the release position specified in that scheduled activity definition has occurred shall be declared when specifying that subservice.

NOTE The number of activities that the subservice can release at a given position depends on the time taken onboard to activate the execution of any activity and the maximum delta position expressed.

f. The position resolution used by the position-based scheduling subservice shall be declared when specifying that subservice.

g. The configuration policy to apply when starting and restarting the position-based scheduling subservice shall be declared when specifying the subservice.

NOTE 1 this includes if a sub-schedule is automatically disabled or not after the last scheduled activity belonging to that sub-schedule is released

NOTE 2 this includes if a group is automatically disabled or not after the last scheduled activity belonging to that group is released

NOTE 3 this covers the ability of loading an alternative content which is necessary in order to address the non-nominal situations under which the initialization takes place, e.g. a safe-mode.

NOTE 4 in case of any specific initialisation needed at PUS level, it will be stated in a dedicated requirement

6.22.6.3 Controlling the position-based schedule execution function

6.22.6.3.1 Status

a. The position-based scheduling subservice shall maintain a status indicating whether the overall position-based schedule execution function is enabled or disabled.

NOTE This status is named "position-based schedule execution function status".

b. When starting the position-based scheduling subservice, the position-based schedule execution function status shall be set to "disabled".

6.22.6.3.2 Enable the position-based schedule execution function

a. The position-based scheduling subservice shall provide the capability to enable the position-based schedule execution function.

NOTE 1 The corresponding requests are of message type "TC[22,1] enable the position-based schedule execution function".

NOTE 2 For the capability to disable the position-based schedule execution function, refer to clause 6.22.6.3.3.

b. Each request to enable the position-based schedule execution function shall contain exactly one instruction to enable the position-based schedule execution function.

NOTE The instructions to enable the position-based schedule execution function contain no argument.

c. For each valid instruction to enable the position-based schedule execution function, the position-based scheduling subservice shall:

1. set the position-based schedule execution function status to "enabled".

NOTE Enabling the position-based schedule execution function does not depend on the presence of scheduled activities in the schedule.

6.22.6.3.3 Disable the position-based schedule execution function

a. The position-based scheduling subservice shall provide the capability to disable the position-based schedule execution function.

NOTE 1 The corresponding requests are of message type "TC[22,2] disable the position-based schedule execution function".

NOTE 2 For the capability to enable the position-based schedule execution function, refer to clause 6.22.6.3.2.

b. Each request to disable the position-based schedule execution function shall contain exactly one instruction to disable the position-based schedule execution function.

NOTE The instructions to disable the position-based schedule execution function contain no argument.

c. For each valid instruction to disable the position-based schedule execution function, the position-based scheduling subservice shall:

1. set the position-based schedule execution function status to "disabled".

NOTE Disabling the position-based schedule execution function does not depend on the presence of scheduled activities in the schedule.

6.22.6.4 Set the orbit number

a. The position-based scheduling subservice capability to set the orbit number shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[22,28] set the orbit number".

b. Each request to set the orbit number shall contain exactly one instruction to set the orbit number.

c. Each instruction to set the orbit number shall contain:

1. the orbit number.

d. For each valid instruction to set the orbit number, the position-based scheduling subservice shall:

1. at the end of the current orbit, set the new orbit number to the orbit number specified in the instruction.

NOTE This Standard does not further elaborate on how the orbit number increments on-board.

6.22.6.5 Reset the position-based schedule

a. The position-based scheduling subservice shall provide the capability to reset the position-based schedule.

NOTE The corresponding requests are of message type "TC[22,3] reset the position-based schedule".

b. Each request to reset the position-based schedule shall contain exactly one instruction to reset the position-based schedule.

NOTE The instructions to reset the position-based schedule contain no argument.

c. For each valid instruction to reset the position-based schedule, the position-based scheduling subservice shall:

1. set the position-based schedule execution function status to "disabled";
2. delete all scheduled activities from the schedule;
3. if sub-schedules are supported, enable all sub-schedules;
4. if groups are supported, enable all groups.

6.22.6.6 Insert activities into the position-based schedule

a. The position-based scheduling subservice shall provide the capability to insert activities into the position-based schedule.

NOTE 1 The corresponding requests are of message type "TC[22,4] insert activities into the position-based schedule".

NOTE 2 Each valid instruction to insert an activity into the position-based schedule results in the creation of a new scheduled activity in the position-based schedule.

NOTE 3 If sub-schedules are supported, the new scheduled activity is associated to the specified sub-schedule.

NOTE 4 If groups are supported, the new scheduled activity is associated to the specified group.

b. Each request to insert activities into the position-based schedule shall contain:

1. if sub-schedules are supported, a sub-schedule identifier,
2. an ordered list of one or more instructions to insert an activity into the position-based schedule.

NOTE For item 1, refer to requirement 6.22.6.1a.

c. Each instruction to insert an activity into the position-based schedule shall contain:

1. if groups are supported, the group identifier associated to the new scheduled activity;
2. the position tag that specifies the release position for the request in the new scheduled activity;
3. if persistent scheduling is supported:
   (a) the activity persistency status;
   (b) if the activity persistency status is "persistent", the persistent activity periodicity;
4. the request to place in the new scheduled activity.

NOTE 1 For item 1, refer to requirement 6.22.6.1b.

NOTE 2 For item 3, refer to requirement 6.22.5a.

d. The list of verification checks that the position-based scheduling subservice shall perform when accepting a request to place in a new scheduled activity shall be declared when specifying that subservice.

e. The position-based scheduling subservice shall reject any request to insert activities into the position-based schedule if any of the following conditions occurs:

1. the activity cannot be added since the maximum number of scheduled activities that can be contemporaneously processed is already reached;
2. that instruction refers to a group that is unknown;
3. that instruction refers to a sub-schedule that is unknown;

4. that instruction refers to a release position that is not consistent with the planned orbit;
5. the activity is non-persistent and the position tag of the activity is earlier than the position obtained by adding the position-based schedule position margin to the current position;
6. the request contained in that instruction fails any of the verification checks.

NOTE 1 For the maximum number of scheduled activities mentioned in item 1, refer to requirement 6.22.6.2c.

NOTE 2 For item 4, the activity is non-persistent if persistent scheduling is not supported or if the activity persistency status of the activity is "non-persistent".

NOTE 3 For item 5, refer to requirement 1.1.1.1.1a.

f. For each request to insert activities into the position-based schedule that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

g. For each request to insert activities into the position-based schedule that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

h. For each valid instruction to insert an activity into the position-based schedule, the position-based scheduling subservice shall:

1. create a new scheduled activity in the schedule;
2. place the request specified in that instruction into the new scheduled activity;
3. set the position tag of the new scheduled activity to the position tag specified in that instruction;
4. if persistent scheduling is supported, set the activity persistency status of the new scheduled activity to "persistent" or "non-persistent" using the status specified in that instruction;
5. if sub-schedules are supported, associate the new scheduled activity to the sub-schedule specified in the request to insert activities into the position-based schedule;
6. if groups are supported, associate the new scheduled activity to the group specified in that instruction.
7. if the activity is "persistent" and the release orbit position for that activity is earlier than the sum of the current position and the position-based schedule position margin, increment the orbit number of that activity by its persistent activity periodicity as many times as necessary for the release position-tag to be above that margin.

NOTE For item 3, when a new scheduled activity is set to be released at the same release position than some scheduled activities, the “insert into schedule” order corresponds to the release order used by the subservice

6.22.6.7 Schedule execution logic

a. The position-based schedule execution process shall process the scheduled activities in the order of their release positions.

b. When several scheduled activities use the same position time, the position-based scheduling subservice shall release these scheduled activities in the order in which they have been inserted in the position-based schedule.

c. The position-based schedule execution process shall consider that a scheduled activity is disabled if:

1. the position-based schedule execution function is disabled,
2. that scheduled activity is associated to a disabled sub-schedule,
3. that scheduled activity is associated to a disabled group.

d. For each scheduled activity whose release position is reached, the position-based schedule execution process shall, in sequence:

1. if that scheduled activity is not disabled and that scheduled activity can be released within the period expressed by the maximum delta position,
   (a) release the related request;
   (b) else generate an event report specifying that scheduled activity has not been released due to the impossibility to release it within the period expressed by the maximum delta position;
2. if the position-based scheduling sub-service provides the capability for persistent scheduling:
   (a) if the activity persistency status of that scheduled activity is "non-persistent", delete that scheduled activity from the schedule;
   (b) if the activity persistency status of that scheduled activity is "persistent", increment the orbit number of that scheduled activity by its persistent activity periodicity;

   NOTE As implicit by the execution logic, verification by service 1 of the requests in the position schedule with the persistence flag set will be performed every time the request is released.
3. if the position-based scheduling sub-service does not provide the capability for persistent scheduling:
   (a) delete that scheduled activity from the schedule;

   NOTE 1 Items 2 and 3 ensure that scheduled activities that cannot be released when their release position is reached are deleted from the schedule or rescheduled according to their activity persistency status.

   NOTE 2 This Standard does not prescribe any notification to ground when requests are deleted without being released.

   NOTE 3 This Standard does not prescribe the release order of activities scheduled at the same exact position.

e. The execution of any request exposed by this service shall be possible at any time, whether the on-board position-based schedule is currently executing any of its entries or not.

NOTE for example, a request to report the content of a sub-schedule is accepted on-board even is that sub-schedule is executing

6.22.7 Managing position-based sub-schedules

6.22.7.1 Position-based sub-schedules

a. The maximum number of sub-schedules that the position-based scheduling subservice can contemporaneously manage shall be declared when specifying that subservice.

b. For each sub-schedule, the position-based scheduling subservice shall maintain a status indicating whether the schedule execution function for that sub-schedule is enabled or disabled.

NOTE This status is named "sub-schedule status".

6.22.7.2 Enabling and disabling position-based sub-schedules

6.22.7.2.1 Enable position-based sub-schedules

a. The position-based scheduling subservice capability to enable position-based sub-schedules shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,20] enable position-based sub-schedules".

NOTE 2 That capability implies that the subservice provides the capability to disable position-based sub-schedules (refer to clause 6.22.7.2.2).

NOTE 3 For the capability to disable position-based sub-schedules, refer to clause 6.22.7.2.2.

b. Each request to enable position-based sub-schedules shall contain exactly one of:

1. an ordered list of one or more instructions to enable a position-based sub-schedule;
2. a single instruction to enable all position-based sub-schedules.

NOTE The instructions to enable all position-based sub-schedules contain no argument.

c. Each instruction to enable a position-based sub-schedule shall contain:

1. the identifier of the sub-schedule to enable.

d. The position-based scheduling subservice shall reject any request to enable position-based sub-schedules if:

1. that instruction refers to an unknown sub-schedule.

e. For each request to enable position-based sub-schedules that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to enable a position-based sub-schedule, the position-based scheduling subservice shall:

1. set the status of that sub-schedule to "enabled".

g. For each valid instruction to enable all position-based sub-schedules, the position-based scheduling subservice shall:

1. for each sub-schedule maintained by that subservice, set its status to "enabled".

6.22.7.2.2 Disable position-based sub-schedules

a. The position-based scheduling subservice shall provide the capability to disable position-based sub-schedules if the capability to enable position-based sub-schedules is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,21] disable position-based sub-schedules".

NOTE 2 For the capability to enable position-based sub-schedules, refer to clause 6.22.7.2.1.

b. Each request to disable position-based sub-schedules shall contain exactly one of:

1. an ordered list of one or more instructions to disable a position-based sub-schedule;
2. a single instruction to disable all position-based sub-schedules.

NOTE The instructions to disable all position-based sub-schedules contain no argument.

c. Each instruction to disable a position-based sub-schedule shall contain:

1. the identifier of the sub-schedule to disable.

d. The position-based scheduling subservice shall reject any request to disable position-based sub-schedules if:

1. that instruction refers to an unknown sub-schedule.

e. For each request to disable position-based sub-schedules that is rejected, the position-based scheduling subservice shall generate the failed start of execution notification.

f. For each request to disable position-based sub-schedules that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable a position-based sub-schedule, the position-based scheduling subservice shall:

1. set the status of that sub-schedule to "disabled".

h. For each valid instruction to disable all position-based sub-schedules, the position-based scheduling subservice shall:

1. for each sub-schedule maintained by that subservice, set its status to "disabled".

6.22.7.2.3 Report the status of each position-based sub-schedule

a. The position-based scheduling subservice capability to report the status of each position-based sub-schedule shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,18] report the status of each position-based sub-schedule". The responses are data reports of message type "TM[22,19] position-based sub-schedule status report".

NOTE 2 That capability requires that the subservice provides:

- the capability to enable position-based sub-schedules (refer to clause 6.22.7.2.1).

b. Each request to report the status of each position-based sub-schedule shall contain exactly one instruction to report the status of each position-based sub-schedule.

NOTE The instructions to report the status of each position-based sub-schedule contain no argument.

c. For each valid instruction to report the status of each position-based sub-schedule, the position-based scheduling subservice shall:

1. generate, for each position-based sub-schedule managed by that subservice, a single position-based sub-schedule status notification that includes:
   (a) the sub-schedule identifier;
   (b) its status.

d. For each valid request to report the status of each position-based sub-schedule, the position-based scheduling subservice shall generate a single position-based sub-schedule status report that includes all related position-based sub-schedule status notifications.

e. The capability to generate autonomously a single position-based sub-schedule status report on a change of a sub-schedule status or sub-schedule deletion shall be declared when specifying the position-based scheduling subservice.

6.22.8 Managing position-based scheduling groups

6.22.8.1 Position-based scheduling groups

a. The maximum number of groups that the position-based scheduling subservice can contemporaneously manage shall be declared when specifying that subservice.

b. For each group, the position-based scheduling subservice shall maintain a status indicating whether the schedule execution function for that group is enabled or disabled.

NOTE This status is named "group status".

6.22.8.2 Creating and deleting position-based scheduling groups

6.22.8.2.1 Create position-based scheduling groups

a. The position-based scheduling subservice capability to create position-based scheduling groups shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,22] create position-based scheduling groups".

NOTE 2 For the capability to delete position-based scheduling groups, refer to clause 6.22.8.2.2.

b. Each request to create position-based scheduling groups shall contain an ordered list of one or more instructions to create a position-based scheduling group.

c. Each instruction to create a position-based scheduling group shall contain:

1. the identifier of the group;
2. the group status at creation time.

d. The position-based scheduling subservice shall reject any request to create position-based scheduling groups if any of the following conditions occurs:

1. that instruction refers to an already existing group;
2. the maximum number of groups that can be contemporaneously managed is already reached.

e. For each request to create position-based scheduling groups that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to create position-based scheduling groups that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to create a position-based scheduling group, the position-based scheduling subservice shall:

1. add the group identifier to the list of groups maintained by that sub-service;
2. set the group status to the value specified in the instruction.

6.22.8.2.2 Delete position-based scheduling groups

a. The position-based scheduling subservice shall provide the capability to delete position-based scheduling groups if the capability to create position-based scheduling groups is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,23] delete position-based scheduling groups".

NOTE 2 For the capability to create position-based scheduling groups, refer to clause 6.22.8.2.1.

b. Each request to delete position-based scheduling groups shall contain exactly one of:

1. an ordered list of one or more instructions to delete a position-based scheduling group;
2. a single instruction to delete all position-based scheduling groups.

NOTE The instructions to delete all position-based scheduling groups contain no argument.

c. Each instruction to delete a position-based scheduling group shall contain:

1. the identifier of the group to delete.

d. The position-based scheduling subservice shall reject any request to delete position-based scheduling groups if any of the following conditions occurs:

1. that instruction refers to a group that does not exist;
2. that instruction refers to a group that has associated activities.

NOTE If there are scheduled activities associated to a group, the group cannot be deleted.

e. For each request to delete position-based scheduling groups that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to delete position-based scheduling groups that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a position-based scheduling group, the position-based scheduling subservice shall:

1. delete the group identifier from the list of groups maintained by that service.

h. For each valid instruction to delete all position-based scheduling groups, the position-based scheduling subservice shall:

1. for each group maintained by that subservice, delete the identifier of that group;
2. for each group that has associated activities, generate a failed execution notification for that group.

6.22.8.3 Enabling and disabling position-based scheduling groups

6.22.8.3.1 Enable position-based scheduling groups

a. The position-based scheduling subservice shall provide the capability to enable position-based scheduling groups if the capability to create position-based scheduling groups is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,24] enable position-based scheduling groups".

NOTE 2 For the capability to disable position-based scheduling groups, refer to clause 6.22.8.3.2.

b. Each request to enable position-based scheduling groups shall contain exactly one of:

1. an ordered list of one or more instructions to enable a position-based scheduling group;
2. a single instruction to enable all position-based scheduling groups.

NOTE The instructions to enable all position-based scheduling groups contain no argument.

c. Each instruction to enable a position-based scheduling group shall contain:

1. the identifier of the group to enable.

d. The position-based scheduling subservice shall reject any request to enable position-based scheduling groups if:

1. that instruction refers to an unknown group.

e. For each request to enable position-based scheduling groups that is rejected, the position-based scheduling subservice shall generate the failed start of execution notification.

f. For each request to enable position-based scheduling groups that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable a position-based scheduling group, the position-based scheduling subservice shall:

1. set the status of that group to "enabled".

h. For each valid instruction to enable all position-based scheduling groups, the position-based scheduling subservice shall:

1. for each scheduling group maintained by that subservice, set the status of that group to "enabled".

6.22.8.3.2 Disable position-based scheduling groups

a. The position-based scheduling subservice shall provide the capability to disable position-based scheduling groups if the capability to enable position-based scheduling groups is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,25] disable position-based scheduling groups".

NOTE 2 For the capability to enable position-based scheduling groups, refer to clause 6.22.8.3.1.

b. Each request to disable position-based scheduling groups shall contain exactly one of:

1. an ordered list of one or more instructions to disable a position-based scheduling group;
2. a single instruction to disable all position-based scheduling groups.

NOTE The instructions to disable all position-based scheduling groups contain no argument.

c. Each instruction to disable a position-based scheduling group shall contain:

1. the identifier of the group to disable.

d. The position-based scheduling subservice shall reject any request to disable position-based scheduling groups if:

1. that instruction refers to an unknown group.

e. For each request to disable position-based scheduling groups that is rejected, the position-based scheduling subservice shall generate the failed start of execution notification.

f. For each request to disable position-based scheduling groups that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable a position-based scheduling group, the position-based scheduling subservice shall:

1. set the status of that group to "disabled".

h. For each valid instruction to disable all position-based scheduling groups, the position-based scheduling subservice shall:

1. for each scheduling group maintained by that subservice, set the status of that group to "disabled".

6.22.8.3.3 Report the status of each position-based scheduling group

a. The position-based scheduling subservice capability to report the status of each position-based scheduling group shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,26] report the status of each position-based scheduling group". The responses are data reports of message type "TM[22,27] position-based scheduling group status report".

NOTE 2 That capability requires the capability for that subservice to create position-based scheduling groups, refer to clause 6.22.8.2.1.

b. Each request to report the status of each position-based scheduling group shall contain exactly one instruction to report the status of each position-based scheduling group.

NOTE The instructions to report the status of each position-based scheduling group contain no argument.

c. For each valid instruction to report the status of a position-based scheduling group, the position-based scheduling subservice shall:

1. for each group managed by that subservice, generate a single position-based scheduling group status notification that includes:

(a) the group identifier;

(b) its status.

d. For each valid request to report the status of each position-based scheduling group, the position-based scheduling subservice shall generate a single position-based scheduling group status report that includes, for each scheduling group maintained by that subservice, the related position-based scheduling group status notification.

e. The capability to generate autonomously a single position-based sub-schedule status report on a change of a group status or group deletion shall be declared when specifying the position-based scheduling subservice.

6.22.9 Reports of position-based scheduled activities

6.22.9.1 Position-based schedule summary report

a. The position-based scheduling subservice shall provide the capability to generate position-based schedule summary reports if any of the capabilities to summary-report scheduled activities is provided by that subservice.

NOTE 1 The corresponding reports are data reports of message type "TM[22,13] position-based schedule summary report".

NOTE 2 The capabilities to summary-report scheduled activities are:

- the capability to summary-report all position-based scheduled activities (refer to clause 6.22.10.3);
- the capability to summary-report position-based scheduled activities identified by request identifier (refer to clause 6.22.11.4);
- the capability to summary-report the position-based scheduled activities identified by a filter (refer to clause 6.22.12.5).

b. Each position-based schedule summary report shall contain, for each scheduled activity to summary report, a notification consisting of:

1. if sub-schedules are supported, the identifier of the sub-schedule;
2. if groups are supported, the identifier of the group;
3. the scheduled release position;
4. if persistent scheduling is supported:
   (a) the activity persistency status;
   (b) if the activity persistency status is "persistent", the persistent activity periodicity;
5. the identifier of the related request consisting of:
   (a) its source identifier;
   (b) its application process identifier;
   (c) its sequence count.

NOTE 1 The position-based scheduled activities to summary report are determined by one of the requests specified in clauses 6.22.10.3, 6.22.11.4 and 6.22.12.5.

NOTE 2 For item 1, refer to requirement 6.22.6.1a.

NOTE 3 For item 2, refer to requirement 6.22.6.1b.

NOTE 4 For item 4, refer to requirement 6.22.5a.

c. The notifications contained in a position-based schedule summary report shall be ordered according to the release positions of the associated scheduled activities.

6.22.9.2 Position-based schedule detail report

a. The position-based scheduling subservice shall provide the capability to generate position-based schedule detail reports if any of the capabilities to detail-report scheduled activities is provided by that subservice.

NOTE 1 The corresponding reports are data reports of message type "TM[22,10] position-based schedule detail report".

NOTE 2 The capabilities to detail-report scheduled activities are:

- the capability to detail-report all position-based scheduled activities (refer to clause 6.22.10.4);
- the capability to detail-report position-based scheduled activities identified by request identifier (refer to clause 6.22.11.5);
- the capability to detail-report the position-based scheduled activities identified by a filter (refer to clause 6.22.12.6).

b. Each position-based schedule detail report shall contain, for each scheduled activity to detail report, a notification consisting of:

1. if sub-schedules are supported, the identifier of the sub-schedule;
2. if groups are supported, the identifier of the group;
3. the scheduled release position;
4. if persistent scheduling is supported:
   (a) the activity persistency status;
   (b) if the activity persistency status is "persistent", the persistent activity periodicity;
5. the request.

NOTE 1 The position-based scheduled activities to detail report are determined by one of the requests specified in clauses 6.22.10.4, 6.22.11.5 and 6.22.12.6.

NOTE 2 The position-based schedule summary report in clause 6.22.9.1 includes only the identifier of the request associated with the scheduled activity. The position-based schedule detail report specified here

includes the complete request, usually in the form of a telecommand packet.

NOTE 3 For item 1, refer to requirement 6.22.6.1a.

NOTE 4 For item 2, refer to requirement 6.22.6.1b.

NOTE 5 For item 4, refer to requirement 6.22.5a.

c. The notifications contained in a position-based schedule detail report shall be ordered according to the release positions of the associated scheduled activities.

6.22.10 Managing all position-based scheduled activities

6.22.10.1 General

NOTE The capability to reset the position-based schedule specified in clause 6.22.6.5 includes the capability to delete all scheduled activities

6.22.10.2 Position-shift all scheduled activities

a. The position-based scheduling subservice capability to position-shift all scheduled activities shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[22,15] position-shift all scheduled activities".

b. Each request to position-shift all scheduled activities shall contain exactly one instruction to position-shift all scheduled activities.

c. Each instruction to position-shift all scheduled activities shall contain:

1. the delta position.

NOTE for the definition of delta position, refer to 6.22.4.c

d. The position-based scheduling subservice shall reject any request to position-shift all scheduled activities if:

1. the position obtained by adding the delta position to the release position of the earliest non-persistent activity contained within the position-based schedule is earlier than the position obtained by adding the position-based schedule position margin to the current position.

NOTE 1 An activity is non-persistent if persistent scheduling is not supported, or if the activity persistency status of the activity is "non-persistent".

NOTE 2 If the delta position is sufficient to result in a non-persistent scheduled activity with a release position in the past, no activities are position-shifted.

NOTE 3 Shifting a scheduled activity that is persistent never results in a past position tag.

e. For each request to position-shift all scheduled activities that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to position-shift all scheduled activities, the position-based scheduling subservice shall:

1. for each scheduled activity contained within the position-based schedule:

(a) set the release position of that scheduled activity to the sum of the current release position of that activity and the delta position;

(b) if the activity is "persistent" and the new release orbit position for that activity is earlier than the sum of the current position and the position-based schedule position margin, increment the orbit number of that activity by its persistent activity periodicity as many times as necessary for the release position-tag to be above that margin.

6.22.10.3 Summary-report all position-based scheduled activities

a. The position-based scheduling subservice capability to summary-report all position-based scheduled activities shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[22,17] summary-report all position-based scheduled activities". The responses are data reports of message type "TM[22,13] position-based schedule summary report" (refer to clause 6.22.9.1).

b. Each request to summary-report all position-based scheduled activities shall contain exactly one instruction to summary-report all position-based scheduled activities.

NOTE The instructions to summary-report all position-based scheduled activities contain no argument.

c. For each valid instruction to summary-report all position-based scheduled activities, the position-based scheduling subservice shall generate, for each scheduled activity contained within the position-based schedule, a single position-based schedule summary notification.

NOTE The position-based schedule summary notification is defined in clause 6.22.9.1.

d. For each valid request to summary-report all position-based scheduled activities, the position-based scheduling subservice shall generate a single position-based schedule summary report that includes all related position-based schedule summary notifications.

NOTE The position-based schedule summary report is defined in clause 6.22.9.1.

6.22.10.4 Detail-report all position-based scheduled activities

a. The position-based scheduling subservice capability to detail-report all position-based scheduled activities shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[22,16] detail-report all position-based scheduled activities". The responses are data reports of message type "TM[22,10] position-based schedule detail report" (refer to clause 6.22.9.2).

b. Each request to detail-report all position-based scheduled activities shall contain exactly one instruction to detail-report all position-based scheduled activities.

NOTE The instructions to detail-report all position-based scheduled activities contain no argument.

c. For each valid instruction to detail-report all position-based scheduled activities, the position-based scheduling subservice shall generate, for each scheduled activity contained within the schedule, a single position-based schedule detail notification.

NOTE The position-based schedule detail notification is defined in clause 6.22.9.2.

d. For each valid request to detail-report all position-based scheduled activities, the position-based scheduling subservice shall generate a single position-based schedule detail report that includes all related position-based schedule detail notifications.

NOTE The position-based schedule detail report is defined in clause 6.22.9.2.

6.22.11 Managing position-based scheduled activities identified by request identifier

6.22.11.1 General

a. Whether the position-based scheduling subservice supports the identification of scheduled activities by request identifier shall be declared when specifying that subservice.

NOTE That support is required for the capabilities to manage scheduled activities identified by request identifier, i.e.:

- the capability to delete position-based scheduled activities identified by request identifier (refer to clause 6.22.11.2);
- the capability to position-shift scheduled activities identified by request identifier (refer to clause 6.22.11.3);
- the capability to summary-report position-based scheduled activities identified by request identifier (refer to clause 6.22.11.4);

- the capability to detail-report position-based scheduled activities identified by request identifier (refer to clause 6.22.11.5).

6.22.11.2 Delete position-based scheduled activities identified by request identifier

a. The position-based scheduling subservice capability to delete position-based scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,5] delete position-based scheduled activities identified by request identifier".

NOTE 2 That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to requirement 6.22.11.1a).

b. Each request to delete position-based scheduled activities identified by request identifier shall contain an ordered list of one or more instructions to delete a position-based scheduled activity identified by request identifier.

c. Each instruction to delete a position-based scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to delete.

NOTE See requirement 6.22.6.2b.

d. The position-based scheduling subservice shall reject any request to delete position-based scheduled activities identified by request identifier if:

1. that request identifier is unknown.

e. For each request to delete position-based scheduled activities identified by request identifier that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to delete position-based scheduled activities identified by request identifier that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a position-based scheduled activity identified by request identifier, the position-based scheduling subservice shall:

1. delete the scheduled activity corresponding to the request identifier.

6.22.11.3 Position-shift scheduled activities identified by request identifier

a. The position-based scheduling subservice capability to position-shift scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,7] position-shift scheduled activities identified by request identifier".

NOTE 2 That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to requirement 6.22.11.1a).

b. Each request to position-shift scheduled activities identified by request identifier shall contain:

1. a delta position,
2. an ordered list of one or more instructions to position-shift a scheduled activity identified by request identifier.

NOTE 1 The delta position in a request to position-shift scheduled activities identified by request identifier applies to all the instructions in that request.

NOTE 2 For the definition of the delta position, refer to 6.22.4.c

c. Each instruction to position-shift a scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to position-shift.

NOTE See requirement 6.22.6.2b.

d. The position-based scheduling subservice shall reject any request to position-shift scheduled activities identified by request identifier if any of the following conditions occurs:

1. the position obtained by adding the delta position to the release position of the earliest non-persistent activity identified within the request is earlier than the position obtained by adding the position-based schedule position margin to the current position;
2. that request identifier is unknown.

NOTE 1 An activity is non-persistent if persistent scheduling is not supported, or if the activity persistency status of the activity is "non-persistent".

NOTE 2 If the delta position is sufficient to result in a non-persistent scheduled activity with a release position in the past, no activities are position-shifted.

NOTE 3 Shifting a scheduled activity that is persistent never results in a past position tag.

e. For each request to position-shift scheduled activities identified by request identifier that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. The position-based scheduling subservice shall reject any instruction to position-shift a scheduled activity identified by request identifier if:

1. that request identifier is unknown.

g. For each instruction to position-shift a scheduled activity identified by request identifier that is rejected, the position-based scheduling subservice shall generate the failed start of execution notification.

h. For each request to position-shift scheduled activities identified by request identifier that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

i. For each valid instruction to position-shift a scheduled activity identified by request identifier, the position-based scheduling subservice shall:

1. set the release position of the scheduled activity specified in the instruction to the sum of the current release position of that activity and the delta position;

2. if the activity is "persistent" and the new release orbit position for that activity is earlier than the sum of the current position and the position-based schedule position margin, increment the orbit number of that activity by its persistent activity periodicity as many times as necessary for the release position-tag to be above that margin.

6.22.11.4 Summary-report position-based scheduled activities identified by request identifier

a. The position-based scheduling subservice capability to summary-report position-based scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,12] summary-report position-based scheduled activities identified by request identifier". The responses are data reports of message type "TM[22,13] position-based schedule summary report"(refer to clause 6.22.9.1).

NOTE 2 That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to requirement 6.22.11.1a).

b. Each request to summary-report position-based scheduled activities identified by request identifier shall contain an ordered list of one or more instructions to summary-report a position-based scheduled activity identified by request identifier.

c. Each instruction to summary-report a position-based scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to summary report.

NOTE See requirement 6.22.6.2b.

d. The position-based scheduling subservice shall reject any request to summary-report position-based scheduled activities identified by request identifier if:

1. that request identifier is unknown.

e. For each request to summary-report position-based scheduled activities identified by request identifier that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to summary-report position-based scheduled activities identified by request identifier that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to summary-report a position-based scheduled activity identified by request identifier, the position-based scheduling subservice shall generate a single position-based schedule summary notification for that scheduled activity.

NOTE The position-based schedule summary notification is defined in clause 6.22.9.1.

h. For each valid request to summary-report position-based scheduled activities identified by request identifier, the position-based scheduling subservice shall generate a single position-based schedule summary report that contains all related position-based schedule summary notifications.

6.22.11.5 Detail-report position-based scheduled activities identified by request identifier

a. The position-based scheduling subservice capability to detail-report position-based scheduled activities identified by request identifier shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,9] detail-report position-based scheduled activities identified by request identifier". The responses are data reports of message type "TM[22,10] position-based schedule detail report"(refer to clause 6.22.9.2).

NOTE 2 That capability implies that the subservice provides the capability to identify scheduled activities by request identifier (refer to requirement 6.22.11.1a).

b. Each request to detail-report position-based scheduled activities identified by request identifier shall contain an ordered list of one or more instructions to detail-report a position-based scheduled activity identified by request identifier.

c. Each instruction to detail-report a position-based scheduled activity identified by request identifier shall contain:

1. the identifier of the scheduled activity to report.

NOTE See requirement 6.22.6.2b.

d. The position-based scheduling subservice shall reject any request to detail-report position-based scheduled activities identified by request identifier if:

1. that request identifier is unknown.

e. For each request to detail-report position-based scheduled activities identified by request identifier that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each request to detail-report position-based scheduled activities identified by request identifier that contains only valid instructions, the position-based scheduling subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to detail-report a position-based scheduled activity identified by request identifier, the position-based scheduling subservice shall generate a single position-based schedule detail notification for that scheduled activity.

NOTE The position-based schedule detail notification is defined in clause 6.22.9.2.

h. For each valid request to detail-report position-based scheduled activities identified by request identifier, the position-based scheduling subservice shall generate a single position-based schedule detail report that contains all related position-based schedule detail notifications.

6.22.12 Managing the position-based scheduled activities identified by a filter

6.22.12.1 General

a. Whether the position-based scheduling subservice supports selecting scheduled activity using a position-window filtering function shall be declared when specifying that subservice.

NOTE 1 For the position-window filtering function refer to clause 6.22.12.2.

NOTE 2 That support is required for the capabilities to manage time-based scheduled activities identified by a filter, i.e.:

- the capability to delete the position-based scheduled activities identified by a filter (refer to clause 6.22.12.3);
- the capability to position-shift the scheduled activities identified by a filter (refer to clause 6.22.12.4);
- the capability to summary-report the position-based scheduled activities identified by a filter (refer to clause 6.22.12.5);
- the capability to detail-report the position-based scheduled activities identified by a filter (refer to clause 6.22.12.6).

6.22.12.2 Position-window filtering function

6.22.12.2.1 Overview

Each request that uses the position-window filtering function contains a single filter that identifies which scheduled activities are concerned in that request, based on a combination of:

- a position window;
- if sub-schedules are supported, zero or more sub-schedules;
- if groups are supported, zero or more groups.

6.22.12.2.2 Position window filtering

a. The position window filtering function shall support the following filtering mechanisms:
   1. "select all activities scheduled from position tag to position tag",
   2. "select all activities scheduled from position tag",
   3. "select all activities scheduled up to position tag".

b. The set of scheduled activities identified by the "select all activities scheduled from position tag to position tag" filtering mechanism shall be all activities that are scheduled between and including the specified "from position tag" and "to position tag".

c. The set of scheduled activities identified by the "select all activities scheduled from position tag" filtering mechanism shall be all activities that are scheduled at and after that specified "from position tag".

d. The set of scheduled activities identified by the "select all activities scheduled up to position tag" filtering mechanism shall be all activities that are scheduled before and at that specified "to position tag".

6.22.12.2.3 Sub-schedule filtering

a. The set of scheduled activities identified by the sub-schedule filtering function shall be all activities that are associated to that sub-schedule.

b. The sub-schedule filtering function shall ignore any unknown sub-schedule that appears in a filter.

6.22.12.2.4 Group filtering

a. The set of scheduled activities identified by the group filtering function shall be all activities that are associated to that group.

6.22.12.2.5 Overall filtering

a. If the overall filtering only includes the position window filtering, the set of scheduled activities identified by the overall filtering function is the set of scheduled activities identified by the position window filtering function.

b. If the overall filtering includes both the position window filtering and the sub-schedule filtering, the set of scheduled activities identified by the overall filtering function is the scheduled activities that result from the intersection of the sets of scheduled activities:

1. identified by the position window filtering function;
2. identified by the sub-schedule filtering function.

NOTE The set of scheduled activities identified by the sub-schedule filtering function consists of the sum of all activities that are associated to the specified sub-schedules. Unknown sub-schedules are ignored.

c. If the overall filtering includes both the position window filtering and the group filtering, the set of scheduled activities identified by the overall filtering function is the scheduled activities that result from the intersection of the sets of scheduled activities:

1. identified by the position window filtering function;
2. identified by the group filtering function.

NOTE The set of scheduled activities identified by the group filtering function consists of the sum of all activities that are associated to the specified groups.

d. If the overall filtering includes the position window filtering, the sub-schedule filtering and the group filtering, the set of scheduled activities identified by the overall filtering function is the scheduled activities that result from the intersection of the sets of scheduled activities:

1. identified by the position window filtering function;
2. identified by the sub-schedule filtering function;
3. identified by the group filtering function.

6.22.12.3 Delete the position-based scheduled activities identified by a filter

a. The position-based scheduling subservice capability to delete the position-based scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,6] delete the position-based scheduled activities identified by a filter".

NOTE 2 That capability implies that the subservice provides the capability of the position-window filtering function (refer to requirement 6.22.12.1a).

b. Each request to delete the position-based scheduled activities identified by a filter shall contain exactly one instruction to delete the position-based scheduled activities identified by a filter.

c. Each instruction to delete the position-based scheduled activities identified by a filter shall contain:

1. a position window, consisting of:
   (a) the type of the position window that is one of "select all", "from position tag", "to position tag", "from position tag to position tag";
   (b) for "from position tag" and "from position tag to position tag", the from position tag;

(c) for "to position tag" and "from position tag to position tag", the to position tag;

2. if sub-schedules are supported, zero or more sub-schedules;

3. if groups are supported, zero or more groups.

NOTE 1 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.22.12.2.

NOTE 2 For sub-schedule support, refer to requirement 6.22.6.1a.

NOTE 3 For group support, refer to requirement 6.22.6.1b.

d. The position-based scheduling subservice shall reject any request to delete the position-based scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid position window type;

2. that request contains an instruction that refers to a "from position tag" that is greater than a "to position tag".

e. For each request to delete the position-based scheduled activities identified by a filter that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to delete the position-based scheduled activities identified by a filter, the position-based scheduling subservice shall:

1. for each scheduled activity identified by that instruction:

(a) delete that scheduled activity;

6.22.12.4 Position-shift the scheduled activities identified by a filter

a. The position-based scheduling subservice capability to position-shift the scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,8] position-shift the scheduled activities identified by a filter".

NOTE 2 That capability implies that the subservice provides the capability of the position-window filtering function (refer to requirement 6.22.12.1a).

b. Each request to position-shift the scheduled activities identified by a filter shall contain exactly one instruction to position-shift the scheduled activities identified by a filter.

c. Each instruction to position-shift the scheduled activities identified by a filter shall contain:

1. a delta position;

2. the position window, consisting of:

(a) the type of the position window that is one of "select all", "from position tag", "to position tag", "from position tag to position tag";

(b) for "from position tag" and "from position tag to position tag", the from position tag;

(c) for "to position tag" and "from position tag to position tag", the to position tag;

3. if sub-schedules are supported, zero or more sub-schedules;

4. if groups are supported, zero or more groups.

NOTE 1 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.22.12.2.

NOTE 2 For sub-schedule support, refer to requirement 6.22.6.1a.

NOTE 3 For group support, refer to requirement 6.22.6.1b.

NOTE 4 For the definition of the delta position, refer to 6.22.4.c

d. The position-based scheduling subservice shall reject any request to position-shift the scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid position window type;

2. that request contains an instruction that refers to a "from position tag" that is greater than a "to position tag";

3. the position obtained by adding the delta position to the release position of the earliest non-persistent activity identified by the filter is earlier than the position obtained by adding the position-based schedule position margin to the current position.

NOTE 1 For item 3, an activity is non-persistent if persistent scheduling is not supported, or if the activity persistency status of the activity is "non-persistent".

NOTE 2 If the delta position is sufficient to result in a non-persistent scheduled activity with a release position in the past, no activities are position-shifted.

NOTE 3 Shifting a scheduled activity that is persistent never results in a past position tag.

e. For each request to position-shift the scheduled activities identified by a filter that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to position-shift the scheduled activities identified by a filter, the position-based scheduling subservice shall:

1. for each scheduled activity identified by the instruction:

(a) set the release position of the scheduled activity to the sum of the current release position of that activity and the delta position;

(b) if the activity is "persistent" and the new release orbit position for that activity is earlier than the sum of the current position and the position-based schedule position margin, increment the orbit number of that activity by its persistent activity periodicity as many times as necessary for the release position-tag to be above that margin.

6.22.12.5 Summary-report the position-based scheduled activities identified by a filter

a. The position-based scheduling subservice capability to summary-report the position-based scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,14] summary-report the position-based scheduled activities identified by a filter". The responses are data reports of message type "TM[22,13] position-based schedule summary report"(refer to clause 6.22.9.1).

NOTE 2 That capability implies that the subservice provides the capability of the position-window filtering function (refer to requirement 6.22.12.1a).

b. Each request to summary-report the position-based scheduled activities identified by a filter shall contain exactly one instruction to summary-report the position-based scheduled activities identified by a filter.

c. Each instruction to summary-report the position-based scheduled activities identified by a filter shall contain the filter to identify the scheduled activities to report that consists of:

1. a position window, consisting of:
   (a) the type of the position window that is one of "select all", "from position tag", "to position tag", "from position tag to position tag";
   (b) for "from position tag" and "from position tag to position tag", the from position tag;
   (c) for "to position tag" and "from position tag to position tag", the to position tag;
2. if sub-schedules are supported, zero or more sub-schedules;
3. if groups are supported, zero or more groups.

NOTE 1 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.22.12.2.

NOTE 2 For item 2, refer to requirement 6.22.6.1a.

NOTE 3 For item 3, refer to requirement 6.22.6.1b.

d. The position-based scheduling subservice shall reject any request to summary-report the position-based scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid position window type;
2. that request contains an instruction that refers to a "from position tag" that is greater than a "to position tag".

e. For each request to summary-report the position-based scheduled activities identified by a filter that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to summary-report the position-based scheduled activities identified by a filter, the position-based scheduling subservice shall generate, for each identified scheduled activity, a single position-based schedule summary notification.

NOTE The position-based schedule summary notification is defined in clause 6.22.9.1.

g. For each valid request to summary-report the position-based scheduled activities identified by a filter, the position-based scheduling subservice shall generate a single position-based schedule summary report that includes all related position-based schedule summary notifications.

NOTE The position-based schedule summary report is defined in clause 6.22.9.1.

6.22.12.6 Detail-report the position-based scheduled activities identified by a filter

a. The position-based scheduling subservice capability to detail-report the position-based scheduled activities identified by a filter shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[22,11] detail-report the position-based scheduled activities identified by a filter". The responses are data reports of message type "TM[22,10] position-based schedule detail report"(refer to clause 6.22.9.2).

NOTE 2 That capability implies that the subservice provides the capability of the position-window filtering function (refer to requirement 6.22.12.1a).

b. Each request to detail-report the position-based scheduled activities identified by a filter shall contain exactly one instruction to detail-report the position-based scheduled activities identified by a filter.

c. Each instruction to detail-report the position-based scheduled activities identified by a filter shall include the filter to identify the scheduled activities to report that consists of:

1. a position window, consisting of:
   (a) the type of the position window that is one of "select all", "from position tag", "to position tag", "from position tag to position tag";
   (b) for "from position tag" and "from position tag to position tag", the from position tag;

(c) for "to position tag" and "from position tag to position tag", the to position tag;

2. if sub-schedules are supported, zero or more sub-schedules;

3. if groups are supported, zero or more groups.

NOTE 1 For the filtering mechanism, including the interaction of the parts of the filter, refer to clause 6.22.12.2.

NOTE 2 For item 2, refer to requirement 6.22.6.1a.

NOTE 3 For item 3, refer to requirement 6.22.6.1b.

d. The position-based scheduling subservice shall reject any request to detail-report the position-based scheduled activities identified by a filter if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid position window type;

2. that request contains an instruction that refers to a "from position tag" that is greater than a "to position tag".

e. For each request to detail-report the position-based scheduled activities identified by a filter that is rejected, the position-based scheduling subservice shall generate a failed start of execution notification.

f. For each valid instruction to detail-report the position-based scheduled activities identified by a filter, the position-based scheduling subservice shall generate, for each scheduled activity identified by the instruction, a single position-based schedule detail notification.

NOTE The position-based schedule detail notification is defined in clause 6.22.9.2.

g. For each valid request to detail-report the position-based scheduled activities identified by a filter, the position-based scheduling subservice shall generate a single position-based schedule detail report that includes all related position-based schedule detail notifications.

NOTE The position-based schedule detail report is defined in clause 6.22.9.2.

6.22.13 Subservice observables

a. The following observables shall be defined for the position-based scheduling subservice:

1. the current orbit number;

2. the position-based schedule execution function status (enabled or disabled);

3. the current number of scheduled activities in the position-based schedule;

4. if sub-schedules are supported, the current number of sub-schedules;

5. if groups are supported, the current number of groups;

6. the execution position of the first and last schedule entries, independently of their allocation to sub-schedule or group;

7. the current number of persistent activities scheduled in the position-based schedule;
8. the current number of non-persistent activities scheduled in the position-based schedule.

6.23 ST[23] file management

6.23.1 Scope

6.23.1.1 General

The file management service type provides the capability to manage on-board file systems and files.

File systems can either be:

- flat, where directory structures are not supported, or
- structured, where files are stored within directories.

To locate and identify files and directories, this standard introduces the repository path and object name concepts.

The repository path is the logical path to where the object is located. A repository path can either represent:

- a physical path such as a directory path within a file system, or
- a logical path such as a mounted device (e.g. "/mm1" pointing to a mass memory device), a directory within a mounted device (e.g. "/mm1/dir1").

An object can be either a file or a directory. An object name is the unique identifier of that object within a repository. The combination of repository path and object name uniquely identifies an object at mission level and is named the object path (i.e. file path or directory path).

The file management service is not concerned with the contents of the files that it manages.

The file management service type defines two standardized subservice types, i.e.:

- the file handling subservice type;
- the file copy subservice type.

6.23.1.2 File handling subservice

The file handling subservice type provides capabilities for interfacing to the on-board file handling system, for file and directory management operations.

6.23.1.3 File copy subservice

The file copy subservice type provides capabilities for copying or moving files within a file system or between different systems. For example, the capabilities include:

- copying a file from an on-ground file system to an on-board file system;

- copying a file from an on-board file system to a ground file system;
- controlling a file copy operation by suspending, resuming or aborting it.

This Standard assumes the presence of a dedicated file transfer layer, e.g. the CCSDS file delivery protocol, to copy files between the ground and the space systems but does not standardize the corresponding protocol.

6.23.2 Service layout

6.23.2.1 Subservice

6.23.2.1.1 File handling subservice

a. Each file management service shall contain exactly one file handling subservice.

6.23.2.1.2 File copy subservice

a. Each file management service shall contain at most one file copy subservice.

6.23.2.2 Application process

a. For each file management service that contains both, a file handling subservice and a file copy subservice, the two subservice providers of that service shall be hosted by the same application process

6.23.3 file systems

6.23.3.1 Overview

File management service access to a file system includes, for example, creating and deleting files and reading and changing file attributes. If the file system is structured, it also includes creating and deleting directories.

The file management service provides an interface to the on-board file handling system. The extent of the access by the file management service to an on-board file system is constrained by the facilities provided by the related file handling system.

File management service requests typically include arguments specifying files or directories in an on-board file system. This Standard does not specify how the validation of such arguments is shared between the file management service and the related on-board file handling system. Therefore, in the specifications for the handling of the service requests, the validation of such arguments is specified without detail.

The specification of the argument validation performed by the on-board file handling system is outside the scope of this Standard. However, this Standard assumes that the on-board file handling system can detect and react to typical errors, such as:

- the attempts to create files that already exist;

- the attempts to delete files that do not exist;
- the attempts to delete files that are in use or protected.

If the file management service detects an error in a request, this results in a failed start of execution. If the error is detected by the file handling system, this results in a failed completion of execution.

The file management service does not have exclusive access to an on-board file system. Other on-board services can also access the file system: for example, the request sequencing subservice can load a request sequence from an on-board file.

Generally, the file management service manages a single on-board file system but it can also manage multiple on-board file systems.

6.23.3.2 Accessibility

6.23.3.2.1 File systems

a. The list of on-board file systems that are accessed by the file management service shall be declared when specifying that service.

NOTE For the on-board file system, refer also to clause 5.4.5.

6.23.3.3 Wildcard characters in an object path

a. For each on-board file system that is accessible to the file management service, the set of wildcard characters recognised by that file system shall be declared when specifying that subservice.

NOTE A wildcard is a special character matching one or more other characters in a repository path or object name.

6.23.3.4 On-board file attributes

6.23.3.4.1 General

a. For each on-board file system, the set of file attributes that the file management service can read shall be declared when specifying that subservice.

NOTE For the list of file attributes supported by the on-board file system, refer to requirement 5.4.5.

b. For each on-board file system, the set of file attributes that the file management service can set shall be declared when specifying that subservice.

6.23.3.4.2 Minimum capability

a. The file management service shall have access to the size in bytes of any file.

6.23.3.4.3 Additional capability

a. The file management service shall have access to the locking status of any file located in file systems that support locking.

NOTE Refer to requirement 5.4.5e.

6.23.4 File handling subservice

6.23.4.1 Creating and deleting files

6.23.4.1.1 Create a file

a. The file handling subservice shall provide the capability to create a file.

NOTE 1 The corresponding requests are of message type "TC[23,1] create a file".

NOTE 2 For the capability to delete a file, refer to clause 6.23.4.1.2.

b. Each request to create a file shall contain exactly one instruction to create a file.

c. Each instruction to create a file shall contain:

1. the object path of the file;
2. if the file system does not support files with unbounded size, the maximum size of the file in bytes;
3. if locking is supported, the file locking status;
4. any additional file attributes supported by the file handling subservice at creation time.

NOTE 1 For item 2, refer to requirement 6.23.3.4.2a.

NOTE 2 For item 3, refer to requirement 6.23.3.4.3a.

NOTE 3 For item 4, refer to requirement 6.23.3.4.1b.

d. The file handling subservice shall reject any request to create a file if any of the following conditions occurs:

1. that request contains an instruction that refers to an object path that is invalid;
2. that request contains an instruction that specifies a maximum size that is invalid.

e. For each request to create a file that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to create a file, the file handling subservice shall:

1. request the underlying file system to create the file referred to by that instruction;
2. if the underlying file system reports an error in creating that file, generate a failed "completion of execution" notification.

6.23.4.1.2 Delete a file

a. The file handling subservice shall provide the capability to delete a file.

NOTE 1 The corresponding requests are of message type "TC[23,2] delete a file".

NOTE 2 If the file is locked, deletion of the file is prevented. See clause 6.23.4.3.

NOTE 3 For the capability to create a file, refer to clause 6.23.4.1.1.

b. Each request to delete a file shall contain exactly one instruction to delete a file.

c. Each instruction to delete a file shall contain:

1. the object path of the file.

d. The file handling subservice shall reject any request to delete a file if any of the following conditions occurs:

1. that request contains an instruction that refers to an object path that is invalid;

2. that request contains an instruction that refers to a repository path or directory path that contains one or more wildcard characters that are recognised by the file system.

NOTE The delete a file request by means of wildcard characters can be used to delete multiple files only when applied to the file name and inside a single directory

3. that request contains an instruction that refers to a locked file

e. For each request to delete a file that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to delete a file, the file handling subservice shall:

1. request the underlying file system to delete the file referred to by that instruction;

2. if the underlying file system reports an error in deleting the file, generate a failed "completion of execution" notification.

6.23.4.2 Report the attributes of a file

a. The file handling subservice shall provide the capability to report the attributes of a file.

NOTE 1 The corresponding requests are of message type "TC[23,3] report the attributes of a file". The responses are data reports of message type "TM[23,4] file attribute report".

NOTE 2 The file attributes to report are those mentioned in requirement 6.23.3.4.1a.

b. Each request to report the attributes of a file shall contain exactly one instruction to report the attributes of a file.

c. Each instruction to report the attributes of a file shall contain:

1. the object path of the file.

d. The file handling subservice shall reject any request to report the attributes of a file if:

1. that request contains an instruction that refers to an object path that is invalid.

e. For each request to report the attributes of a file that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to report the attributes of a file, the file handling subservice shall:

1. request the underlying file system to provide the attributes of the file referred to by that instruction;
2. if the underlying file system reports an error in providing the attributes of the file, generate a failed completion of execution notification.
3. if no error is reported by the underlying file system, generate a single file attribute notification that includes:
   (a) the file path;
   (b) the file size;
   (c) if the file system supports locking files, the file locked status;
   (d) any additional file attributes supported by the file handling subservice.

   NOTE 1 For item 3(c), refer to requirement 6.23.3.4.3a.

   NOTE 2 For item 3(d), refer to requirement 6.23.3.4.1b.

g. For each valid request to report the attributes of a file, the file handling subservice shall generate a single file attribute report that includes the related file attribute notification.

6.23.4.3 File access protection

6.23.4.3.1 Lock a file

a. The file handling subservice capability to lock a file shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,5] lock a file".

NOTE 2 Locking a file in an on-board file system means that the file is read-only. This implies that the related file handling system provides write protection for a locked file and that it prevents any operations to delete or move a locked file

NOTE 3 File handling systems generally also provide protection to a file while the file is open.

NOTE 4 No change can be done on a file that is locked except unlocking it.

NOTE 5 For a file system that supports file locking, the file management service has access to the locking status of any file. See requirement 6.23.3.4.3a.

NOTE 6 For the capability to unlock a file, refer to clause 6.23.4.3.2.

b. Each request to lock a file shall contain exactly one instruction to lock a file.

c. Each instruction to lock a file shall contain:

1. the object path of the file.

d. The file handling subservice shall reject any request to lock a file if any of the following conditions occurs:
   1. that request contains an instruction that refers to an object path that is invalid;
   2. that request contains an instruction that refers to an object path that contains one or more wildcard characters that are recognised by the file system.

      NOTE The lock a file request cannot be used to lock multiple files by means of wildcard characters.

e. For each request to lock a file that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to lock a file, the file handling subservice shall:
   1. request the underlying file system to lock the file referred to by that instruction;
   2. if the underlying file system reports an error in locking the file, generate a failed "completion of execution" notification.

6.23.4.3.2 Unlock a file

a. The file handling subservice shall provide the capability to unlock a file if the capability to lock a file is provided by that subservice.

   NOTE 1 The corresponding requests are of message type "TC[23,6] unlock a file".

   NOTE 2 For the capability to lock a file, refer to clause 6.23.4.3.1.

b. Each request to unlock a file shall contain exactly one instruction to unlock a file.

c. Each instruction to unlock a file shall contain:
   1. the object path of the file.

d. The file handling subservice shall reject any request to unlock a file if any of the following conditions occurs:
   1. that request contains an instruction that refers to an object path that is invalid;
   2. that request contains an instruction that refers to an object path that contains one or more wildcard characters that are recognised by the file system.

      NOTE The request to unlock a file cannot be used to unlock multiple files by means of wildcard characters.

e. For each request to unlock a file that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to unlock a file, the file handling subservice shall:
   1. request the underlying file system to unlock the file referred to by that instruction;
   2. if the underlying file system reports an error in unlocking the file, generate a failed "completion of execution" notification.

6.23.4.4 Find files

a. The file handling subservice capability to find files shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,7] find files". The responses are data reports of message type "TM[23,8] found files report".

NOTE 2 Finding files in an on-board file system implies that the related file handling system finds the files whose names match the search pattern.

NOTE 3 The extent of the search depends on the capabilities of the related file handling system. The search can be restricted to the files in the directory specified in the request, or it can extend to files in all directories below the specified directory.

b. The file handling subservice shall support the use of search patterns containing wildcards.

NOTE A wildcard is a special character matching one or more other characters.

c. The search pattern wildcards supported by the file handling subservice shall be declared when specifying that subservice.

NOTE These wildcards are those used by the underlying file systems.

d. The extent of the search provided by the file handling subservice for finding files shall be declared when specifying that subservice.

NOTE For example, only files local to the selected repository, or recursively within all sub-directories of the repository.

e. Each request to find files shall contain exactly one instruction to find files.

f. Each instruction to find files shall contain:

1. the repository path;
2. the search pattern.

NOTE Wildcards are limited to the search pattern. The find files request cannot be used to search for files in multiple repositories by means of wildcard characters in the repository path.

g. The file handling subservice shall reject any request to find files if any of the following conditions occurs:

1. that request contains an instruction that refers to a repository path that is invalid;
2. that request contains an instruction that refers to a repository path that contains one or more wildcard characters that are recognised by the file system;
3. that request contains an instruction that specifies an invalid search pattern.

h. For each request to find files that is rejected, the file handling subservice shall generate a failed start of execution notification.

i. For each valid instruction to find files, the file handling subservice shall:

1. request the underlying file system to find the files that match the search pattern in the repository referred to by that instruction;
2. if the underlying file system reports an error in finding the files, generate a failed "completion of execution" notification.
3. generate, for each found file, a single found file notification that includes:
   (a) the searched repository path;
   (b) the searched name pattern;
   (c) the list of all matching file paths, if any.

NOTE If no other error is reported, a failure by the underlying file system to find files that match the search pattern is not considered an error.

j. For each valid request to find files, the file handling subservice shall generate a single found files report that includes all related found file notifications.

NOTE If no files have been found as a result of the search, the list of files in the found files report is empty.

6.23.4.5 Checksum a file

a. The file handling subservice capability to checksum a file shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[23,25] checksum a file". The responses are data reports of message type "TM[23,26] checksum report of a file".

b. Each request to checksum a file shall contain exactly one instruction to checksum a file.

c. Each request to checksum a file shall contain:

1. the object path of the file.

d. The file handling subservice shall reject any request to checksum a file if:

1. that request contains an instruction that refers to an object path that is invalid.

e. For each request to checksum a file that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to report the checksum of a file, the file handling subservice shall:

1. calculate the checksum of the file specified by that instruction;
2. generate a single file checksum notification that includes:
   (a) the object path of the file
   (b) the calculated checksum of the file

NOTE the selection of the checksum type is mission-specific, refer to Annex B

g. For each valid request to checksum a file, the file handling subservice shall generate a single file checksum report that includes the related file checksum notification.

6.23.4.6 Managing directories

6.23.4.6.1 Create a directory

a. The file handling subservice capability to create a directory shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,9] create a directory".

NOTE 2 For the capability to delete a directory, refer to clause 6.23.4.6.2.

b. Each request to create a directory shall contain exactly one instruction to create a directory.

c. Each instruction to create a directory shall contain:

1. the object path of the directory.

d. The file handling subservice shall reject any request to create a directory if:

1. that request contains an instruction that refers to an object path that is invalid.

e. For each request to create a directory that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to create a directory, the file handling subservice shall:

1. request the underlying file system to create the directory referred to by that instruction;
2. if the underlying file system reports an error in creating the directory, generate a failed "completion of execution" notification.

6.23.4.6.2 Delete a directory

a. The file handling subservice shall provide the capability to delete a directory if the capability to create a directory is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,10] delete a directory".

NOTE 2 The type of directory deletion depends on the capabilities of the related file handling system. For example, deletion can be restricted to an empty directory, or to a directory with no sub-directories. If the file handling system supports recursive deletion, then deletion extends to all files and directories below the specified directory.

NOTE 3 The presence of a locked file in a directory, or in any directory below it, prevents deletion of the directory.

NOTE 4 For the capability to create a directory, refer to clause 6.23.4.6.1.

b. The type of directory deletion provided by the file handling subservice shall be declared when specifying that subservice.

c. Each request to delete a directory shall contain exactly one instruction to delete a directory.

d. Each instruction to delete a directory shall contain:

1. the object path of the directory.

e. The file handling subservice shall reject any request to delete a directory if any of the following conditions occurs:

1. that request contains an instruction that refers to an object path that is invalid;

2. that request contains an instruction that refers to an object path that contains one or more wildcard characters that are recognised by the file system.

NOTE The delete a directory request cannot be used to delete multiple directories by means of wildcard characters

3. that request contains an instruction that refers to an object path of a protected directory

4. that request contains an instruction that refers to an object path of a directory which is not empty

NOTE a directory is not empty if it contains subdirectories (protected or not) or files (locked or not)

f. For each request to delete a directory that is rejected, the file handling subservice shall generate a failed start of execution notification.

g. For each valid instruction to delete a directory, the file handling subservice shall:

1. request the underlying file system to delete the directory referred to by that instruction;

2. if the underlying file system reports an error in deleting the directory, generate a failed completion of execution notification.

6.23.4.6.3 Rename a directory

a. The file handling subservice shall provide the capability to rename a directory if the capability to create a directory is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,11] rename a directory".

NOTE 2 The presence of a locked file in a directory, or in any directory below it, prevents renaming of the directory.

NOTE 3 For the capability to create a directory, refer to clause 6.23.4.6.1.

b. Whether the file handling subservice supports the renaming of directories shall be declared when specifying that subservice.

c. Each request to rename a directory shall contain exactly one instruction to rename a directory.

d. Each instruction to rename a directory shall contain:

1. the repository path and current directory name of the directory;
2. the new directory name of the directory.

e. The file handling subservice shall reject any request to rename a directory if any of the following conditions occurs:

1. that request contains an instruction that refers to a repository path that is invalid;
2. that request contains an instruction that refers to a current directory name that is invalid;
3. that request contains an instruction that refers to a new directory name that is invalid.

f. For each request to rename a directory that is rejected, the file handling subservice shall generate a failed start of execution notification.

g. For each valid instruction to rename a directory, the file handling subservice shall:

1. request the underlying file system to rename the directory to the new name referred to by that instruction;
2. if the underlying file system reports an error in renaming the directory, generate a failed "completion of execution" notification.

6.23.4.6.4 Enable directory protection

a. The file handling subservice capability to protect a directory shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,27] enable directory protection".

NOTE 2 Enabling the protection of a directory in an on-board file system means that the directory can not be to deleted, moved or renamed. This implies that the related file handling system provides protection against any of these operations.

NOTE 3 For a file system that supports directory protection, the file management service has access to the protection status of any directory.

NOTE 4 For the capability to unprotect a directory, refer to clause 6.23.4.3.2.

b. Each request to enable directory protection shall contain exactly one instruction to enable directory protection.

c. Each instruction to enable directory protection shall contain:

1. the object path of the directory.

d. The file handling subservice shall reject any request to enable directory protection if any of the following conditions occurs:

1. that request contains an instruction that refers to an object path that is invalid;

2. that request contains an instruction that refers to an object path that contains one or more wildcard characters that are recognised by the file system.

NOTE the enable directory protection request cannot be used to protect multiple directories by means of wildcard characters.

e. For each request to enable directory protection that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to enable directory protection, the file handling subservice shall:

1. request the underlying file system to enable the protection the directory referred to by that instruction;

2. if the underlying file system reports an error when enabling the protection of the directory, generate a failed "completion of execution" notification.

6.23.4.6.5 Disable directory protection

a. The file handling subservice capability to disable directory protection shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,28] disable directory protection".

NOTE 2 Disabling the protection of a directory in an on-board file system means that the directory can be deleted, moved or renamed. This implies that the related file handling system provides protection against any of these operations.

NOTE 3 For a file system that supports directory protection, the file management service has access to the protection status of any directory.

NOTE 4 For the capability to enable directory protection, refer to clause 6.23.4.3.2.

b. Each request to disable directory protection shall contain exactly one instruction to disable directory protection.

c. Each instruction to disable directory protection shall contain:

1. the object path of the directory.

d. The file handling subservice shall reject any request to disable directory protection if any of the following conditions occurs:

1. that request contains an instruction that refers to an object path that is invalid;

2. that request contains an instruction that refers to an object path that contains one or more wildcard characters that are recognised by the file system.

NOTE the disable directory protection request cannot be used to disable the protection of multiple directories by means of wildcard characters.

e. For each request to disable directory protection that is rejected, the file handling subservice shall generate a failed start of execution notification.

f. For each valid instruction to disable directory protection, the file handling subservice shall:

1. request the underlying file system to disable the protection the directory referred to by that instruction;

2. if the underlying file system reports an error when disabling the protection of the directory, generate a failed "completion of execution" notification.

6.23.4.7 Report the content of a repository

a. The file handling subservice capability to report the content of a repository shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[23,12] report the content of a repository". The responses are data reports of message type "TM[23,13] repository content report".

b. When reporting repository content, the file handling subservice shall report only those objects that are direct children of the repository specified in the request.

NOTE This request does not report recursively on objects in directories below the directory specified in the request.

c. Each request to report the content of a repository shall contain exactly one instruction to report the content of a repository.

d. Each instruction to report the content of a repository shall contain:

1. the repository path.

e. The file handling subservice shall reject any request to report the content of a repository if any of the following conditions occurs:

1. that request contains an instruction that refers to a repository path that is invalid;

2. that request contains an instruction that refers to a repository path that contains one or more wildcard characters that are recognised by the file system.

NOTE The report the content of a repository request cannot be used to report the contents of multiple directories by use of wildcard characters.

f. For each request to report the content of a repository that is rejected, the file handling subservice shall generate a failed start of execution notification.

g. For each valid instruction to report the content of a repository, the file handling subservice shall:

1. request the underlying file system to provide a list of the objects in the repository referred to by that instruction;
2. if the underlying file system reports an error in providing the list of objects, generate a failed "completion of execution" notification.
3. generate, for each object contained within the repository, a single repository content notification that includes:
   (a) the repository path;
   (b) the object type that is one of "file" or "directory";
   (c) the object name;
   (d) the file attributes if the object type is "file".

NOTE 1 If there are no objects in the repository, the list of objects in the repository content summary report is empty.

NOTE 2 A report from the underlying file system that the repository is empty is not considered an error.

h. For each valid request to report the content of a repository, the file handling subservice shall generate a single repository content report that includes all related repository content notifications.

6.23.4.8 Subservice observables

a. The following observables shall be defined for the file handling subservice:

1. for each file system:
   (a) the available unallocated memory.

6.23.5 File copy subservice

6.23.5.1 File systems

a. When specifying the file copy subservice, the list of file systems that are accessible to that subservice as source, as destination or as both source and destination shall be declared.

NOTE The list contains the on-board file systems and the remote file systems, e.g. ground.

b. A file in a remote file system shall be uniquely identified to the file copy subservice by a remote file path that is the combination of a repository path and a file name.

6.23.5.2 File copy operations

6.23.5.2.1 General

a. Each file copy operation shall have an identifier that is unique during the lifetime of the operation.

NOTE 1 That unique identifier is used in the requests to copy a file and to move a file.

NOTE 2 During the lifetime of the file copy operation, the identifier can be used in other requests, for example, to suspend or abort the operation. It is also used in reports of the status of file copy operations.

6.23.5.2.2 Copy a file

a. The file copy subservice shall provide the capability to copy a file.

NOTE 1 The corresponding requests are of message type "TC[23,14] copy a file".

NOTE 2 The attributes of the created target file (e.g. permissions) are file system and implementation dependent.

b. Each request to copy a file shall contain exactly one instruction to copy a file.

c. Each instruction to copy a file shall contain:

1. the identifier for the file copy operation;
2. the object path of the source file;
3. the object path of the target file.

d. The file copy subservice shall support source file names containing wildcards when copying files.

NOTE If the source file name contains wildcards, the target file name is ignored as the source file name is kept

e. The file copy subservice shall reject any request to copy a file if any of the following conditions occurs:

1. that request contains a file copy operation identifier that is already allocated to another on-going file copy operation;
2. that request contains an instruction that refers to an object path for the source file that is invalid;
3. that request contains an instruction that refers to an object path for the target file that is invalid;
4. that request contains an instruction for which both the source and the target object paths refer to a remote file system.

NOTE The copy a file request cannot be used to copy a file from a remote source to a remote target.

f. For each request to copy a file that is rejected, the file copy subservice shall generate a failed start of execution notification.

g. For each valid instruction to copy a file, the file copy subservice shall:

1. use the file copy operation identifier contained in that instruction as the identifier of the new file copy operation that it starts;
2. from the file paths of the source file and the target file in that instruction, determine the underlying file transfer handler to use for copying the file;
3. request the underlying file transfer handler to copy the source file to the target file.

NOTE The file copy subservice uses an underlying file transfer handler to perform the file copy. For example, this can be a file transfer layer or it can be a capability of a local file system. The choice is implementation dependent and it also depends on the file systems affected by the copy request.

h. For each file copy operation that it starts in response to a file copy request, the file copy subservice shall process each related execution notification that it receives from the underlying file transfer handler.

NOTE For example, if the target file system has insufficient available memory, the underlying file transfer handler fails to copy the file, causing the raising of a failed execution notification.

i. For each file copy successful execution notification that it receives, if the corresponding successful execution verification report has been requested, the file copy subservice shall generate exactly one successful execution verification report of type deduced from the type of the received notification.

j. For each file copy failed execution notification that it receives, the file copy subservice shall generate exactly one failed execution verification report of type deduced from the type of the received notification.

6.23.5.2.3 Move a file

a. The file copy subservice capability to move a file shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,15] move a file".

NOTE 2 The attributes of the created target file (e.g. permissions) are file system and implementation dependent.

b. Each request to move a file shall contain exactly one instruction to move a file.

c. Each instruction to move a file shall contain:

1. the identifier for the file copy operation;
2. the object path of the source file;
3. the object path of the target file.

d. The file copy subservice shall support source file names containing wildcards when moving files.

NOTE If the source file name contains wildcards, the target file name is ignored as the source file name is kept

e. The file copy subservice shall reject any request to move a file if any of the following conditions occurs:

1. that request contains a file copy operation identifier that is already allocated to another on-going file copy operation;
2. that request contains an instruction that refers to an object path for the source file that is invalid;
3. that request contains an instruction that refers to an object path for the target file that is invalid;
4. the source object path and the target object path in the instruction each identify a remote file system.

NOTE The move a file request cannot be used to move a file from a remote source to a remote target.

5. that request contains an instruction that refers to a locked file

f. For each request to move a file that is rejected, the file copy subservice shall generate a failed start of execution notification.

g. For each valid instruction to move a file, the file copy subservice shall:

1. use the file copy operation identifier contained in that instruction as the identifier of the new file copy operation that it starts;
2. from the file paths of the source file and the target file in that instruction, determine the underlying file transfer handler to use for moving the file;
3. request the underlying file transfer handler to move the source file to the target file.

NOTE The file copy subservice uses an underlying file transfer handler to perform the file move. For example, this can be a file transfer layer or it can be a capability of a local file system. The choice is implementation dependent and it also depends on the file systems affected by the move request.

h. For each file copy operation that it starts in response to a file move request, the file copy subservice shall process each related execution notification that it receives from the underlying file transfer handler.

i. For each file move successful execution notification that it receives, if the corresponding successful execution verification report has been requested, the file copy subservice shall generate exactly one successful execution verification report of type deduced from the type of the received notification.

j. For each file move failed execution notification that it receives, the file copy subservice shall generate exactly one failed execution verification report of type deduced from the type of the received notification.

6.23.5.3 Suspending and resuming the file copy operations

6.23.5.3.1 Suspend file copy operations

a. The file copy subservice capability to suspend file copy operations shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,16] suspend file copy operations".

NOTE 2 This capability applies both to the file copy operations that have been initiated using a request to copy a file or to move a file.

NOTE 3 For the capability to resume file copy operations, refer to clause 6.23.5.3.2.

b. Each request to suspend file copy operations shall contain an ordered list of one or more instructions to suspend a file copy operation.

c. Each instruction to suspend a file copy operation shall contain:

1. the identifier of the file copy operation to suspend.

d. The file copy subservice shall reject any request to suspend file copy operations if:

1. the identifier in that instruction refers to a file copy operation that does not exist.

e. For each request to suspend file copy operations that is rejected, the file copy subservice shall generate a failed start of execution notification.

f. For each request to suspend file copy operations that contains only valid instructions, the file copy subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to suspend a file copy operation, the file copy subservice shall:

1. request the associated underlying file transfer handler to suspend the file copy operation.

6.23.5.3.2 Resume file copy operations

a. The file copy subservice shall provide the capability to resume file copy operations if the capability to suspend file copy operations is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,17] resume file copy operations".

NOTE 2 This capability applies to both the file copy operations that have been initiated using a copy a file request and a move a file request.

NOTE 3 For the capability to suspend file copy operations, refer to clause 6.23.5.3.1.

b. Each request to resume file copy operations shall contain an ordered list of one or more instructions to resume a file copy operation.

c. Each instruction to resume a file copy operation shall contain:

1. the identifier of the file copy operation to resume.

d. The file copy subservice shall reject any request to resume file copy operations if:

1. the identifier in that instruction refers to a file copy operation that does not exist.

e. For each request to resume file copy operations that is rejected, the file copy subservice shall generate the failed start of execution notification.

f. For each request to resume file copy operations that contains only valid instructions, the file copy subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to resume a file copy operation, the file copy subservice shall:

1. request the associated underlying file transfer handler to resume the file copy operation.

6.23.5.3.3 Suspend all file copy operations involving a repository path

a. The file copy subservice capability to suspend all file copy operations involving a repository path shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,19] suspend all file copy operations involving a repository path".

NOTE 2 This capability applies both to the file copy operations that have been initiated using a request to copy a file or to move a file.

NOTE 3 This allows for example to suspend all file copies involving ground by specifying the logical path representing the ground, or to suspend all file copy operations by specifying the root path.

NOTE 4 For the capability to resume all file copy operations involving a repository path, refer to clause 6.23.5.3.4.

b. Each request to suspend all file copy operations involving a repository path shall contain exactly one instruction to suspend all file copy operations involving a repository path.

c. Each instruction to suspend all file copy operations involving a repository path shall contain:

1. the repository path.

d. For each valid instruction to suspend all file copy operations involving a repository path, the file copy subservice shall:

1. for each on-going file copy operation, if either the source or the destination of the copy is constrained within the provided repository path, request the associated underlying file transfer handler to suspend that file copy operation.

6.23.5.3.4 Resume all file copy operations involving a repository path

a. The file copy subservice shall provide the capability to resume all file copy operations involving a repository path if the capability to suspend all file copy operations involving a repository path is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,20] resume all file copy operations involving a repository path".

NOTE 2 This capability applies both to the file copy operations that have been initiated using a request to copy a file or to move a file.

NOTE 3 This allows for example to resume all file copies involving ground by specifying the logical path representing the ground, or to resume all file copy operations by specifying the root path.

NOTE 4 For the capability to suspend all file copy operations involving a repository path, refer to clause 6.23.5.3.3.

b. Each request to resume all file copy operations involving a repository path shall contain exactly one instruction to resume all file copy operations involving a repository path.

c. Each instruction to resume all file copy operations involving a repository path shall contain:

1. the repository path.

d. For each valid instruction to resume all file copy operations involving a repository path, the file copy subservice shall:

1. for each file copy operation that is on-hold, request the associated underlying file transfer handler to resume that file copy operation.

6.23.5.4 Abort the file copy operations

6.23.5.4.1 Abort file copy operations

a. The file copy subservice capability to abort file copy operations shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,18] abort file copy operations".

NOTE 2 This capability applies both to the file copy operations that have been initiated using a request to copy a file or to move a file.

b. Each request to abort file copy operations shall contain an ordered list of one or more instructions to abort a file copy operation.

c. Each instruction to abort a file copy operation shall contain:

1. the identifier of the file copy operation to abort.

d. The file copy subservice shall reject any request to abort file copy operations if:

1. the identifier in that instruction refers to a file copy operation that does not exist.

e. For each request to abort a file copy operation that is rejected, the file copy subservice shall generate a failed start of execution notification.

f. For each request to abort file copy operations that contains only valid instructions, the file copy subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to abort a file copy operation, the file copy subservice shall request the associated underlying file transfer handler to abort that file copy operation.

6.23.5.4.2 Abort all file copy operations involving a repository path

a. The file copy subservice capability to abort all file copy operations involving a repository path shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,21] abort all file copy operations involving a repository path".

NOTE 2 This capability applies both to the file copy operations that have been initiated using a request to copy a file or to move a file.

NOTE 3 This allows for example to abort all file copies involving ground by specifying the logical path representing the ground, or to abort all file copy operations by specifying the root path.

b. Each request to abort all file copy operations involving a repository path shall contain exactly one instruction to abort all file copy operations involving a repository path.

c. Each instruction to abort all file copy operations involving a repository path shall contain:

1. the repository path.

d. The file copy subservice shall reject any request to abort all file copy operations involving a repository path if:

1. no file copy operations are on-going.

e. For each request to abort all file copy operations involving a repository path that is rejected, the file copy subservice shall generate a failed start of execution notification.

f. For each valid instruction to abort all file copy operations involving a repository path, the file copy subservice shall:

1. For each on-going file copy operation, request the associated underlying file transfer handler to abort that file copy operation.

6.23.5.5 Periodic file copy status reporting

6.23.5.5.1 General

a. Whether the file copy subservice provides means to report, within the file copy status reports, the progress of the copy operations shall be declared when specifying that subservice.

6.23.5.5.2 Enable the periodic reporting of the file copy status

a. The file copy subservice capability to enable the periodic reporting of the file copy status shall be declared when specifying that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,22] enable the periodic reporting of the file copy status".

NOTE 2 That capability applies both to the file copy operations that have been initiated using a request to copy a file or to move a file.

NOTE 3 For the capability to disable the periodic reporting of the file copy status, refer to clause 6.23.5.5.3.

b. Each request to enable the periodic reporting of the file copy status shall contain exactly one instruction to enable the periodic reporting of the file copy status.

c. Each instruction to enable the periodic reporting of the file copy status shall contain:

1. the periodic reporting interval.

d. The file copy subservice shall reject any request to enable the periodic reporting of the file copy status if:

1. that request contains an instruction that specifies an invalid reporting interval.

e. For each request to enable the periodic reporting of the file copy status that is rejected, the file copy subservice shall generate a failed start of execution notification.

f. For each valid instruction to enable the periodic reporting of the file copy status, the file copy subservice shall:

1. set the file copy status periodic reporting status to "enabled";
2. set the file copy status periodic reporting interval to the specified interval.

6.23.5.5.3 Disable the periodic reporting of the file copy status

a. The file copy subservice shall provide the capability to disable the periodic reporting of the file copy status if the capability to enable the periodic reporting of the file copy status is provided by that subservice.

NOTE 1 The corresponding requests are of message type "TC[23,24] disable the periodic reporting of the file copy status".

NOTE 2 This capability applies both to the file copy operations that have been initiated using a request to copy a file or to move a file.

NOTE 3 For the capability to enable the periodic reporting of the file copy status, refer to clause 6.23.5.5.2.

b. Each request to disable the periodic reporting of the file copy status shall contain exactly one instruction to disable the periodic reporting of the file copy status.

NOTE The instructions to disable the periodic reporting of the file copy status contain no argument.

c. For each valid instruction to disable the periodic reporting of the file copy status, the file copy subservice shall:

1. set the file copy status periodic reporting status to "disabled".

6.23.5.5.4 File copy status report

a. The file copy subservice shall provide the capability for generating the file copy status reports if the capability to enable the periodic reporting of the file copy status is provided by that subservice.

NOTE 1 The corresponding reports are data reports of message type "TM[23,23] file copy status report".

NOTE 2 For the capability to enable the periodic reporting of the file copy status, refer to clause 6.23.5.5.2.

b. Each file copy status report shall contain exactly one file copy status notification for each file copy operation that is on-going.

c. Each file copy status notification shall contain:

1. the identifier of an on-going file copy operation;
2. whether that operation is in-progress, pending waiting on-board resources or on-hold;
3. if the file copy subservice provides means to report the progress of a copy operation, the progress indicator as a percentage of completion.

NOTE For item 3, refer to requirement 6.23.5.5.1a.

d. When the file copy status periodic reporting is enabled, the file copy subservice shall generate exactly one file copy status report at the end of each file copy status periodic reporting interval.

NOTE The enabling of the file copy status periodic reporting results from the execution of a request to enable the periodic reporting of the file copy status, refer to clause 6.23.5.5.2.

6.23.5.6 Subservice observables

a. The following observables shall be defined for the file copy subservice:

1. a flag signalling that at least one file copy operation is in-progress;
2. a flag signalling that at least one file copy operation is on-hold;

3. a flag signalling whether the file copy status reporting is enabled or disabled.

6.24 ST[24] file transfer

6.24.1 Scope

6.24.1.1 File Based Operations (FBO) overview

File Based Operations (FBO) refers to the capability to manage and transfer data (downlink for data generated on board to be transmitted to ground, uplink for data generated on ground to be transmitted on board) via files guaranteeing the integrity of the transferred data without any manual intervention from ground. The autonomous file transmission is especially beneficial for the downlink of telemetry generated on-board which data flow is continuous in time.

This standard supports File Based Operations thriugh the following entities:

- comunication links through which file data transfer is performed
- file management systems handling on-board files, standardized in service 23
- file transaction entities in charge of managing the transmission and reception of files through a file transaction protocol, standardized in service 24 adopting the CCSDS File Delivery Protocol (CDFP) as file transaction protocol (refer to CCSDS 727.0-B-5).
- downlink managers to automatize the downlink of files generated on-board, standardized in service 24
- data sources definitions with the purpose of managing the storage of data produced on-board into files and directories, standardized in service 25

Figure 6.24-1 shows the different entities involved in FBO and the relation between them in a possible configuration:

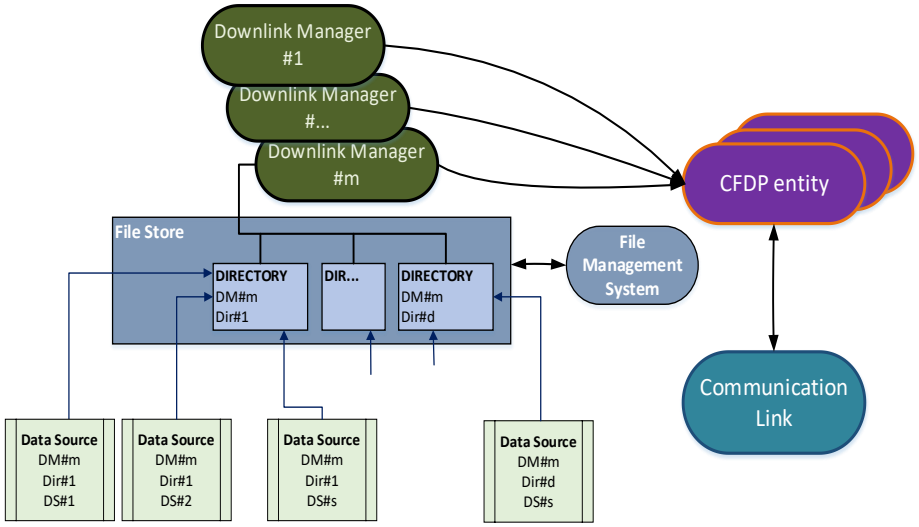

Figure 6.24-1 File Based Operations related entities

The number of available communications links, file transaction entities and downlink managers is mission specific. For simplification the figure above considers only one file transaction entity implementing the CFDP protocol (CFDP entity), one file management system and one communication link but generally:

- CFDP entity to Communication Link relation is n-to-1. This implies that a given CFDP entity will transmit all type of protocol data units through the same communication link (refer to CCSDS 720.1-G-4 1.4.2 for CFDP definitions).
- Downlink manager to CFDP entity relation is n-to-1. This implies that several downlink managers can make use of the same CFDP entity for file downlinking. The policy to synchronize the access to the CFDP entity between downlink managers is mission specific.

As already introduced, in this standard the protocol adopted to perform file transactions between the space and the ground segments (both uplink and downlink) is the CCSDS File Delivery Protocol (CFDP), see normative references in chapter 2.

The downlink manager is the on-board component in charge of handling the downlink of files in an autonomous and controlled manner. Files are organized in directories, and each directory has a priority for downlinking. As telemetry data is generated, files are:

1. automously created inside directories
2. dispatched for transmission by the downlink managers
3. transmitted by the CFDP entity through the communication link

Telemetry data can be generated by different sources. The data source concept allows to identify univocally telemetry packets containing data to be stored into files. The downlink manager concept allows prioritizing the transfer of those files by mapping them into directories.

The File Based Operations concept is deployed in two services: the file transfer service ST[24] and the file data storage service ST[25]. The file transfer service contains two subservices, the file transaction subservice and the file downlink manager subservice. The file data storage service contains two subservices, the file data storage subservice and the PUS-report-type data source control subservice. There are implicit dependencies between these subservices: file data storage subservice implies that at least one file downlink manager subservice is defined in the space system. The file downlink manager subservice implies that at least one file transaction subservice is defined in the space system. These subservices need a file management system deployed on-board which implies that at least one file handling subservice is defined in the space system, already covered in the standardized file management service ST[23].

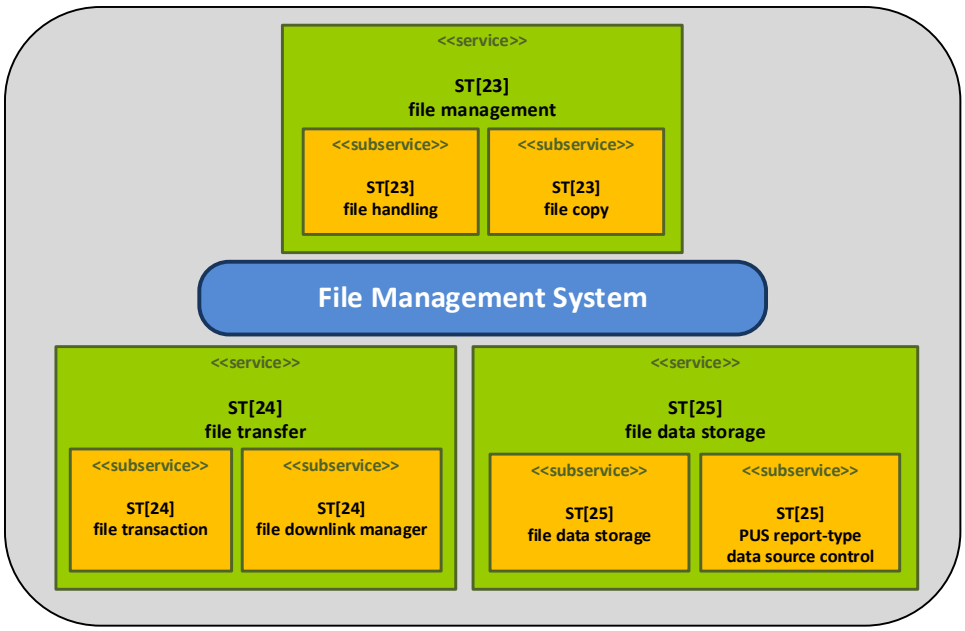

Figure 6.24-2 File Based Operations services, subservices and FMS object

This section covers the file transfer service (service 24), for the file data storage service refer to service 25 in section 6.25.

The file transaction subservice provides capabilities to performe file transactions using the CCSDS file delivery protocol (CFDP) in both directions, i.e. uplink and downlink. CFDP consists of file copy procedures between CFDP entities supported by a communication link, as shown in Figure 6.24-3.

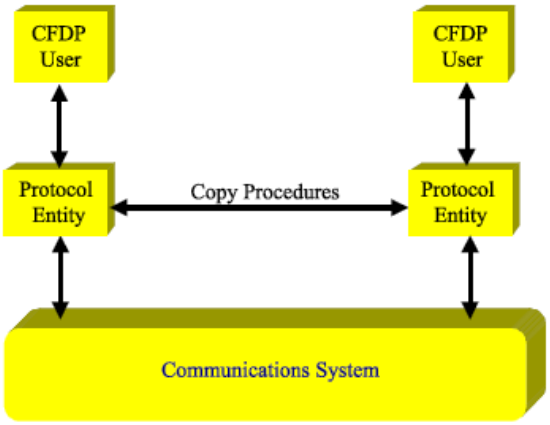

Figure 6.24-3 CFDP file copy concept, from CCSDS 727.0-B-5

Each CFDP entity is identified by a unique entity ID. The source CFDP entity is the entity from which the file is sent (the sender). The destination CFDP entity is the entity to which the file is sent (the receiver).
The local CFDP entity is the entity which is monitored and controlled (i.e., the on-board entity), which could act as sender (downlink) or receiver (uplink), as shown in Figure 6.24-4.

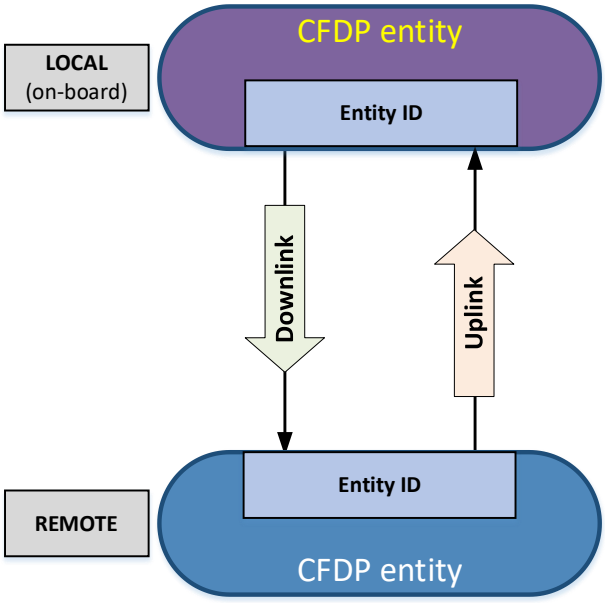

Figure 6.24-4 CFDP entities

The remote CFDP entity could be located on ground or another on-board entity.

The end-to-end transfer of a file between two CFDP entities is performed through a transaction. This standard adopts the following CFDP transaction statuses from CCSDS 727.0-B-5: ACTIVE, TERMINATED. When in ACTIVE status, a transaction can be RUNNING or SUSPENDED, as shown in Figure 6.24-5.

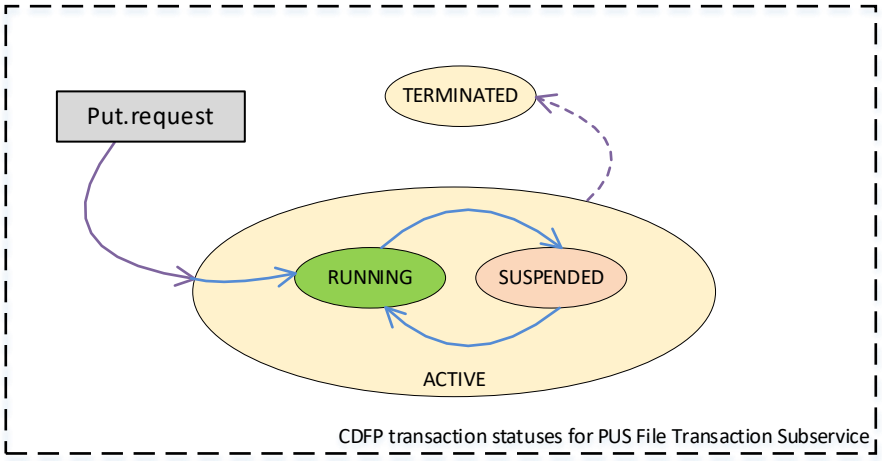

Figure 6.24-5 CFDP transaction statuses

The transaction is initiated with a Put request from the CFDP user to the source CFDP entity becoming ACTIVE, then the transaction can eventually be suspended (SUSPENDED) / resumed (RUNNING). The transaction will become TERMINATED upon succesful file transfer, cancellation or abandonment. Refer to CCSDS 727.0-B-5 for details depending on the transmission mode.

The functional concatenation of the file content and related metadata is called File Delivery Unit (FDU). The term 'metadata' is used to refer to any data exchanged between CFDP entities in addition to file data content, which is needed to support the transaction.

The individual messages transmitted between CFDP entities are called Protocol Data Units (PDUs). CFDP PDUs can be of two types:

- File Data PDUs convey the contents of the files being delivered
- File Directive PDUs convey only metadata and other non-file information that supports the operation of the protocol

Any single transaction entails the transmission and reception of multiple PDUs.

CFDP defines two types/classes of file transfer:

- unreliable transfer (also known as class 1), where the completeness of the transaction cannot be guaranteed (then it is responsibility of the CFDP user)
- reliable transfer (also known as class 2), where the receiver requests the retransmission of lost data with NAKs and notifies the correct reception of all PDUs and the end of file (EOF)

The downlink manager is the entity in charge of handling autonomously the downlink of on-board files according to configurable priority schemes. The downlink manager organizes files into prioritized directories.

The downlink manager is agnostic of the file transaction protocol adopted, i.e. the file downlink manager subservice should be compatible with other file transaction protocols different from CFDP.

The file transfer request rate in the downlink managers should be compatible with the number of concurrent file transactions supported by the on-board implementation and the communication link capabilities. Generated files wait in the corresponding directory for downlinking until a free file transaction is available and allocated to it. The exchange mechanism betweeen downlink managers and CFDP entities is out of the scope of this standard.

The statuses and transitions defined for a downlink manager are shown in Figure 6.24-4.

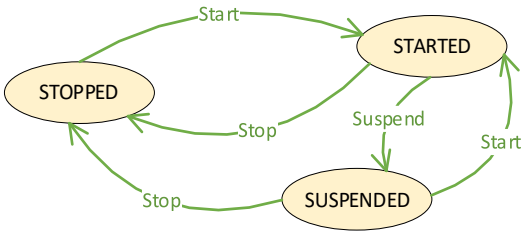

Figure 6.24-6 Downlink manager state machine

The downlink manager default state is mission specific and it can be STARTED, SUSPENDED or STOPPED. Each transition between statuses has an associated action which execution is under downlink manager responsibility, refer to 6.24.5.8, 6.24.5.9 and 6.24.5.10.

The number of downlink managers defined on-board is mission specific.

When a file has been successfully downlinked it can be directly deleted or moved to a back-up directory where it will stay for a configurable maximum time before deletion (refer to service 25 in 6.25).

6.24.1.2 General (file transfer)

The file transfer service type provides the capability to manage the transfer of files between ground and space segments in both directions, uplink and downlink.

The file transfer service type defines the following standardized subservice types:

- file transaction subservice type
- file downlink manager subservice type

6.24.1.3 File transaction subservice

The file transaction subservice type provides capabilities for performing file transactions between space and ground segments using the CCSDS file delivery protocol (CFDP) in both directions, i.e. uplink and downlink.

6.24.1.4 File downlink manager subservice

The file downlink manager subservice type provides capabilities for handling autonomously the downlink of on-board files according to configurable priority schemes supported by directories.

6.24.2 Service layout

6.24.2.1 Subservice

6.24.2.1.1 File transaction subservice

a. Each file transfer service shall contain exactly one file transaction subservice

6.24.2.1.2 File downlink manager subservice

a. Each file transfer service shall contain at most one downlink manager subservice.

6.24.2.2 Application process

a. Each file transaction subservice shall be hosted by exactly one application process

b. Each application process shall host at most one file transaction subservice provider

c. Each file downlink manager subservice shall be hosted by exactly one application process

d. Each application process shall host at most one file downlink manager subservice provider

NOTE 1 a file downlink manager subservice provider can manage more than one downlink manager

NOTE 2 the same application process can host both subservices, the file transaction subservice and the file downlink manager subservice

6.24.3 Accesibility

a. The list of on-board file systems that are accessed by the file transfer service shall be declared when specifying that service

b. The on-board resources allocated to the file transfer service shall be declared when specifying that service

NOTE on-board resources include as a minimum mass memory size and logical partitions

6.24.4 File transaction subservice

6.24.4.1 File transaction protocol

a. The CCSDS File Delivery Protocol shall be adopted by the file transaction subservice as file transaction protocol

b. The file transaction subservice shall provide capabilities for transmitting and receiving files through CFDP transactions

6.24.4.2 CFDP entity

a. the CFDP entity shall be identified by:

1. a unique destination entity ID if it is acting as destination entity within the context of the overall space system
2. a unique source entity ID if it is acting as source entity within the context of the overall space system

NOTE a CFDP entity may act as a source entity, destination entity or both

b. the communication channels associated to the CFDP entity shall be declared when specifying the service

c. the CFDP entity shall be associated to exactly one communication channel for uplink and exactly one communication channel for downlink

d. Each remote CFDP entity configuration shall contain:

1. the default transmission mode

NOTE the default transmission mode is applicable when the CFDP entity is starting the transaction

2. the positive ACK timer interval

NOTE the maximum time interval that the CFDP entity waits prior to reissue a positive acknowledgement directive

3. the negative ACK timer interval

NOTE the maximum time interval that the CFDP entity waits prior to reissue a negative acknowledgement directive

4. the maximum number of expirations of positive ACK timer intervals used by the CFDP entity

5. the maximum number of expirations of negative ACK timer intervals used by the CFDP entity

6. the transaction inactivity time limit

NOTE the time the CFDP entity waits for a PDU reception, prior to the issuance of a fault indication for the transaction

7. the transaction check limit, if it is supported

8. the transaction closure requested flag, if it is supported

NOTE 1 in case of unreliable transfer (class 1), only the corresponding parameters will be considered

NOTE 2 for the remote entity configuration information, refer to CCSDS 727.0-B-5 Table 8-2

NOTE 3 for the local entity configuration information, refer to CCSDS 727.0-B-5 Table 8-1

NOTE 4 the local CFDP entity configuration is static along mission lifetime and defined at mission specification

6.24.4.3 Controlling file transactions with CFDP

a. The spacecraft shall maintain at least one CFDP entity to handle file transactions

NOTE the number of CFDP entities is mission dependent

b. The CFDP procedures supported by each CFDP entity shall be declared when specifying this subservice.

NOTE The list of CFDP procedures can be found in CCSDS 727.0-B-5 4

c. The default transmission mode for each link direction (uplink/downlink) shall be specified when declaring the file transaction subservice.

NOTE 1 transmission mode can be unreliable (class 1) or reliable (class 2)

NOTE 2 in case of reliable transmission mode (class 2), this standard assumes Deferred NAK mode as a minimum, refer to CCSDS 727.0-B-5 2.6.3.1

d. The number of file transactions that can be concurrently in each stage, for transmission and reception, shall be declared for each CFDP entity when specifying this subservice.

e. Whether the transaction check limit is considered shall be declared when specifying this subservice

NOTE transaction check limit is the time that the sending entity waits between the transmission of the end-of-file PDU and the reception of the finished PDU issued by the receiving entity prior to the issuance of a fault indication for the transaction; it applies only to class 1 and it is useful when PDUs may be received out-of-order

f. Whether the transaction closure requested flag is considered shall be declared when specifying this subservice

NOTE transaction closure requested flag only applies to class 1 and is used by the receiving entity to decide whether that receiving entity issues or not a finished PDU at the end of each transaction

g. Each CFDP transaction shall be identified by a unique ID that consists of:

1. the transaction's source CFDP entity ID, and
2. a transaction sequence number that the source entity assigns when it initiates the transaction

NOTE the PFC supporting the transaction sequence number is mission specific

h. For each CFDP transaction managed by the file transaction subservice, the subservice shall maintain a status indicating if the transaction is ACTIVE-RUNNING, ACTIVE-SUSPENDED or TERMINATED.

NOTE These statuses are only used at PUS level and not at CFDP level.

i. For each CFDP transaction that is initiated, the subservice shall generate an informative event report including as a minimum the following information:

2. CFDP source entity ID
3. Transaction Sequence Number
4. CFDP destination entity ID
5. File path
6. File size

NOTE this event can be disabled during nominal operations if not considered needed

j. The configuration policy to apply when starting and restarting the file transaction subservice shall be declared when specifying the subservice.

NOTE the configuration policy includes at least: activation status for each direction and CFDP entities configuration

k. The file transaction subservice shall notify through event reports the errors detected by the CFDP entity

6.24.4.4 Accesibility

a. The file transaction subservice shall be associated to exactly one event reporting subservice

b. Each CFDP entity shall access exactly one on-board file system

NOTE 1 then the CFDP entity can implicitly identify the associated file system

NOTE 2 a fie system can support more than one repository, refer to service 23

6.24.4.5 Start of a file transmission opportunity window

a. The file transaction subservice capability to start of a file transmission opportunity window shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[24,1] start of a file transmission opportunity window".

b. Each request to start of a file transmission opportunity window shall contain exactly one instruction to start of a file transmission opportunity window.

c. Each instruction to start of a file transmission opportunity window shall contain:

1. the identifier of the local CFDP entity for which the file transmission opportunity window starts;
2. the identifier of the remote CFDP entity for which the file transmission opportunity window starts.

d. The file transaction subservice shall reject any request to start a file transmission opportunity window if:

1. the instruction contained in the request refers to an unknown local CFDP entity or an unknown remote CFDP entity

e. For each request to start of a file transmission opportunity window that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each request to start of a file transmission opportunity window that contains a valid instruction to start of a file transmission opportunity window, the file transaction subservice shall notify the specified local CFDP entity of the start of an opportunity window to transmit to the specified remote CFDP entity.

NOTE The way in which this notification is delivered is an implementation choice

6.24.4.6 Stop of a file transmission opportunity window

a. The file transaction subservice shall provide the capability to stop of a file transmission opportunity window if the capability to start of a file transmission opportunity window is provided by that subservice.

NOTE The corresponding requests are of message type "TC[24,2] stop of a file transmission opportunity window".

b. Each request to stop of a file transmission opportunity window shall contain exactly one instruction to stop of a file transmission opportunity window.

c. Each instruction to stop of a file transmission opportunity window shall contain:

1. the identifier of the local CFDP entity for which the file transmission opportunity window stops;
2. the identifier of the remote CFDP entity for which the file transmission opportunity window stops.

d. The file transaction subservice shall reject any request to stop a file transmission opportunity window if:

1. the instruction contained in the request refers to an unknown local CFDP entity or an unknown remote CFDP entity

e. For each request to stop of a file transmission opportunity window that is rejected, the file transaction subservice shall generate a failed start of execution notification.

f. For each request to stop a file transmission opportunity window that contains a valid instruction, the file transaction subservice shall notify the specified local CFDP entity of the end of an opportunity window to transmit to the specified remote CFDP entity.

NOTE The way in which this notification is delivered is an implementation choice

6.24.4.7 Start of a file reception opportunity window

a. The file transaction subservice capability to start of a file reception opportunity window shall be declared when specifying that subservice.

NOTE The corresponding requests are of message type "TC[24,3] start of a file reception opportunity window".

b. Each request to start of a file reception opportunity window shall contain exactly one instruction to start of a file reception opportunity window.

c. Each instruction to start of a file reception opportunity window shall contain:

1. the identifier of the local CFDP entity for which the file reception opportunity window starts;
2. the identifier of the remote CFDP entity for which the file reception opportunity window starts.

d. The file transaction subservice shall reject any request to start a file reception opportunity window if:

1. the instruction contained in the request refers to an unknown local CFDP entity or an unknown remote CFDP entity

e. For each request to start of a file reception opportunity window that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each request to start a file reception opportunity window that contains a valid instruction, the file transaction subservice shall notify the specified local CFDP entity of the start of an opportunity window to receive from the specified remote CFDP entity.

NOTE The way in which this notification is delivered is an implementation choice

6.24.4.8 Stop of a file reception opportunity window

a. The file transaction subservice shall provide the capability to stop of a file reception opportunity window if the capability to start of a file reception opportunity window is provided by that subservice.

NOTE The corresponding requests are of message type "TC[24,4] stop of a file reception opportunity window".

b. Each request to stop of a file reception opportunity window shall contain exactly one instruction to stop of a file reception opportunity window.

c. Each instruction to stop of a file reception opportunity window shall contain:

1. the identifier of the local CFDP entity for which the file reception opportunity window stops;
2. the identifier of the remote CFDP entity for which the file reception opportunity window stops.

d. The file transaction subservice shall reject any request to stop a file reception opportunity window if:

1. the instruction contained in the request refers to an unknown local CFDP entity or an unknown remote CFDP entity

e. For each request to stop of a file reception opportunity window that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each request to stop a file reception opportunity window that contains a valid instruction, the file transaction subservice shall notify the specified local CFDP entity of the end of an opportunity window to receive from the specified remote CFDP entity.

NOTE The way in which this notification is delivered is an implementation choice

6.24.4.9 Suspend a file transaction

a. The file transaction subservice shall provide the capability to suspend a file transaction.

NOTE The corresponding requests are of message type "TC[24,5] suspend a file transaction".

b. Each request to suspend a file transaction shall contain exactly one instruction to suspend a file transaction.

c. Each instruction to suspend a file transaction shall contain:

1. the identifier of the CFDP entity that suspends the file transaction;
2. the identifier of the transaction to suspend, made of:
   (a) the identifier of the source CFDP entity;
   (b) the transaction sequence number.

d. The file transaction subservice shall reject any request to suspend a file transaction if any of the following conditions occur:

1. the instruction contained in the request refers to an invalid CFDP entity;
2. the instruction contained in the request refers to a transaction which is not in ACTIVE status.

NOTE the validity of the transaction identifier is under CFDP entity responsibility

e. For each request to suspend a file transaction that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to suspend a file transaction, the file transaction subservice shall deliver the corresponding suspend request to the CFDP entity

6.24.4.10 Resume a file transaction

a. The file transaction subservice shall provide the capability to resume a file transaction.

NOTE The corresponding requests are of message type "TC[24,6] resume a file transaction".

NOTE For the capability to suspend a file transaction, refer to clause 6.24.4.9.

b. Each request to resume a file transaction shall contain exactly one instruction to resume a file transaction.

c. Each instruction to resume a file transaction shall contain:

1. the identifier of the CFDP entity that resumes the file transaction;
2. the identifier of the transaction to resume, made of:
   (a) the identifier of the source CFDP entity;
   (b) the transaction sequence number.

d. The file transaction subservice shall reject any request to resume a file transaction if any of the following conditions occur:

1. the instruction contained in the request refers to an unknown CFDP entity;
2. the instruction contained in the request refers to a transaction which is not in ACTIVE status.

NOTE the validity of the transaction identifier is under CFDP entity responsibility

e. For each request to resume a file transaction that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to resume a file transaction, the file transaction subservice shall deliver the corresponding resume request to the CFDP entity

6.24.4.11 Cancel a file transaction

a. The file transaction subservice shall provide the capability to cancel a file transaction.

NOTE The corresponding requests are of message type "TC[24,7] cancel a file transaction".

b. Each request to cancel a file transaction shall contain exactly one instruction to cancel a file transaction.

c. Each instruction to cancel a file transaction shall contain:

1. the identifier of the CFDP entity that cancels the file transaction;
2. the identifier of the transaction to cancel, made of:
   (a) the identifier of the source CFDP entity;
   (b) the transaction sequence number.

d. The file transaction subservice shall reject any request to cancel a file transaction if any of the following conditions occur:

1. the instruction contained in the request refers to an unknown CFDP entity;
2. the instruction contained in the request refers to a transaction which is not in ACTIVE status.

NOTE the validity of the transaction identifier is under CFDP entity responsibility

e. For each request to cancel a file transaction that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to cancel a file transaction, the file transaction subservice shall deliver the corresponding cancel request to the CFDP entity

6.24.4.12 Report file transactions status

a. The file transaction subservice shall provide the capability to report the status of file transactions.

NOTE The corresponding requests are of message type "TC[24,8] report status of file transactions". The responses are data reports of message type "TM[24,9] file transactions status report".

b. Each request to report the status of file transactions shall contain exactly one instruction to report the status file transactions.

c. Each instruction to report the status of file transactions shall contain:

1. the identifier of the CFDP entity that reports the status of its file transactions
2. the direction of the reported transactions

NOTE direction can be transmission (from the CFDP entity that reports the status), reception (to the CFDP entity that reports the status) or both

d. The file transaction subservice shall reject any request to report the status of file transactions if the contained instruction:
   1. refers to an unknown CFDP entity
   2. refers to an unknown direction

e. For each request to report the status of active file transactions that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each valid instruction to report the status of active file transactions, the file transaction subservice shall generate a notification for each active file transaction that includes:
   1. source CFDP entity ID
   2. destination CFDP entity ID
   3. identifier of the file transaction, that consist of:
      (a) transaction's source CFDP entity ID
      (b) transaction sequence number
   4. transaction status
   5. on-board file path
   6. size of the file on-board
   7. transaction progress
   8. NAK counters:
      (a) number of negative acknowledgement directives issued if the CFDP entity is receiving the file (acting as destination)
      (b) number of negative acknowledgement directives received if the CFDP entity is transmitting the file (acting as source)

g. For each request that contains a valid instruction to report the status of active file transactions, the file transaction subservice shall generate a single file transaction status report that includes the number of active file transactions and all active file transaction notifications

NOTE the transaction progress is expressed as the byte offset from the beginning of the file that the CFDP entity reporting the status has transmitted or received at the time of the report request processing

6.24.4.13 Modify a remote CFDP entity configuration

a. The file transaction subservice shall provide the capability to modify a remote CFDP entity configuration.

NOTE The corresponding requests are of message type "TC[24,10] modify a remote CFDP entity configuration".

b. Each request to modify a remote CFDP entity configuration shall contain exactly one instruction to modify a remote CFDP entity configuration.

c. Each instruction to modify a remote CFDP entity configuration shall contain:

1. the identifier of the targeted CFDP entity
2. the CFDP entity configuration to be applied

d. The file transaction subservice shall reject any request to modify a remote CFDP entity configuration if the contained instruction:

1. refers to an invalid CFDP entity
2. refers to an invalid default transmission mode

e. For each request to modify a remote CFDP entity configuration that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to modify a remote CFDP entity configuration, the file transaction subservice shall modify the configuration of the CFDP entity referred by the instruction

6.24.4.14 Report a remote CFDP entity configuration

a. The file transaction subservice shall provide the capability to report a remote CFDP entity configuration.

NOTE The corresponding requests are of message type "TC[24,11] report remote CFDP entity configuration". The responses are data reports of message type "TM[24,12] remote CFDP entity configuration report".

b. Each request to report a remote CFDP entity configuration shall contain exactly one instruction to report a remote CFDP entity configuration.

c. Each instruction to report a remote CFDP entity configuration shall contain:

1. the identifier of the targeted CFDP entity

d. The file transaction subservice shall reject any request to report a remote CFDP entity configuration if the contained instruction:

1. refers to an invalid CFDP entity

e. For each request to report a remote CFDP entity configuration that is rejected, the file transaction subservice shall generate a failed start of execution notification

f. For each valid instruction to report a remote CFDP entity configuration, the file transaction subservice shall generate a remote CFDP entity configuration notification that includes:

1. the identifier of the CFDP entity
2. the CFDP entity configuration

g. For each request that contains a valid instruction to report a remote CFDP entity configuration, the file transaction subservice shall generate a single remote CFDP entity configuration report that includes the related remote CFDP entity configuration notification

6.24.4.15 Subservice observables

a. The following observables shall be defined for the file transaction subservice:

1. number of active transactions
2. ID of the last transaction for which a file data PDU has been processed
3. progress (current offset in bytes) of the last transaction for which a file data PDU has been processed
4. ID of the last terminated transaction
5. for each destination CFDP entity, the accummulated data volume retransmitted within the current transmission/reception opportunity

6.24.5 File downlink manager subservice

6.24.5.1 Capability

a. The file downlink manager subservice shall maintain at least one downlink manager, acting as user of an on-board file transaction entity

NOTE if CFDP is the file transaction protocol in use, the on-board file transaction entity is the local CFDP entity, the owner of the file before the transmission starts (source CFDP entity)

b. A downlink manager shall be user of a unique file transaction entity

NOTE 1 if CFDP is the file transaction protocol in use, the on-board file transaction entity is the local CFDP entity

NOTE 2 a local CFDP entity can be associated to more than one user, in particular to more than one downlink manager

c. A downlink manager shall request the initiation of file transmissions to its file transaction entity

NOTE if CFDP is the file transaction protocol in use, a file transmission is initiated through a Put.request primitive, see 4.3 in CCSDS 727.0-B-5

6.24.5.2 Downlink manager processing logic

a. For each file downlink manager managed by the file downlink management subservice, that subservice shall maintain a status indicating whether that downlink manager is STOPPED, STARTED or SUSPENDED

NOTE 1 For valid transitions between statuses, refer to 6.24.5.7, 6.24.5.8 and 6.24.5.9

NOTE 2 No effect on downlink manager behaviour is expected for self-state transitions

b. Once started, the downlink manager shall periodically assess if a new file transaction can be initiated according to the directory priority scheme

c. Once started, if a new file transaction can be initiated, the downllink manager shall deliver to the associated local file transaction entity a put request for the new file

NOTE the CFDP standard flow label can be used to propagate file priority to the local CFDP entity

d. Once started, for each previous file not found the downlink manager shall deliver to the associated local file transaction entity a cancel request for the associated downlink transaction and generate an event report of medium severity

NOTE A file will not be found by the downlink manager in case of file move or deletion out of downlink manager subservice (e.g. file handling subservice)

6.24.5.3 Downlink manager configuration

a. Each file downlink manager shall have the following configurable attributes:

1. identifier of the file transaction entity used by the downlink manager

NOTE if CFDP is the file transaction protocol in use, the file transaction entity is the local CFDP entity

2. the file transmission QoS, if different QoS are supported by the file transaction protocol in use

NOTE if CFDP is the file transaction protocol in use, QoS refers to unreliable transfer (class 1) or reliable transfer (class 2)

3. the maximum number of concurrent file transactions supported by the downlink manager

4. a set of directories containing the files to be transmitted, for each directory:

- path in the file system
- a downlink priority
- destination file transaction entity ID

NOTE 1 if CFDP is the file transaction protocol in use, the destination file transaction entity is the destination

CFDP entity, the receiver of the file to be transmitted

NOTE 2 this standard does not prevent a directory to be in the directory configuration of more than one downlink manager, however it is not recommended as it can lead to unexpected behaviour

5. whether oldest or newest files are prioritised with respect to other files of same priority for downlink
6. whether files are deleted or moved to a back-up directory on-board after successful transaction completion
7. the path of the back-up directory, if this option has been selected

b. Whether if unbounded files are available for downlink shall be declared when specifying this subservice

NOTE 1 unbounded files refers to created files upon closure

NOTE 2 if not declared, only closed files are available for downlink

c. The configuration policy to apply when starting and restarting the file downlink manager shall be declared when specifying the subservice.

NOTE the configuration policy includes as a minimum unbounded files policy and for each downlink manager the default state and configuration

6.24.5.4 Modifying downlink manager configuration

a. The file downlink manager subservice shall provide the capability to modify the downlink manager configuration.

NOTE The corresponding requests are of message type "TC[24,13] modify the downlink manager configuration".

b. Each request to modify the downlink manager configuration shall contain exactly one instruction to modify the downlink manager configuration.

c. Each instruction to modify the downlink manager configuration shall contain:

1. the ID of the downlink manager which configuration is to be modified
2. the file transaction entity ID
3. the file transmission QoS
4. maximum number of concurrent file transactions
5. file age ordering criteria
6. action after file succesful downlink
7. back-up directory path

NOTE 1 if CFDP is the file transaction protocol in use, the file transaction entity is the local CFDP entity

NOTE 2 file age ordering criteria can be oldest first or newest first

NOTE 3 the action after file succesful downlink can be: “delete the file” or “move it to backup directory”, the back-up directory path is useful only for the later

NOTE 4 when a file is moved to the back-up directory its age should remain unchanged

d. The file downlink manager subservice shall reject any request to modify the downlink manager configuration if any of the following conditions occurs:

1. the instruction contained refers to an invalid downlink manager ID
2. the downlink manager state is STARTED
3. the instruction contained refers to an invalid local file transaction entity ID
4. the instruction contained refers to an invalid QoS
5. the instruction contained refers to an invalid file age ordering criteria
6. the instruction contained refers to an invalid action after file succesful downlink
7. the instruction contained refers to an unknown backup directory path

e. For each request to modify the downlink manager configuration that is rejected, the file downlink manager subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to modify the downlink manager configuration, the file transaction subservice shall modify the configuration of the downlink manager referred in the instruction

6.24.5.5 Adding directories to the downlink manager directory configuration

a. The file downlink manager subservice shall provide the capability to add directories to the downlink manager directory configuration.

NOTE The corresponding requests are of message type "TC[24,14] add directories to the downlink manager directory configuration".

b. Each request to add directories to the downlink manager directory configuration shall contain:

1. the ID of the downlink manager which directory configuration is to be updated
2. an ordered list of one or more instructions to add a directory to the downlink manager directory configuration

c. Each instruction to add a directory to the downlink manager directory configuration shall contain:

1. directory path
2. directory priority

3. destination file transaction entity ID

NOTE 1 this standard does not prevent asigning the same priority to more than one directory, but in this case the expected behaviour is mission specific

NOTE 2 if CFDP is the file transaction protocol in use, the destination file transaction entity is the remote CFDP entity

d. The file downlink manager subservice shall reject any request to add directories to the downlink manager directory configuration if any of the following conditions occurs:

1. that request refers to an invalid downlink manager ID
2. the referred downlink manager is not in STOPPED state
3. that request contains an instruction where any of the following conditions occurs:
   (a) refers to an unknown directory path
   (b) refers to an invalid priority
   (c) refers to an invalid destination file transaction entity
   (d) refers to a directory which is already in the set of directories of the downlink manager

e. For each request to add directories to the downlink manager directory configuration that is rejected, the file downlink manager subservice shall generate a failed start of execution notification

f. For each request to add directories to the downlink manager directory configuration that contains only valid instructions, the file downlink manager subservice shall execute those instructions in the order of their appearance in that request

g. For each valid instruction to add a directory to the downlink manager directory configuration, the file downlink manager subservice shall add the directory to the downlink manager set of directories

6.24.5.6 Removing directories from the downlink manager configuration

a. The file downlink manager subservice shall provide the capability to remove directories from the downlink manager directory configuration.

NOTE The corresponding requests are of message type "TC[24,15] remove directories from the downlink manager directory configuration".

b. Each request to remove directories from the downlink manager directory configuration shall contain:

1. the ID of the downlink manager which directory configuration is to be updated
2. an ordered list of one or more instructions to remove a directory from the downlink manager directory configuration

c. Each instruction to remove a directory from the downlink manager directory configuration shall contain:

1. the path of the directory to be removed

d. The file downlink manager subservice shall reject any request to remove directories to the downlink manager directory configuration if any of the following conditions occurs:

1. that request refers to an invalid downlink manager ID
2. the referred downlink manager is not in STOPPED state
3. that request contains an instruction where any of the following conditions occurs:

(e) refers to a directory not found in the set of directories allocated to the downlink manager

e. For each request to remove directories from the downlink manager directory configuration that is rejected, the file downlink manager subservice shall generate a failed start of execution notification

f. For each request to remove directories from the downlink manager directory configuration that contains only valid instructions, the file downlink manager subservice shall execute those instructions in the order of their appearance in that request

g. For each valid instruction to remove a directory from the downlink manager directory configuration, the file downlink manager subservice shall remove the directory from the downlink manager set of directories

NOTE if the request to remove directories from the downlink manager directory configuration leads to an empty set of directories, the file downlink manager subservice can generate an event report of low severity warning of the configuration as it will never perform any data downlink

6.24.5.7 Reporting downlink manager configuration

a. The file downlink manager subservice shall provide the capability to report the configuration of a downlink manager.

NOTE The corresponding requests are of message type "TC[24,16] report the configuration of a downlink manager". The responses are data reports of message type "TM[24,17] downlink manager configuration report".

b. Each request to report the configuration of a downlink manager shall contain exactly one instruction to report the configuration of a downlink manager

c. Each instruction to report the configuration of a downlink manager shall contain:

1. the ID of the downlink manager which report is requested

d. The file downlink manager subservice shall reject any request to report the configuration of a downlink manager if the instruction contained refers to an invalid downlink manager ID

e. For each request to report the configuration of a downlink manager that is rejected, the file downlink manager subservice shall generate a failed start of execution notification

f. For each valid instruction to report the configuration of a downlink manager, the file downlink manager subservice shall generate a notification that includes:

1. the ID of the downlink manager which directory report is requested;
2. the unbounded files permission flag;
3. the file transaction entity ID;
4. the file transaction QoS;
5. the downlink manager status;
6. the maximum number of concurrent file transactions;
7. the current number of initiated file transactions;
8. the file age ordering criteria;
9. the action on succesful file downlinked;
10. the back up directory path;
11. for each directory in the set of directories:
    (a) the directory path;
    (b) the priority of the directory assigned within the downlink manager;
    (c) the destination file transaction entity ID.

g. For each request that contains a valid instruction to report the configuration of a downlink manager, the file downlink manager subservice shall generate a single downlink manager configuration report that includes the related downlink manager configuration notification

6.24.5.8 Starting the downlink manager

a. The file downlink manager subservice shall provide the capability to start a downlink manager.

NOTE The corresponding requests are of message type "TC[24,18] start downlink manager".

b. Each request to start a downlink manager shall contain exactly one instruction to start a downlink manager

c. Each instruction to start a downlink manager shall contain:

1. the downlink manager ID

d. The file downlink manager subservice shall reject any request to start a downlink manager if any of the following conditions occur:

1. the instruction contained refers to an invalid downlink manager ID
2. the downlink manager status is different from STOPPED or SUSPENDED

e. For each request to start a downlink manager that is rejected, the file downlink manager subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to start a downlink manager the file downlink manager subservice shall:

1. set the downlink manager state to STARTED
2. deliver to the associated local file transaction entity a resume request for each downlink transaction initiated by that downlink manager and currently being serviced

NOTE 1 this applies to any file in any of the directories from the downlink manager configuration

NOTE 2 for the processing of new transactions, refer to 6.24.5.2

6.24.5.9 Suspend downlink manager

a. The file downlink manager subservice shall provide the capability to suspend a downlink manager.

NOTE The corresponding requests are of message type "TC[24,19] suspend downlink manager".

b. Each request to suspend a downlink manager shall contain exactly one instruction to suspend a downlink manager.

c. Each instruction to suspend a downlink manager shall contain:

1. the downlink manager ID

d. The file downlink manager shall reject any request to suspend a downlink manager if any of the following conditions occurs:

1. the instruction contained refers to an invalid downlink manager ID
2. the downlink manager status is different from STARTED

e. For each request to suspend a downlink manager that is rejected, the downlink manager subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to suspend a downlink manager the file downlink manager subservice shall:

1. set the state of the refered downlink manager to SUSPENDED
2. deliver to the associated local file transaction entity a suspend request for each downlink transaction initiated by that downlink manager and currently in ACTIVE-RUNNING status

6.24.5.10 Stop downlink manager

a. The file downlink manager subservice shall provide the capability to stop a downlink manager.

NOTE The corresponding requests are of message type "TC[24,20] stop downlink manager".

b. Each request to stop a downlink manager shall contain exactly one instruction to stop a downlink manager.

c. Each instruction to stop a downlink manager shall contain:

1. the downlink manager ID

d. The file downlink manager subservice shall reject any request to stop a downlink manager if any of the following conditions occurs:

1. the instruction contained refers to an invalid downlink manager ID
2. the downlink manager status is different from STARTED or SUSPENDED

e. For each request to stop a downlink manager that is rejected, the file downlink manager subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to stop a downlink manager the file downlink manager subservice shall:

1. set the downlink manager state to STOPPED
2. for each of the file transactions initiated by that downlink manager and currently in ACTIVE-RUNNING status, deliver a cancel request to the associated file transaction entity

NOTE if CFDP is the file transaction protocol in use, the file transaction entity if the local CFDP entity

6.24.5.11 Subservice observables

a. The following observables shall be defined for the downlink manager subservice:

1. for each downlink manager:

(a) downlink manager ID

(b) current status of the downlink manager

(c) current number of file transactions initiated

NOTE for downlink manager configuration refer to 6.24.5.7

6.25 ST[25] file data storage

6.25.1 Scope

6.25.1.1 General

The file data storage service type provides the capability to manage the storing process of on-board data generated from different sources into files for later downlink using the file transfer service.

Files are organized in directories, where all the files generated from data sources are maintained on-board.

The file data storage service type defines two standardized subservice types, i.e.:

- the data storage control subservice type;
- the PUS report-type data source control subservice type.

6.25.1.2 Data storage control subservice

The concept of data source was introduced in clause 6.24.1.1. Data sources can be associated to either a physical data link interface (for instance the spacewire link of an intelligent payload unit) or a set of PUS reports. The file data storage service only standardizes the later ones, referred to as PUS report-type data sources. The definition of data sources based on physical data link interfaces is mission-specific.

A data source can be mapped to one and only one directory, where all the files generated from that data source will be maintained.

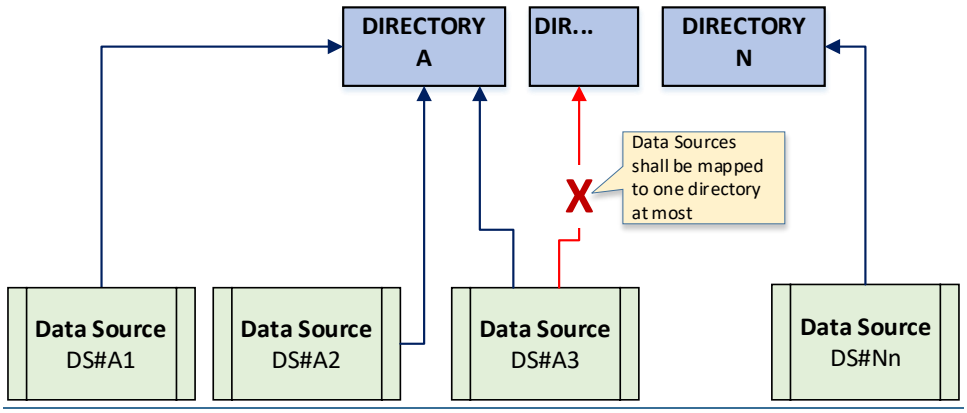

Figure 6.25-1 Data Source concept

Each data source has a recording status, which can be enabled or disabled. When the recording status of a data source is enabled its data stream is stored in files located in the directory where is mapped.

The subservice provides the capability to split autonomously the recorded data into several files if some conditions occur. For a given data source only one file is being written at a time. Writing into the only open file stops and a new one is autonomously created to continue storage ensuring no data loss under the following conditions:

- after writing a given amount of data into the current file
- after writing into the current file for a given duration
- upon manual request

When a file has been downlinked it can be autonomously deleted or moved to a back-up directory as stated in ST[24]. The downlinked file will stay in the back-up directory for a configurable maximum time (the maximum file age) after which it will be permanently deleted from the on-board file management system.

The data storage control subservice type provides capabilities for:

- configuring and reporting the directory attributes managed by this subservice
- configuring and reporting the mapping of data sources to directories including the data storage attributes associated to such mapping
- enabling and disabling the data source recording status
- requesting closure of the files currently in use for storing on-board data from data sources

When data is received from a PUS report-type data source, the data storage control subservice also ensures that files are always closed at PUS report boundaries and thus also that PUS reports are not split between two consecutive files upon autonomous closure/opening of corresponding recording file.

6.25.1.3 PUS report-type data source control subservice

The PUS report-type data source control subservice allows to configure combinations of PUS reports identified by APID, type and subtype as data sources.

As any other on-board data source, a PUS report-type data source can be assigned to a storage directory using the data storage control subservice.

The PUS report-type data source control subservice type provides capabilities for:

- adding and deleting PUS report-type data source control configurations
- report the content of the PUS report-type data source control configurations

6.25.2 Service layout

6.25.2.1 Subservice

6.25.2.1.1 Data storage control subservice

a. Each file data storage service shall contain exactly one data storage control subservice.

6.25.2.1.2 PUS report-type data source control subservice

a. Each file data storage service shall contain at most one PUS report-type data source control subservice.

6.25.2.2 Application process

a. For each file data storage service that contains both, a data storage control subservice and a PUS report-type data source control subservice, the two subservice providers of that service shall be hosted by the same application process.

6.25.2.3 Accessibility

6.25.2.3.1 Service

a. Each file data storage service shall be associated to exactly one event reporting subservice.

NOTE 1 This event reporting subservice (refer to clause 6.5) is responsible for catching the events raised by the file data storage service and issuing the corresponding event notifications.

NOTE 2 The events that can be raised by the file data storage service are identified by the combination of the identifier of the application process that hosts the event reporting subservice and an event definition identifier.

b. The event reporting subservice that is associated to the file data storage service shall be declared when specifying that file data storage service.

6.25.3 Data storage control subservice

6.25.3.1 Data sources

a. The list of data source types and allowed range for their identifiers that the data storage control subservice can support shall be declared when specifying that subservice.

b. The maximum number of data sources that the data storage control subservice can maintain at any time shall be declared when specifying that subservice.

NOTE it covers all data source definitions: PUS report-type as well as any other type of data source

c. The list of pre-defined data sources maintained by the data storage control subservice shall be declared when specifying that subservice.

d. The data storage control subservice shall maintain a status for each data source indicating whether its recording status is enabled or disabled.

NOTE This status is named “data source recording status”.

e. The default data source recording status for each data source shall be declared when specifying the data storage control subservice

f. Each data source shall be managed by exactly one data storage control subservice.

NOTE Within a subservice, each data source is uniquely identified by a data source identifier.

6.25.3.2 Usage of directories

a. The maximum number of directories that the data storage control subservice can maintain at any time shall be declared when specifying that subservice.

b. Each directory used for file data storage shall support the following attributes:

1. a maximum file age, beyond which files within this directory shall be deleted automatically.
2. a maximum storage size usable by this directory.

NOTE 1 file age is defined as time elapsed since file creation time

NOTE 2 automatic deletion above maximum file age is disabled if the file age is set to 0

NOTE 3 if a maximum file age greater than 0 is set, the value is greater than the maximum duration of data write operation declared when creating the mapping between a specific data source and the directory. This needs to be true considering all data sources mapped to the directory

6.25.3.3 Data storage control subservice configuration policy

a. The data storage control subservice configuration policy to apply when starting and restarting the data storage control subservice shall be declared when specifying the subservice.

NOTE The configuration includes as a minimum:

- the data sources definitions
- the configuration of each data source (this includes the recording status)
- the value of the attributes managed by this subservice for each directory
- the directory mapping to data sources and the value of all corresponding attributes

6.25.3.4 Autonomous file control

a. The data storage control subservice shall autonomously close a file for storage of incoming data from a data source if any of the following conditions occurs:

1. the data source recording status changes from enabled to disabled
2. the file reaches the maximum file size allowed for this data source by the corresponding directory mapping to data source definition;
3. the file write duration reaches the maximum duration of data write operation allowed for this data source by the corresponding directory mapping to data source definition;
4. the maximum size usable in this directory by this data source as allowed by the corresponding directory mapping to data source definition is reached;

b. The data storage control subservice shall generate an informative event report to notify the autonomous closure of a file for data storage

c. The data storage control subservice shall autonomously create and open a new file for storage of incoming data from a data source if all the following conditions occur:

1. the data source recording status is enabled;
2. a directory is mapped to the data source and the maximum size usable in this directory by this data source as allowed by the corresponding directory mapping to data source definition is not reached ;
3. there is an incoming data stream from the data source;
4. no other file is open (i.e. write active) for the data source.

d. The data storage control subservice shall ensure that no incoming data is lost if all the conditions to automatically create and open a new file for data storage are met

e. The data storage control subservice shall assign a unique file name upon autonomous file creation, which shall be a variable character string composed of a prefix followed by a suffix

NOTE 1 no naming or length rules are imposed in this standard

NOTE 2 the inclusion of an extension in the suffix is a specific mission choice

NOTE 3 the inclusion of the file creation date/time in the suffix is a specific mission choice but recommended to ensure file name uniqueness

f. When receiving data coming from PUS report-type data sources, the data storage control subservice shall ensure that files are always closed at PUS report boundaries and thus also that PUS reports are not split between two consecutive files upon autonomous closure/opening of files.

g. The data storage control subservice shall ensure the protection of open files against move or deletion

NOTE this can be performed through the file locking mechanism already described in ST[23]

6.25.3.5 Set directory data storage attributes

a. The data storage control subservice shall provide the capability to set the data storage attributes of a directory.

NOTE The corresponding requests are of message type "TC[25,1] set directory data storage attributes".

b. Each request to set the data storage attributes of a directory shall contain exactly one instruction to set the data storage attributes of a directory.

c. Each instruction to set the data storage attributes of a directory shall contain:

1. the directory path;
2. maximum storage size usable by this directory;
3. the maximum file age.

d. The data storage control subservice shall reject any request to set the data storage attributes of a directory if any of the following conditions occurs:

1. that request contains an instruction that refers to an unknown directory path
2. any of the data sources mapped to the directory has its recording status set to “enabled”

e. For each request to set the data storage attributes of a directory that is rejected, the data storage control subservice shall generate a failed start of execution notification

f. For each request that contains a valid instruction to set the data storage attributes of a directory, the data storage control subservice shall set the data storage attributes of the referred directory with the values provided

NOTE Other directory attributes non data storage specific are covered at file management system level and ST[23]

6.25.3.6 Reporting directory data storage attributes

a. The data storage control subservice shall provide the capability to report the data storage attributes of directories.

NOTE The corresponding requests are of message type "TC[25,2] report directory data storage attributes". The responses are data reports of message type "TM[25,3] directory data storage attributes report".

b. Each request to report directory data storage attributes shall contain exactly one of:

1. an ordered list of one or more instructions to report directory data storage attributes;
2. a single instruction to report data storage attributes of all directories

NOTE The instructions to report data storage attributes of all directories contain no argument.

c. Each instruction to report directory data storage attributes shall contain:

1. the directory path.

d. The data storage control subservice shall reject any request to report directory data storage attributes if any of the following conditions occurs:

1. that request contains an instruction that refers to an unknown directory path

e. For each request to report directory data storage attributes that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to report directory data storage attributes that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report directory data storage attributes, the data storage control subservice shall generate a single directory data storage attributes notification that includes:

1. the directory path;
2. the maximum storage size usable by this directory;
3. the free storage size in this directory;
4. the maximum file age;
5. the number of files in this directory;
6. for each file in this directory:

(a) file name

h. For each valid instruction to report the data storage attributes of all directories, the data storage control subservice shall generate a single directory data storage attributes notification for each directory

NOTE For the content of the data storage attributes notification, see 6.25.3.6.g above

i. For each valid request to report directory data storage attributes, the data storage control subservice shall generate a single directory data storage

attributes report that contains all related directory data storage attributes notifications.

6.25.3.7 Adding data sources to directory mapping

a. The data storage control subservice shall provide the capability to add data sources to directory mapping.

NOTE The corresponding requests are of message type "TC[25,4] add data sources to directory mapping".

b. Each request to add data sources to directory mapping shall contain an ordered list of one or more instructions to add data sources to the directory mapping.

c. Each instruction to add data sources to the directory mapping shall contain:

1. the data source identifier;
2. the directory path;
3. the prefix in the name of any new file associated to the data source;

NOTE suffix will be automatically generated according to mission specific rules

4. the maximum file size allowed for this data source in this directory;
5. the maximum duration of data write operation allowed for this data source in this directory;
6. the maximum storage size usable by this data source in this directory;

d. The data storage control subservice shall reject any request to add data sources to the directory mapping if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid data source identifier;
2. that request contains an instruction that refers to an unknown directory path;
3. that request contains an instruction that refers to a data source already mapped to a different directory;

e. For each request to add data sources to the directory mapping that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to add data sources to the directory mapping that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to add data sources to the directory mapping, the data storage control subservice shall:

1. add the data source to the directory set of data sources if not already existing;
2. protect the directory.

NOTE for directory protect/unprotect refer to 6.23.4.6.4/6.23.4.6.5

6.25.3.8 Removing data sources from directory mapping

a. The data storage control subservice shall provide the capability to remove data sources from directory mapping.

NOTE The corresponding requests are of message type "TC[25,5] remove data sources from directory mapping".

b. Each request to remove data sources from directory mapping shall contain an ordered list of one or more instructions to remove data sources from directory mapping.

c. Each instruction to remove data sources from directory mapping shall contain:

1. the data source identifier.

d. The data storage control subservice shall reject any request to remove data sources from directory mapping if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid data source identifier;
2. the data source has its recording status set to enabled.

e. For each request to remove data sources from directory mapping that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to remove data sources from directory mapping that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to remove data sources from directory mapping, the data storage control subservice shall:

1. remove the data source from the directory set of data sources
2. if the directory set of data sources becomes empty, unprotect the directory.

NOTE for directory protect/unprotect refer to 6.23.4.6.4/6.23.4.6.5

6.25.3.9 Reporting data sources mapped to directory

a. The data storage control subservice shall provide the capability to report data sources mapped to directory.

NOTE The corresponding requests are of message type "TC[25,6] report data sources mapped to directory ". The responses are data reports of message type "TM[25,7] data sources mapped to directory report".

b. Each request to report data sources mapped to directory shall contain exactly one of:

1. an ordered list of one or more instructions to report data sources mapped to directory;
2. a single instruction to report all the data sources mapping to directories.

NOTE The instructions to report all the data sources mapping to directories contain no argument.

c. Each instruction to report data sources mapped to directory shall contain:

1. the data source identifier.

d. The data storage control subservice shall reject any request to report data sources mapped to directories if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid data source identifier

e. For each request to report data sources mapped to directory that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to report data sources mapped to directory that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report data sources mapped to directory, the data storage control subservice shall generate, for each data source, a single data sources mapped to directory notification that includes:

1. the data source identifier;
2. the directory path;
3. the prefix in the name of the files associated to the data source;
4. the maximum file size allowed for this data source in this directory;
5. the maximum duration of data write operation allowed for this data source in this directory;
6. the maximum storage size usable by this data source in this directory;

h. For each valid instruction to report all data sources mapping to directories, the data storage control subservice shall generate a single data sources mapped to directory notification for each data source

NOTE For the content of the data sources mapped to directory notification, see 6.25.3.9.g above

i. For each valid request to report data sources mapped to directory, the data storage control subservice shall generate a single data sources mapped to directory report that contains all related data sources mapped to directory notifications.

6.25.3.10 Reporting data sources recording status

a. The data storage control subservice shall provide the capability to report data sources recording status.

NOTE The corresponding requests are of message type "TC[25,8] report data sources recording status". The responses are data reports of message type "TM[25,9] data sources recording status report".

b. Each request to report data sources recording status shall contain exactly one of:

1. an ordered list of one or more instructions to report data sources recording status;
2. a single instruction to report the recording status of all data sources

NOTE The instructions to report the recording status of all data sources contain no argument.

c. Each instruction to report data sources recording status shall contain:

1. the data source identifier.

d. The data storage control subservice shall reject any request to report data sources recording status if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid data source identifier

e. For each request to report data sources recording status that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to report data sources recording status that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to report data sources recording status, the data storage control subservice shall generate a single data sources recording status notification that includes:

1. the data source identifier;
2. the recording status.

h. For each valid instruction to report all data sources recording status, the data storage control subservice shall generate a single data sources recording status notification for each data source

NOTE For the content of the data sources recording status notification, see 6.25.3.10.g above

i. For each valid request to report data sources recording status, the data storage control subservice shall generate a single data sources recording status report that contains all related data sources recording status notifications.

1.

6.25.3.11 Enabling data source recording status

a. The data storage control subservice shall provide the capability to enable data sources recording status.

NOTE The corresponding requests are of message type "TC[25,10] enable data sources recording status".

b. Each request to enable data sources recording status shall contain exactly one of:

1. an ordered list of one or more instructions to enable data sources recording status;
2. a single instruction to enable the recording status of all data sources.

NOTE 1 The subservice user is responsible for ensuring the correct processing of the data stream generated after enabling all data sources

NOTE 2 The instructions to enable the recording status of all data sources contain no argument

c. Each instruction to enable data sources recording status shall contain:

1. the data source identifier.

d. The data storage control subservice shall reject any request to enable data sources recording status if any of the following conditions occurs:

1. that request contains an instruction that refers to an invalid data source
2. that request contains an instruction that refers to a data source identifier which is not mapped to a directory

e. For each request to enable data sources recording status that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to enable data sources recording status that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to enable data sources recording status, the data storage control subservice shall:

1. set the recording status to “enabled”

NOTE refer to 6.25.3.4.c autonomous file control for the conditions to be fulfilled for opening a file apart from the data source recording status

h. For each valid instruction to enable all data sources recording status, the data storage control subservice shall:

1. for each data sources, set the recording status to “enabled”

6.25.3.12 Disabling data source recording status

a. The data storage control subservice shall provide the capability to disable data sources recording status.

NOTE The corresponding requests are of message type "TC[25,11] disable data sources recording status".

b. Each request to disable data sources recording status shall contain exactly one of:
   1. an ordered list of one or more instructions to disable data sources recording status,
   2. a single instruction to disable the recording status of all data sources.

      NOTE The instructions to disable the recording status of all data sources contain no argument

c. Each instruction to disable data sources recording status shall contain:
   1. the data source identifier.

d. The data storage control subservice shall reject any request to disable data sources recording status if any of the following conditions occurs:
   1. that request contains an instruction that refers to an invalid data source

e. For each request to disable data sources recording status that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to disable data sources recording status that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to disable data sources recording status, the data storage control subservice shall:
   1. set the recording status to “disabled”
   2. stop routing the data from the data source stream to the file currently used for recording
   3. close the relevant file associated to this data source, i.e. the one currently in use

h. For each valid instruction to disable all data sources recording status, the data storage control subservice shall:
   1. for each data source,
      (a) set the recording status to “disabled”;
      (b) stop routing the data from the data source stream to the file currently used for recording;
      (c) close the relevant file associated to this data source, i.e. the one currently in use

6.25.3.13 Closure of the file currently used for data storage

a. The data storage control subservice shall provide the capability to close the file currently used for data storage for one or more data sources.

NOTE The corresponding requests are of message type "TC[25,12] close file currently used for data storage".

b. Each request to close the file currently used for data storage shall contain exactly one of:

1. an ordered list of one or more instructions to close the file currently used for data storage;
2. a single instruction to close all files currently used for data storage.

NOTE The instructions to close all files currently used for data storage contain no argument

c. Each instruction to close the file currently used for data storage shall contain:

1. the data source identifier.

d. The data storage control subservice shall reject any request to close the file currently used for data storage if any of the following conditions occur:

1. that request contains an instruction that refers to an invalid data source.

e. For each request to close the file currently used for data storage that is rejected, the data storage control subservice shall generate a failed start of execution notification.

f. For each request to close the file currently used for data storage that contains only valid instructions, the data storage control subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to close the file currently used for data storage, the data storage control subservice shall:

1. close the file currently used for recording by the data source referred to by that instruction.

h. For each valid instruction to close all files currently used for data storage, the data storage control subservice shall:

1. For each data source,

(a) close the file currently used for recording by the data source referred to by that instruction.

6.25.3.14 Subservice observables

This Standard does not define any observables for the data storage control subservice.

NOTE all related information can be retrieved through specific report packets defined within the subservice

6.25.4 PUS report-type data source control subservice

6.25.4.1 Accessibility

6.25.4.1.1 Application process

a. The list of application processes that are controlled by the PUS report-type data source control subservice shall be declared when specifying that subservice.

NOTE The PUS report-type data source control subservice always controls the storage of reports generated by the application process that hosts that subservice.

b. The PUS report-type data source control subservice shall be able to handle, at any time, all reports that are generated by each application process that is controlled by that subservice.

6.25.4.1.2 Access to storage directories

a. The PUS report-type data source control subservice shall, at any time, have access to the directories maintained by the data storage control subservice of the parent file data storage service.

6.25.4.2 PUS report-type data source control definitions

6.25.4.2.1 PUS report-type data source with application process control configuration

a. The maximum number of PUS report-type data sources that the PUS report-type data source control subservice can maintain at any time shall be declared when specifying that subservice.

b. Each application process PUS report-type data source control definition shall contain:

1. the identifier of the application process to control;
2. a list of zero or more application process related "service type data source control definitions", each one containing:
   (a) the identifier of the service type to control;
   (b) a list of zero or more application process and service type related "report type data source control definitions", each one containing the message subtype identifier of a report type.

NOTE The PUS report-type data source control subservice has knowledge about the application processes that it controls but no knowledge about the service types and report types that they can generate. This lack of knowledge results in the possibility for the subservice to handle on-board, service type data source control definitions or report type data source control definitions that can be meaningless. It is

ground operations responsibility to ensure consistency in this respect.

c. The maximum number of service type data source control definitions that can be contained within an application process data source control definition shall be declared when specifying the PUS report-type data source control subservice.

d. The maximum number of report type data source control definitions that can be contained within a service type data source control definition shall be declared when specifying the PUS report-type data source control subservice.

6.25.4.3 PUS report-type data source control subservice configuration policy

a. The PUS report-type data source control subservice configuration policy to apply when starting and restarting the PUS report-type data source control subservice shall be declared when specifying the subservice

NOTE The configuration includes as a minimum the PUS report-type data source control configurations

6.25.4.4 Managing the application process PUS report-type data source control configuration

6.25.4.4.1 Add report types to the application process PUS report-type data source control configuration

a. The PUS report-type data source control subservice shall provide the capability to add report types to the application process PUS report-type data source control configuration.

NOTE 1 The corresponding requests are of message type "TC[25,13] add report-type to the application process PUS report-type control configuration".

NOTE 2 For the capability to delete report-types from the application process PUS report-type data source control configuration, refer to clause 6.25.4.4.2.

b. Each request to add report-type to the application process PUS report-type data source control configuration shall contain:

1. the data source identifier;
2. at least one of:
   (a) one or more instructions to add a report type to the application process PUS report-type data source control configuration,
   (b) one or more instructions to add all report types of a service type to the application process PUS report-type data source control configuration,
   (c) if the PUS report-type data source control subservice only controls the application process that hosts it, exactly one instruction to add all report types of an application process to

the application process PUS report-type data source control configuration,

(d) if the PUS report-type data source control subservice controls more than one application process, one or more instructions to add all report types of an application process to the application process PUS report-type data source control configuration.

c. The PUS report-type data source control subservice shall reject any request to add report types to the application process PUS report-type data source control configuration if:

1. that request refers to an invalid data source identifier
2. the creation would exceed the maximum number of PUS report-type data sources

d. Each instruction to add a report type to the application process PUS report-type data source control configuration shall contain:

1. if the PUS report-type data source control subservice controls more than one application process, the application process identifier addressed by that instruction,
2. the report type identifier consisting of:
   (a) the service type identifier;
   (b) the message subtype identifier.

   NOTE For item 1, refer to requirement 6.15.4.1.1a.

e. Each instruction to add all report types of a service type to the application process PUS report-type data source control configuration shall contain:

1. if the PUS report-type data source control subservice controls more than one application process, the application process identifier addressed by that instruction,
2. the service type identifier.

f. Each instruction to add all report types of an application process to the application process PUS report-type data source control configuration shall contain:

1. if the PUS report-type data source control subservice controls more than one application process, the application process identifier addressed by that instruction.

g. The PUS report-type data source control subservice shall reject any request containing instructions to add a report type to the application process PUS report-type data source control configuration if any of the following conditions occur:

1. that request contains an instruction that refers to an application process that is not controlled by that subservice;
2. that request contains an instruction that implies the addition of a service type data source control definition and the maximum number of service type data source control definitions for the corresponding application process data source control definition is already reached;

3. that request contains an instruction for which the maximum number of report type data source control definitions that can be contained within the corresponding service type data source control definition is already reached;
4. that request contains an instruction for which the corresponding service type data source control definition has no report type data source control definition already defined;
5. that request contains an instruction for which the corresponding application process data source control definition has no service type data source control definition already defined.

NOTE 1 For item 4, if all report types of a service type are already in the application process PUS report-type data source control configuration, it is meaningless to ask for the addition of a report type for that service type.

NOTE 2 For item 5, if all report types of an application process are already in the application process PUS report-type data source control configuration, it is meaningless to ask for the addition of a report type for that application process.

h. The PUS report-type data source control subservice shall reject any request continaing instructions to add all report types of a service type to the application process PUS report-type data source control configuration if any of the following conditions occur:
   1. that request contains an instruction that refers to an application process that is not controlled by that subservice;
   2. that request contains an instruction that implies the addition of a service type data source definition and the maximum number of service type data source definitions for the corresponding application process data source definition is already reached;
   3. that request contains an instruction for which the corresponding application process data source control definition has no service type data source control definition already defined.

   NOTE For item 3, if all report types of an application process of an application process are already in the application process PUS report-type data source control configuration, it is meaningless to ask for the addition of a service type for that application process.

i. The PUS report-type data source control subservice shall reject any request containing instructions to add all report types of an application process to the application process PUS report-type data source control configuration if:
   1. that request contains an instruction that refers to an application process that is not controlled by that subservice.

j. For each request to add report types to the application process PUS report-type data source control configuration that is rejected, the PUS report-type

data source control subservice shall generate a failed start of execution notification.

k. For each request to add report types to the application process PUS report-type data source control configuration that contains only valid instructions, the PUS report-type data source control subservice shall execute those instructions in the order of their appearance in that request.

l. For each valid instruction to add a report type to the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall, for the related data source:

1. add, for the specified application process identifier, an application process data source control definition if not already existing;
2. add, for the related application process data source control definition and the specified service type identifier, a service type data source control definition, if not already existing;
3. add, for the related service type data source control definition and the specified message subtype identifier, a report type data source control definition, if not already existing.

m. For each valid instruction to add all report types of a service type to the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall, for the related data source:

1. add, for the specified application process identifier, an application process data source control definition if not already existing;
2. add, for the related application process data source control definition and the specified service type identifier, a service type data source control definition to the related application process data source control definition, if not already existing;
3. delete, if any, all report type data source control definitions of the related service type data source control definition.

n. For each valid instruction to add all report types of an application process to the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall, for the related data source:

1. add, for the specified application process identifier, an application process data source control definition if not already existing;
2. delete, if any, all service type data source control definitions of the related application process data source control definition.

6.25.4.4.2 Delete report types from the application process PUS report-type data source control configuration

a. The PUS report-type data source control subservice shall provide the capability to delete report types from the application process PUS report-type data source control configuration.

NOTE 1 The corresponding requests are of message type "TC[25,14] delete report-type from the application process PUS report-type control configuration".

NOTE 2 For the capability to add report-types to the application process PUS report-type data source control configuration, refer to clause 6.25.4.4.1.

b. Each request to delete report types from the application process PUS report-type data source control configuration shall contain the data source identifier of the PUS report-type data source control configuration to change and exactly one of:

1. at least one of:

(a) one or more instructions to delete a report type from the application process PUS report-type data source control configuration,

(b) one or more instructions to delete a service type from the application process PUS report-type data source control configuration,

(c) if the PUS report-type data source control subservice controls more than one application process, one or more instructions to delete an application process from the application process PUS report-type data source control configuration,

2. a single instruction to empty the entire application process PUS report-type data source control configuration.

c. The PUS report-type data source control subservice shall reject any request to delete report types from the application process PUS report-type data source control configuration if:

1. that request refers to a data source identifier which does not exist

d. Each instruction to delete a report type from the application process PUS report-type data source control configuration shall contain:

1. if the PUS report-type data source control subservice controls more than one application process, the application process identifier addressed by that instruction,

2. the report type identifier consisting of:

(a) the service type identifier;

(b) the message subtype identifier.

NOTE For item 1, refer to requirement 6.15.4.1.1a.

e. The PUS report-type data source control subservice shall reject any request containing instructions to delete a report type from the application process PUS report-type data source control configuration if:

1. that request contains an instruction that refers to a report type identifier that is not in the application process PUS report-type data source control configuration.

f. Each instruction to delete a service type from the application process PUS report-type data source control configuration shall contain:

1. if the PUS report-type data source control subservice controls more than one application process, the application process identifier addressed by that instruction,

2. the service type identifier.

g. The PUS report-type data source control subservice shall reject any request containing instructions to delete a service type from the application process PUS report-type data source control configuration if:

1. that request contains an instruction that refers to a service type identifier that is not in the application process PUS report-type data source control configuration.

h. Each instruction to delete an application process from the application process PUS report-type data source control configuration shall contain:

1. if the PUS report-type data source control subservice controls more than one application process, the application process identifier addressed by that instruction.

i. The PUS report-type data source control subservice shall reject any request containing instructions to delete an application process from the application process PUS report-type data source control configuration if:

1. that request contains an instruction that refers to an application process identifier that is not in the application process PUS report-type data source control configuration.

j. For each request to delete report types from the application process PUS report-type data source control configuration that is rejected, the PUS report-type data source control subservice shall generate a failed start of execution notification.

k. For each request to delete report types from the application process PUS report-type data source control configuration that contains only valid instructions, the PUS report-type data source control subservice shall execute those instructions in the order of their appearance in that request.

l. For each valid instruction to delete a report type from the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall, for the related data source:

1. delete the report type data source control definition related to that specified application process identifier, service type identifier and message subtype identifier;
2. if that report type data source control definition deletion results in an emptied service type data source control definition, delete that service type data source control definition;
3. if that service type data source control definition deletion results in an emptied application process data source control definition, delete that application process data source control definition.

m. For each valid instruction to delete a service type from the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall, for the related data source:

1. delete the service type data source control definitions related to that specified application process identifier and service type identifier;
2. if that service type data source control definition deletion results in an emptied application process data source control definition, delete that application process data source control definition.

n. For each valid instruction to delete an application process from the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall, for the related data source:

1. delete the application process data source control definition related to that specified application process identifier.

o. For each valid instruction to empty the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall, for the related data source:

1. delete, if any, all application process data source control definitions.

6.25.4.4.3 Report the content of the application process PUS report-type data source control configuration

a. The PUS report-type data source control subservice shall provide the capability to report the content of the application process PUS report-type data source control configuration.

NOTE 1 The corresponding requests are of message type "TC[25,15] report the content of the application process PUS report-type data source control configuration". The responses are data reports of message type "TM[25,16] application process PUS report-type data source control configuration content report".

NOTE 2 That capability requires the capability for that subservice to add report types to the application process PUS report-type data source control configuration, refer to clause 6.15.4.4.1.

b. Each request to report the content of the application process PUS report-type data source control configuration shall contain exactly one instruction to report the content of the application process PUS report-type data source control configuration.

c. Each instruction to report the content of the application process PUS report-type data source control configuration shall contain:

1. the data source identifier of the PUS report-type data source.

d. The PUS report-type data source control subservice shall reject any request containing an instruction to report the content of the application process PUS report-type data source control configuration if:

1. that request contains an instruction that refers to an invalid data source.

e. For each request to report the content of the application process PUS report-type data source control configuration that is rejected, the PUS report-type data source subservice shall generate a failed start of execution notification.

f. For each valid instruction to report the content of the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall generate, for each existing application

process data source control definition for that PUS report-type data source, a single application process data source control definition notification that includes:

1. if the PUS report-type data source control subservice controls more than one application process, the related application process identifier;
2. for each related service type data source control definition, if any:
   (a) the related service type identifier;
   (b) for each related report type data source control definition, if any, the related message subtype identifier.

   NOTE For item 1, refer to requirement 6.15.4.1.1a.

g. For each valid request to report the content of the application process PUS report-type data source control configuration, the PUS report-type data source control subservice shall generate a single application process PUS report-type data source control configuration content report that includes:

1. the data source identifier of the related PUS report-type data source;
2. all related application process data source control definition notifications.

6.25.4.5 Subservice observables

This Standard does not define any observables for the PUS report-type data source control subservice.

all related information can be retrieved through report packets.

6.26 ST[26] parameter extraction

6.26.1 Scope

6.26.1.1 General

For missions with different application processes on-board, the parameter extraction service provides a mechanism to make a subset of parameters generated by these application processes accessible to the central autonomy services, for instance: FDIR (as monitored or validity parameter), OBCPs, etc.

The parameter extraction service achieves this goal by extracting parameters from telemetry packets of the corresponding application processes.

The parameter extraction service type defines a single standardized subservice type, i.e. the parameter extraction subservice type.

6.26.1.2 Parameter extraction subservice

The parameter extraction subservice type includes the capability to add parameter extraction definitions, delete parameter extraction definitions and report the content of parameter extraction definitions.

6.26.2 Service layout

6.26.2.1 Subservice

a. Each parameter extraction service shall contain at least one parameter extraction subservice.

6.26.2.2 Application process

a. Each application process shall host at most one parameter extraction subservice provider.

6.26.3 Parameter extraction definition

a. The parameter extraction subservice configuration policy to apply when starting and restarting the parameter extraction subservice shall be declared when specifying the subservice.

NOTE The choice to define default parameter extraction definitions in the on-board software image and/or other non-volatile memory structure is mission dependent.

b. The maximum number of parameter extraction definitions that the parameter extraction subservice can contemporaneously process at any time shall be declared when specifying that parameter extraction subservice.

c. Each parameter extraction definition shall contain:

1. the identifier of the on-board parameter to be used to store the extracted data;
2. the source application process identifier of the report from which the parameter is to be extracted;
3. the message type identifier, composed of the service type identifier and the message subtype identifier, of the report from which the parameter is to be extracted;
4. the housekeeping parameter structure identifier (in case of housekeeping) or the event identifier (in case of Event report) of the report from which the parameter is to be extracted;
5. the offset in bytes of the data to be extracted, starting from the beginning of the telemetry packet data field.

NOTE Each parameter extraction definition is uniquely identified by the identifier of the on-board parameter used to store its extracted data.

d. The list of on-board parameters that can be used for storage of extracted data shall be declared when specifying the parameter extraction subservice.

NOTE These on-board parameters are expected to be dedicated to the parameter extraction subservice usage and not be written by any other application.

6.26.4 Parameter extraction logic

a. When the parameter extraction subservice detects the occurrence of a report matching one or more of its parameter extraction definitions, it shall extract the corresponding data from this report and copy it into the selected on-board parameters.

NOTE the number of bytes to be extracted from the offset is implicitly based on the type and length of the specified on-board parameter. For parameter length definitions, refer to clause 6.20.

b. A report shall be considered matching a parameter extraction definition when the following condition is fulfilled:

1. For a housekeeping report, when its source application process identifier, message type identifier and housekeeping parameter structure identifier are equal to those contained in the parameter extraction definition.
2. For an event report, when its source application process identifier, message type identifier and event identifier are equal to those contained in the parameter extraction definition.

6.26.5 Managing parameter extraction definitions

6.26.5.1 Add parameter extraction definitions

a. The parameter extraction subservice shall provide the capability to add parameter extraction definitions in the parameter extraction definitions table.

NOTE 1 The corresponding requests are of message type "TC[26,1] add parameter extraction definitions".

NOTE 2 For the capability to delete parameter statistics definitions, refer to clause 6.26.5.2.

b. Each request to add parameter extraction definitions shall contain an ordered list of one or more instructions to add a parameter extraction definition.

c. Each instruction to add a parameter extraction definition shall a single parameter extraction definition.

NOTE The content of such parameter extraction definition are specified in clause 6.26.3.c.

d. The parameter extraction subservice shall reject any request to add a parameter extraction definition if:

1. that request contains an instruction which refers to a parameter identifier that cannot be used for storage of extracted data.
2. that request contains an instruction which implies adding a parameter extraction definition, but the maximum number of definitions supported has been already reached.

NOTE 1 correctness of SID/EID, APID, Type, Subtype and possible combinations are user responsibility.

NOTE 2 the definition of the check related to a parameter identifier already present in the parameter extraction table is mission dependent. The parameter identifier is expected to be used as index in the parameter extraction definitions table, and therefore unique.

e. For each request to add parameter extraction definitions that is rejected, the parameter extraction subservice shall generate a failed start of execution notification.

f. For each request to add parameter extraction definitions that contains only valid instructions, the parameter extraction subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to add a parameter extraction definition, the parameter extraction subservice shall add the parameter extraction definition to the parameter extraction definitions table.

6.26.5.2 Delete parameter extraction definitions

a. The parameter extraction subservice shall provide the capability to delete parameter extraction definitions from the parameter extraction definitions table.

NOTE 1 The corresponding requests are of message type "TC[26,2] delete parameter extraction definitions".

NOTE 2 For the capability to add parameter extraction definitions, refer to clause 6.26.5.1.

b. Each request to delete parameter extraction definitions shall contain exactly one of the following:

12. one or more instructions to delete a parameter extraction definition;

13. a single instruction to delete all parameter extraction definitions.

NOTE The instructions to delete all parameter extraction definitions contain no argument

c. Each instruction to delete a parameter extraction definition shall contain:

1. the on-board parameter identifier used by the parameter extraction definition to be deleted.

NOTE this parameter identifier uniquely identifies the parameter extraction definition.

d. The parameter extraction subservice shall reject any request to delete parameter extraction definitions if:

1. that request contains an instruction which refers to an on-board parameter identifier that is not used by any parameter extraction definition.

e. For each request to delete parameter extraction definitions that is rejected, the parameter extraction subservice shall generate a failed start of execution.

f. For each request to delete parameter extraction definitions that contains only valid instructions, the parameter extraction subservice shall execute those instructions in the order of their appearance in that request.

g. For each valid instruction to delete a parameter extraction definition, the parameter extraction subservice shall remove the parameter extraction definition from the parameter extraction definitions table.

h. For each valid instruction to delete all parameter extraction definitions, the parameter extraction subservice shall clear the parameter extraction definitions table.

6.26.5.3 Report parameter extraction definitions

a. The parameter extraction subservice shall provide the capability to report the parameter extraction definitions.

NOTE The corresponding requests are of message type "TC[26,3] report parameter extraction definitions ". The responses are data reports of message type "TM[26,4] parameter extraction definitions report"

b. Each request to report the parameter extraction definitions shall contain exactly one instruction to report the parameter extraction definitions.

NOTE the instruction to request the report of the parameter extraction definitions contains no argument

c. For each valid instruction to report the parameter extraction definitions, the parameter extraction subservice shall:

1. generate, for each parameter extraction definition, a single parameter extraction definition notification that includes the complete parameter extraction definition.

NOTE The contents of such parameter extraction definition are specified in clause 6.26.3.c.

d. For each valid request to report the parameter extraction definitions, the parameter extraction subservice shall generate a single parameter extraction definition report that includes all related parameter extraction definitions notifications.

6.26.6 Subservice observables

a. The following observables shall be defined for the parameter extraction subservice:

1. number of used parameter extraction definitions.

NOTE the number of unused parameter extraction definitions can be deduced from the maximum number of parameter extraction definitions and the number of used parameter extraction definitions.

6.27 ST[27] critical packet log management

6.27.1 Scope

6.27.1.1 General

The critical packet log management service type provides the capability:

- to store critical packets generated by other on-board services in a dedicated buffer on-board whose content should never be lost
- to make those critical packets available for downlink on request from ground
- to ensure that critical packets not yet downlinked by ground can not be deleted from this dedicated buffer

It is noted that the capabilities offered by the critical packet log management service type are fully independent from those provided by the on-board storage and retrieval service type specified in ST[15].

The critical packet log management service type defines a single standardized subservice type, i.e. the critical packet log management subservice type.

6.27.1.2 Critical packet log management subservice

The critical packet log management subservice type provides the ground operator with a rapid access to the history of event packets with mission specified severity and the negative telecommand acknowledgement reports specified for the mission.

The on-board buffer dedicated to the storage of the critical packets by the critical packet log management subservice, referred to as critical packet log, is managed using a 'bounded' policy: storage terminates and new critical packets are discarded when the buffer is full.

Only full retrieval mode is provided to ground by the critical packet log management subservice so that the entire content of the buffer at the time of the request start of execution is downlinked, with the highest priority and segregated from the real-time channels.

The critical packet log management subservice guarantees that only packets already downlinked to ground can be deleted from the critical packet log.

6.27.2 Service layout

6.27.2.1 Service

a. Each space system shall contain exactly one critical packet log management service.

6.27.2.2 Subservice

6.27.2.2.1 Critical packet log management subservice

a. Each critical packet log management service shall contain exactly one critical packet log management subservice.

6.27.2.3 Application process

a. Each application process shall host at most one critical packet log management subservice provider.

6.27.3 Accessibility

6.27.3.1 Request verification

a. The list of routing and reporting, acceptance and reporting and execution reporting subservices that generate the failed verification reports processed by the critical packet log management subservice shall be declared when specifying that critical packet log management subservice.

NOTE routing and reporting, acceptance and reporting and execution reporting subservices are specified in clause 6.1.

b. The critical packet log management subservice shall be able to detect and process all failed verification reports generated by the associated routing and reporting, acceptance and reporting and execution reporting subservices.

6.27.3.2 Event reporting

a. The list of event reporting subservices that generate the event reports processed by the critical packet log management subservice shall be declared when specifying that critical packet log management subservice.

NOTE The event reporting subservice is specified in clause 6.5.

b. The critical packet log management subservice shall be associated to at least one event reporting subservice.

c. The critical packet log management subservice shall be able to detect and process all event reports generated by the associated event reporting subservices.

6.27.4 Critical packet log management subservice

6.27.4.1 Critical packet log management configuration

a. The critical packet log management subservice shall provide the capability to store in the critical packet log all event reports of medium and high severity.

NOTE These reports are of message types TM[5,3] and TM[5,4]

b. Whether all low severity event reports are also stored in the critical packet log shall be declared when specifying the critical packet log management subservice.

NOTE These reports are of message type TM[5,2].

c. Whether failed acceptance, failed start of execution, failed progress of execution, failed completion of execution and failed routing verification reports are also stored in the critical packet log shall be declared when specifying the critical packet log management subservice.

NOTE These reports are respectively of message types TM[1,2], TM[1,4], TM[1,6], TM[1,8] and TM [1,10].

d. The maximum number of packets that the critical packet log can contain shall be declared when specifying the critical packet log management subservice.

NOTE This maximum number is driven by mission specific characteristics such as non-coverage period duration, level of on-board autonomy and level of event report usage.

e. The virtual channel used by the critical packet log management subservice to transmit the packets retrieved from the critical packet log shall be declared when specifying that subservice.

NOTE It is recommended to downlink the packets in the critical packet log in a dedicated non-real time virtual channel with a higher priority that the ones used to return real-time data. Using a real-time virtual channel for this has the drawback that the associated packets fail the time plausibility check if applied by the ground system.

6.27.4.2 Storing critical packets in log

a. For each critical packet that it receives, the critical packet log management subservice shall store it in the critical packet log.

NOTE this is ensured by maintaining the critical packet storing function always enabled

b. The critical packet log management subservice shall manage the critical packet log with a “bounded type” policy: storage terminates and new critical packets are discarded when the maximum number of packets has been reached.

c. The content of the critical packet log shall be made persistent, such that the critical packet log remains available also in case of a severe spacecraft anomaly, leading to software reboot or processor redundancy switch-over.

NOTE This is to ensure that the critical packet log is accessible by ground operators under any circumstance and that its content is never lost.

6.27.4.3 Downlinking critical packets from log

6.27.4.3.1 Critical packet log downlink process

a. The critical packet log management subservice shall maintain a status indicating whether the critical packet log downlink is “in-progress“ or “on-hold“.

NOTE This status is named “critical packet log downlink status”.

b. When the critical packet log downlink status is “in-progress”, the critical packet log management subservice shall:

1. retrieve all critical packets stored in the critical packet log in chronological order;
2. route these packets to the virtual channel associated with the critical packet log;
3. when the last packet contained in the critical packet log is reached, set the critical packet log downlink status to “on-hold”.

c. The critical packet log management subservice shall be able to record newly received critical packets while the critical packet log downlink status is “in-progress”.

6.27.4.3.2 Downlink the critical packet log

a. The critical packet log management subservice shall provide the capability to downlink the critical packet log.

NOTE The corresponding requests are of message type “TC[27,1] downlink critical packet log”.

b. Each request to downlink the critical packet log shall contain exactly one instruction to downlink the critical packet log.

NOTE The instruction to downlink the critical packet log contains no arguments.

c. The critical packet log management subservice shall reject any request to downlink the critical packet log content if:

1. the critical packet log downlink status is “in-progress”;
2. the critical packet log clearing status is “in-progress”.

d. For each request to downlink the critical packet log that is rejected, the critical packet log management subservice shall generate a failed start of execution notification.

e. For each request that contains a valid instruction to downlink the critical packet log, the critical packet log management subservice shall:

1. set the critical packet log downlink status to “in-progress”;
2. start the critical packet log downlink process.

6.27.4.4 Clearing downlinked packets from log

6.27.4.4.1 Critical packet log clearing process

a. The critical packet log management subservice shall be able to record newly received critical packets while the critical packet log clearing is in progress.

6.27.4.4.2 Clear downlinked packets from critical packet log

a. The critical packet log management subservice shall provide the capability to clear the downlinked packets from the critical packet log.

NOTE The corresponding requests are of message type "TC[27,2] clear downlinked packets from critical packet log".

b. Each request to clear downlinked packets from the critical packet log shall contain exactly one instruction to clear downlinked packets from the critical packet log.

NOTE The instruction to clear downlinked packets from the critical packet log contains no arguments.

c. The critical packet log management subservice shall reject any request to clear downlinked packets from the critical packet log if any of the following conditions occurs:

1. the critical packet log downlink status is “in-progress”;
2. the critical packet log clearing status is “in-progress”.

d. For each request to clear downlinked packets from the critical packet log that is rejected, the critical packet log management subservice shall generate a failed start of execution notification

e. For each request that contains a valid instruction to clear downlinked packets from the critical packet log, the critical packet log management subservice shall:

1. set the critical packet log clearing status to “in-progress”;
2. delete all critical packets stored in the critical packet log already downlinked to ground;
3. when the last previously downlinked critical packet has been deleted, set the critical packet log clearing status to “on-hold”.

6.27.4.5 Subservice observables

a. The following observables shall be defined for the critical packet log management subservice:

1. number of packets in the critical packet log;
2. number of packets discarded because the critical packet log is full;
3. generation time of the first logged packet;
4. generation time of the last logged packet;

5. source application identifier, service type identifier and message subtype identifier of the first logged packet;
6. source application identifier, service type identifier and message subtype identifier of the last logged packet;
7. event identifier of the first logged event packet;
8. event identifier of the last logged event packet.

NOTE These observables are not expected to be updated during the processing of ground requests, only at request completion

7
Space to ground interface requirements

7.1 Introduction

7.1.1 Packets

This Standard promotes using space packets compliant to the CCSDS space packet protocol to transport the PUS messages. It does not prescribe the protocol used to transport requests initiated on-board and reports destined for on-board.

In this Standard:

- a "**telecommand packet**" is the data unit that is used to carry a service request from an application process on the ground to an application process on-board;
- a "**telemetry packet**" is the data unit that is used to carry a service report from an application process on board to an application process on the ground.

The initiation of a request by a subservice user on the ground results in the transmission of a telecommand packet to the on-board subservice provider, the reception of which initiates the execution of the corresponding activity.

The initiation of a report by an on-board subservice provider results in the sending of a telemetry packet to a subservice user.

The specification of the activities performed by the ground as a subservice user (e.g. to generate requests or to process reports) is beyond the scope of this Standard.

Some of the PUS services defined in this Standard imply an exchange of messages between on-board application processes. The mechanisms used to exchange such messages on-board are mission-dependent and therefore outside the scope of this Standard.

The data format for telemetry packets and for telecommand packets is the "**space packet**" specified in CCSDS 133.0-B-1.

Clause 7.3.14 specifies how the common fields of a space packet are used for a telemetry or telecommand packet.

Service-specific fields are specified in clause 8.

Clauses 7.3.14 and 8 uses the standard PUS field types specified in clause 7.3.

This Standard does not exclude the use of other packet protocols that are fully compatible with its requirements for telemetry and telecommand packets.

7.1.2 Packet transport

7.1.2.1 Introduction

The telemetry or telecommand systems through which the packets are transported are layered, with each layer drawing upon a well-defined set of services provided by the layer below and providing a similarly well-defined set of services to the layer above (see ECSS-E-ST-50-03 and ECSS-E-ST-50-04).

7.1.2.2 Telemetry link

On the **telemetry link,** the physical channel can be shared between multiple Master Channels, for example, when one spacecraft acts as a relay for another spacecraft such as in a planetary orbiter/lander mission (see ECSS-E-ST-50-03). Each master channel is identified by a unique spacecraft identifier field in the telemetry frame header. However, for a typical mission comprising a single spacecraft, all the frames on a physical channel have the same value for the spacecraft identifier, so there is only one master channel on the physical channel.

Some spacecraft can use several physical channels for their telemetry data and can further differentiate the data transmitted on those channels by using different frame formats (for examples, see ECSS-E-ST-50-03 and CCSDS 732.0-B-2), or by other means outside the scope of this Standard.

Virtual Channels provide a technique for multiple on-board packet sources (application processes) to share the finite capacity of a physical link through multiplexing. Each virtual channel is identified by a unique virtual channel identifier field in the telemetry frame header and the frames from different virtual channels are multiplexed together on a master channel (see Figure 7-1). Up to eight virtual channels (refer to ECSS-E-ST-50-03) or up to 64 virtual channels (refer to CCSDS 732.0-B-2) can be supported on a master channel. Virtual channels can be used for a variety of purposes, such as:

- flow control to prevent long packets from blocking the physical channel;
- separating different types of data for stream splitting on the ground. For example, separating low-rate engineering data from high-rate science data for onward transmission on the ground or separating real-time data from playback data.

Whilst a long packet is being transmitted, the transmission of any other packets for the same virtual channel is delayed. To overcome this, a mission may define a maximum length for the telemetry packets to use by the mission, which is considerably shorter than the maximum length supported by the packet protocol used.

7.1.2.3 Telecommand link

On the telecommand link, the physical channel can also be shared between multiple master channels and virtual channels (see ECSS-E-ST-50-04). In addition, an optional identifier, called the multiplexer access point identifier (MAP ID), can be used to create multiple streams of telecommand data within a virtual channel. All the transfer frames on a given virtual channel with the same MAP ID constitute a MAP channel. Up to sixty-four MAP channels can be supported on a virtual channel. The choice of multiplexing algorithm and the

allocation of priorities to the individual virtual channels and MAPs is implementation dependent. For example, MAPs can be used for:

- flow control purposes;
- telecommand prioritization i.e. a telecommand on a high-priority MAP can be transmitted before a telecommand arriving earlier on a lower-priority MAP;
- telecommand routing as part of the telecommand decoding process.

Whilst there is a theoretically huge multiplexing capability available, real implementations generally use a very modest repertoire of MAP ID and virtual channel ID assignments.

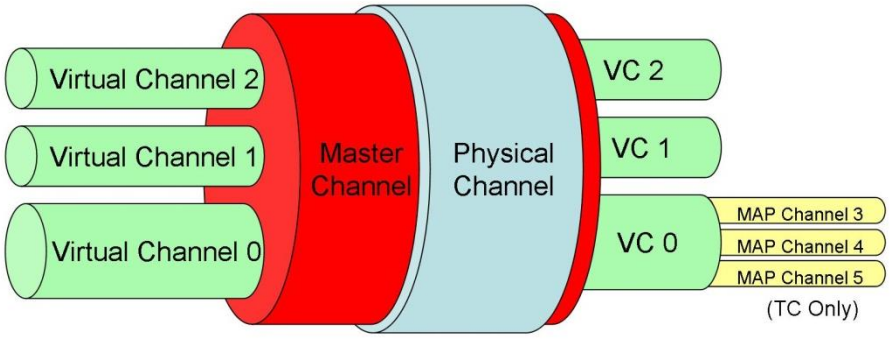

Figure 7-1 Sharing a physical channel

7.2 Convention

7.2.1 Structure diagram

In the remainder of this Standard, sequences of packet fields are presented in structure diagrams as shown in Figure 7-2.

repeated N times

N	packet field 1	packet field 2
unsigned integer	Boolean	enumerated

optional

Figure 7-2 An example of a packet field structure diagram

For each field contained in the corresponding structure:

- the field name is specified in the first row of the diagram;
- the field type is specified in the second row.

Where the presence of a field, or group of fields, is optional, this is indicated by the text "*optional*" below the corresponding fields. A field or group of fields is

optional if its presence is determined at the level of the mission, application process or service instance.

The omission of an optional field can imply that the value is known by both the subservice provider and the subservice user. For example, the subservice provider can use a fixed value or a "current value" which can be set by the subservice user through other means. The subservice provider can even use the values of other preceding fields in the request or report to access a fixed or modifiable look-up table in which the values are contained.

Where a field, or group of fields, constitutes an entry in a fixed-length or a variable-length array, this is indicated above the table by the text "*repeated N times*", where N is the number of repetitions within the array. In the case of a variable-length array, N is given explicitly at the start of the array (N is set to 0 when the array is empty). In the case of a fixed-length array, N is known implicitly for the mission.

7.2.2 Bit-field numbering

Each bit in a field (a n-bit field) is identified and numbered from left to right as follows:

- The first bit, i.e. the leftmost justified bit on a figure, i.e. the most significant bit, is called "Bit 0";
- The second bit is called "Bit 1";
- and so on, up to "Bit N-1".

A group of 8 adjacent bits is called an octet or a byte.

7.3 Packet field type code

7.3.1 General

a. Each packet field shall be associated to a packet field code that indicates the data type of any value carried by that packet field.

NOTE The packet field code specified in this Standard are uniquely identified by the combination of:

- a packet field type code (PTC), and
- a packet field format code (PFC).

The interpretation of each PFC is fully and only dependent on the associated PTC.

b. Tailoring this Standard for a mission, for each new message type defined for that mission, the packet field type code of each field of that new message type shall be declared when specifying that message type.

c. Tailoring this Standard for a mission, for each message type field that packet field format code is unknown, the packet field format code of that field shall be declared when specifying the application process that uses the related message type.

d. The PTC specified in Table 7-1 shall be used to declare the PTC of each packet field.

Table 7-1 PTC – packet field type code

PTC	simple type correspondence
1	Boolean
2	enumerated
3	unsigned integer
4	signed integer
5	real
6	bit-string
7	octet-string
8	character-string
9	absolute time
10	relative time
11	deduced
12	packet

e. The PTC of each packet field shall be declared when specifying the structure of each packet type.

7.3.2 Boolean

a. Each packet field used to carry Boolean values shall be of PTC 1.

NOTE 1 A Boolean value > 0 denotes TRUE.

NOTE 2 A Boolean value = 0 denotes FALSE.

b. The PFCs specified in Table 7-2 shall be used for packet fields carrying Boolean values.

Table 7-2 PFC for Boolean values

PFC	format definition
0	1-bit Boolean parameter value
n > 1	The PFC identifies the length in bits of the Boolean parameter value, e.g. PFC = 8 means an 8-bits Boolean parameter value.

7.3.3 Enumerated

a. Each packet field used to carry enumerated values shall be of PTC 2.

NOTE 1 An enumerated value is an unsigned integer value that can be involved in logical and comparative expressions but not in numeric and relational expressions.

NOTE 2 An enumerated value has a meaning that is interpreted as a character-string value. An error code is a typical example (e.g. 0 means "unchecked", 3 means "invalid").

b. The PFCs specified in Table 7-3 shall be used for packet fields carrying enumerated values.

Table 7-3 PFC for enumerated values

PFC	format definition
1 to 64	The PFC identifies the length in bits of the enumerated parameter, e.g. PFC = 1 means one-bit parameter value.

7.3.4 Unsigned integer

a. Each packet field used to carry unsigned integer values shall be of PTC 3.

b. Each unsigned integer value shall be encoded with Bit 0 being the most significant bit (MSB) and Bit N-1 the least significant bit (LSB).

c. The PFCs specified in Table 7-4 shall be used for packet fields carrying unsigned integer values.

Table 7-4 PFC for unsigned integer values

PFC	format definition	lowest value	highest value
0 to 12	(PFC + 4) bits, unsigned	0	$2^{PFC+4}-1$ (15 *to* 65 535)
13	3 octets, unsigned	0	$2^{24}-1$ (16 777 215)
14	4 octets, unsigned	0	$2^{32}-1$ ($\approx 4{,}3 \times 10^{9}$)
15	6 octets, unsigned	0	$2^{48}-1$ ($\approx 2{,}8 \times 10^{14}$)
16	8 octets, unsigned	0	$2^{64}-1$ ($\approx 18{,}5 \times 10^{18}$)
17	1 bit, unsigned	0	1
18	2 bits, unsigned	0	3
19	3 bits, unsigned	0	7

7.3.5 Signed integer

a. Each packet field used to carry signed integer values shall be of PTC 4.

b. Bit 0 of each signed integer parameter shall be used to determine the sign of the parameter value.

NOTE 1 Bit 0 = 0 denotes a positive value.

NOTE 2 Bit 0 = 1 denotes a negative value.

NOTE 3 Negative values are represented as 2's complement of the absolute value.

c. The PFCs specified in Table 7-5 shall be used for packet fields carrying signed integer values.

Table 7-5 PFC for signed integer values

PFC	format definition	lowest value	highest value
0 to 12	(PFC + 4) bits, signed	-2^{PFC+3} $(-8\ to\ -32\,768)$	$2^{PFC+3}-1$ $(7\ to\ 32\,767)$
13	3 octets, signed	-2^{23} $(-8\,388\,608)$	$2^{23}-1$ $(8\,388\,607)$
14	4 octets, signed	-2^{31} $(\approx -2{,}15\times10^{9})$	$2^{31}-1$ $(\approx 2{,}15\times10^{9})$
15	6 octets, signed	-2^{47} $(\approx -1{,}4\times10^{14})$	$2^{47}-1$ $(\approx 1{,}4\times10^{14})$
16	8 octets, signed	-2^{63} $(\approx -9{,}2\times10^{18})$	$2^{63}-1$ $(\approx 9{,}2\times10^{18})$

7.3.6 Real

a. Each packet field used to carry real values shall be of PTC 5.

b. The PFCs specified in Table 7-6 shall be used for packet fields carrying real values.

Table 7-6 PFC for real values

PFC	format definition
1	4 octets simple precision format (IEEE)
2	8 octets double precision format (IEEE)
3	4 octets simple precision format (MIL-STD)
4	6 octets extended precision format (MIL-STD)
NOTE 1	The IEEE simple precision and double precision formats are defined in "IEEE 754 Standard for Binary Floating-point Arithmetic" (Reference [7]), see also annex 0.
NOTE 2	The MIL-STD simple precision and extended precision formats are defined in the "Military Standard Sixteen-Bit Computer Instruction Set Architecture" MIL-STD-1750a, 2nd July 1980 (Reference [8]), see also annex A.2.

7.3.7 Bit-string

a. Each packet field used to carry bit-string values shall be of PTC 6.

b. The PFCs specified in Table 7-7 shall be used for packet fields carrying bit-string values:

Table 7-7 PFC for bit-string values

PFC	format definition
0	variable-length bit-string
n > 0	fixed-length bit-string with a number of bits equal to PFC
NOTE	The meaning and interpretation of a bit-string value is application process specific.

c. The variable-length bit-string shall have the structure specified in Figure 7-3.

variable-length bit-string	
length	data
unsigned integer	N bits
NOTE	The packet field code "N bits" means that a value carried in the data field of a variable-length bit-string has a fixed number of bits that equals to the value carried in the corresponding length field.

Figure 7-3 PTC 6 PFC 0 structure

d. For each application process that uses variable-length octet-strings, the PFC of the length field of the variable-length bit-string format shall be declared when specifying that application process.

e. Each spare field of a telemetry or a telecommand packet shall be of fixed-length PTC 6.

f. For each spare field of a telemetry or a telecommand packet, all bits of that field shall be set to zero.

g. For each packet field containing a fixed-length bit-string whose length is deduced, the definition used to deduce that length shall be declared when specifying the related packet field type.

NOTE The deduced length corresponds to a fixed length PFC.

h. For each packet field containing a fixed-length bit-string whose length is deduced, the deduction of the length shall only result from the content of one or more preceding fields of the same packet, of one or more mission constants or a combination of both.

7.3.8 Octet-string

a. Each packet field used to carry octet-string values shall be of PTC 7.

b. The PFCs specified in Table 7-8 shall be used for packet fields carrying octet-string values.

Table 7-8 PFC for octet-string values

PFC	format definition
0	Variable-length octet-string
n > 0	Fixed-length octet-string with a number of octets equal to PFC
NOTE	The meaning and interpretation of an octet-string value is application process specific.

c. The variable-length octet-string shall have the structure specified in Figure 7-4.

variable-length octet-string	
length	data
unsigned integer	N octets
NOTE	The packet field code "N octets" means that a value carried in the data field of a variable-length octet-string has a fixed number of octets that equals to the value carried in the corresponding length field.

Figure 7-4 PTC 7 PFC 0 structure

d. For each application process that uses variable-length octet-strings, the PFC of the length field of the variable-length octet-string format shall be declared when specifying that application process.

e. For each packet field containing a fixed-length octet-string whose length is deduced, the definition used to deduce that length shall be declared when specifying the related packet field type.

NOTE The deduced length corresponds to a fixed length PFC.

f. For each packet field containing a fixed-length octet-string whose length is deduced, the deduction of the length shall only result from the content of one or more preceding fields of the same packet, of one or more mission constants or a combination of both.

7.3.9 Character-string

a. Each packet field used to carry character-string values shall be of PTC 8.

b. The values that character-string parameters can take shall be sequences of visible characters.

NOTE 1 The visible characters are defined in ANSI X3.4 and represented by their ASCII code on one octet. The visible characters consist of:

- the ASCII code 0x20 "nonprinting spacing character", and
- the printable graphic characters, i.e. from ASCII code 0x21 to ASCII code 0x7E.

NOTE 2 In the case a character-string value is of length lower than the length specified by the PFC, nonprinting spacing characters (ASCII code 0x20) are padded at the end of the value to produce the required character-string length.

c. The PFCs specified in Table 7-9 shall be used for packet fields carrying character-string values.

Table 7-9 PFC for character-string values

PFC	format definition
0	Variable-length character-string
n > 0	Fixed-length character-string with a number of characters equal to PFC
NOTE	The meaning and interpretation of a character-string value is application process specific.

d. The variable-length character-string format shall have the structure specified in Figure 7-5:

variable-length character-string	
length	data
unsigned integer	N characters

NOTE 1 The packet field code "N character" means that a value carried in the data field of a variable-length character-string has a fixed number of characters that equals to the value carried in the corresponding length field.

NOTE 2 Each character of the value field is represented in ASCII on one octet.

Figure 7-5 PTC 8 PFC 0 structure

e. For each application process that uses variable-length character-strings, the PFC of the length field of the variable-length character-string format shall be declared when specifying that application process.

f. For each packet field containing a fixed-length character-string whose length is deduced, the definition used to deduce that length shall be declared when specifying the related packet field type.

NOTE The deduced length corresponds to a fixed length PFC.

g. For each packet field containing a fixed-length character-string whose length is deduced, the deduction of the length shall only result from the content of one or more preceding fields of the same packet, of one or more mission constants or a combination of both.

7.3.10 Absolute time

a. Each packet field used to carry absolute time values shall be of PTC 9.

b. Each absolute time parameter value shall be a positive time offset that is a number of seconds and fractions of a second from a given epoch.

NOTE 1 If the CUC format is used, either the standard CCSDS epoch of 1958 January 1 or an Agency defined epoch can be used. In the latter case, the parameter corresponds to a free-running counter that is converted on ground using the applicable time correlation coefficients.

NOTE 2 The CUC format is specified in CCSDS 301.0-B-4. The CCSDS offers means to define CUC coarse time values using 1 to 7 octets and fine time values using 1 to 10 octets. This Standard implements means to define CUC coarse time values using 1 to 4 octets and fine time values using 1 to 10 octets.

c. If the absolute time parameter has CDS format, the standard CCSDS epoch of 1958 January 1 shall be used.

NOTE The CDS format is specified in CCSDS 301.0-B-4.

d. The PFCs specified in Table 7-10 shall be used for packet fields carrying absolute time values.

Table 7-10 PFC for absolute time values

PFC	format definition
0	Explicit definition of time format (CUC or CDS), i.e. including the P-field
1	2 octets day CDS format without a µs field The parameter field has a length equal to 6 octets.
2	2 octets day CDS format with a µs field The parameter field has a length equal to 8 octets.
3 to 18	CUC format with: The number of octets of coarse time equals the integer quotient of (PFC number + 1) divided by 4, and The number of octets of fine time equals the remainder of (PFC number + 1) divided by 4. The P-field is implicit and derived from the PFC.
19 to 46	CUC format with: The number of octets of coarse time equals the integer quotient of (PFC number -12) divided by 7, and The number of octets of fine time equals 4 + the remainder of (PFC number -12) divided by 7. The P-field is implicit and derived from the PFC.

NOTE 1 The CUC and CDS time formats are defined in CCSDS 301.0-B-4.

NOTE 2 The CDS Format with µs, i.e. PFC = 2 has the structure shown in figure below. The value of day is an unsigned integer in the range 0 *to* $2^{16} - 1$.

day	ms of day	µs of ms
2 octets	4 octets	2 octets

NOTE 3 The full CUC format, i.e. PFC 18 has the structure shown in figure below. The time in seconds from the given Agency epoch is given by $t = C1 \times 256^3 + C2 \times 256^2 + C3 \times 256 + C4 + F1 \times 256^{-1} + F2 \times 256^{-2} + F3 \times 256^{-3}$.

C1	C2	C3	C4	F1	F2	F3
1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet

7.3.11 Relative time

a. Each packet field used to carry relative time values shall be of PTC 10.

b. Each relative time parameter value shall be a positive or a negative time offset expressed according to the selected PFC, from the occurrence time of an event whose identification can be derived from other parameters in the packet (identifying a type of on-board event) or between two absolute times.

c. The PFCs specified in Table 7-11 shall be used for packet fields carrying relative time values.

Table 7-11 PFC for relative time values

PFC	format definition
1	2 octets day CDS format without a µs field The parameter field has a length equal to 6 octets.
2	2 octets day CDS format with a µs field The parameter field has a length equal to 8 octets.
3 to 18	CUC format with: The number of octets of coarse time equals the integer quotient of (PFC number + 1) divided by 4, and The number of octets of fine time equals the remainder of (PFC number + 1) divided by 4. The P-field is implicit and derived from the PFC.
NOTE The full CUC format, i.e. PFC 18 has the structure shown in figure below. A positive time offset is given by $t = C1 \times 256^3 + C2 \times 256^2 + C3 \times 256 + C4 + F1 \times 256^{-1} + F2 \times 256^{-2} + F3 \times 256^{-3}$ where C1 is in the range 0 to 127.	

C1	C2	C3	C4	F1	F2	F3
1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet

7.3.12 Deduced

a. Each packet field whose structure and format is deduced shall be of PTC 11 PFC 0.

b. For each packet field whose structure and format is deduced, the definition used to deduce that structure and format shall be declared when specifying the related packet field type.

c. For each packet field whose structure and format is deduced, the deduction of the structure and format shall only result from the content of one or more preceding fields of the same packet, of one or more mission constants or a combination of both.

7.3.13 Packet

a. Each packet field used to carry packets shall be of PTC 12.

b. The PFCs specified in Table 7-12 shall be used for packet fields carrying packets.

Table 7-12 PFC for packet values

PFC	format definition
0	CCSDS telemetry packet compliant with this Standard

PFC	format definition
1	CCSDS telecommand packet compliant with this Standard
NOTE	For PFC 0 and PFC 1, refer to clause 7.3.14.

7.3.14 Key elements size

a. For the key elements defined in this standard, sizes from table 7-13 shall be adopted.

NOTE This is to ensure inter-operability of PUS-C implementations from different providers

Table 7-13 Key elements size

Key element	Size
Number of instructions included in request	2 bytes
Boolean	1 byte (exception Secondary Header: 1 bit)
Collection interval	4 bytes
Parameter ID	4 bytes
Event ID	4 bytes
HK Report Structure ID	2 bytes
Memory ID	2 bytes
Memory start address, Register address, Memory Bit-Mask	4 bytes
Sub-schedule ID	2 bytes
Group ID	2 bytes
PMON ID	2 bytes
FMON ID	2 bytes
APID	2 bytes (exception Primary Header: 11 bits)
PacketStore ID	2 bytes
OBCP ID	2 bytes
CPDU ID	1 byte
CFDP entity ID	2 bytes
CFDP transaction seq number	4 bytes
Data Source ID	2 bytes

7.4 The CCSDS Space Packet

7.4.1 Overview

The CCSDS Space Packet Protocol is defined in CCSDS 133.0-B-1. The generic structure of a CCSDS space packet is shown in Figure 7-6.

packet primary header							packet data field	
packet version number	packet ID			packet sequence control		packet data length	packet secondary header	user data field
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count or packet name			
3 bits	1 bit	1 bit	11 bits	2 bits	14 bits	16 bits	variable	variable
2 octets				2 octets		2 octets	1 to 65536 octets	

Figure 7-6 The space packet structure

The *packet version number is set to 0 and identifies it as a space packet defined by CCSDS 133. 0-B-1. A space packet is also referred to as a version 1 CCSDS packet*.

The *packet type* bit distinguishes between telemetry packets, for which this bit is set to 0, and telecommand packets, for which this bit is set to 1.

The *secondary header flag* indicates the presence or absence of the packet secondary header. With the exception of spacecraft time packets (refer to clause 6.9.4), all telemetry packets defined in this Standard have a packet secondary header field. With the exception of CPDU command packets (refer to clause 9.3.1), all telecommand packets defined in this Standard have a packet secondary header field.

The *application process ID* uniquely identifies the on-board application process that is source of the telemetry packet and destination of the telecommand packet. Some values of the application process ID field are reserved by the CCSDS standard, making them unavailable for use by PUS services.

The *sequence flags are defined by CCSDS but not used by the space packet protocol. This Standard* uses the binary value "11" for the *sequence flags*, to indicate a stand-alone packet. All telemetry packets and telecommand packets defined within this Standard are stand-alone packets.

The *packet sequence count* is used for telemetry packets. It is incremented by 1 whenever the source application process releases a packet. The packet sequence count wraps around from 2^{14}-1 to zero.

The telecommand packets carry either a *packet sequence count* or a *packet name* to identify them within the same communication session. For the purpose of this Standard, the telecommand *packet sequence count* or *packet name* field carries an

identifier that used in combination with the source identifier specified in clause 7.4.4.1, uniquely identify the telecommand packet.

The *packet data length* field specifies the length of the *packet data field*. The value of the unsigned integer in the packet data length field is one less than the number of octets contained within the packet data field. The length of the entire packet, including the packet primary header, is 6 octets more than the length of the packet data field.

The structure of the *packet data field* depends on the packet type.

- for telemetry packets that field is composed of:
  - the *telemetry packet secondary header* specified in clause 7.4.3.1;
  - the *telemetry user data field* specified in clause 7.4.3.2;
- for telecommand packets that field is composed of:
  - the *telecommand packet secondary header* specified in clause 7.4.4.1;
  - the *telecommand user data field* specified in clause 7.4.4.2.

7.4.2 General

a. Once a telecommand or a telemetry packet has been generated by an application process, shall stay unchanged.

b. The telemetry packet maximum size shall be declared for the mission and not be larger than the one defined in the CCSDS space packet protocol.

c. The telecommand packet maximum size shall be declared for the mission and not be larger than the one defined in the CCSDS space packet protocol.

NOTE The maximum packet size defined in the CCSDS space packet protocol is 65542 bytes.

7.4.3 Telemetry packet data field

7.4.3.1 Telemetry packet secondary header

a. With the exception of the spacecraft time packets specified in clauses 6.9.4.2 and 6.9.4.3, all telemetry packets defined in this Standard shall have a telemetry packet secondary header.

NOTE Missions can also require other types of telemetry packets that do not have telemetry packet secondary header. This is the case of the CCSDS File Delivery Protocol (CFDP) protocol data units and idle packets.

b. Each telemetry packet secondary header shall have the structure specified in Figure 7-7.

TM packet PUS version number	spacecraft time reference status	message type ID		message type counter	destination APID	time	spare
		service type ID	message subtype ID				
enumerated (4 bits)	enumerated (4 bits)	enumerated (8 bits)	enumerated (8 bits)	unsigned integer (16 bits)	enumerated (16 bits)	absolute time	fixed-size bit-string

optional

NOTE The spare field is used to constrain the length of the telemetry packet secondary header to an integral number of words. Its optional presence is driven by requirement 7.4.3.1l.

Figure 7-7 Packet secondary header for telemetry packets

c. Each application process shall set the TM packet PUS version number of each telemetry packet it generates to 2.

NOTE The TM packet PUS version number reflects the different versions of this Standard.

- Version 0 was used by the ESA PUS (ESA PSS-07-101).
- Version 1 corresponds to the ECSS-E-70-41A.

d. Each application process that provides the capability to report the spacecraft time reference status used when time tagging telemetry packets shall set the spacecraft time reference status field of each telemetry packet it generates to the status of the spacecraft time reference used when time tagging that telemetry packet.

NOTE 1 For the capability to report the status of the spacecraft time reference, refer to requirement 5.4.2.1l.

NOTE 2 For the possible values of the spacecraft time reference status, refer to requirement 5.4.3.2c. If the reporting of the spacecraft time reference status is not supported, the spacecraft time reference status field value is set to 0.

NOTE 3 The time tag of the telemetry packet is stored in the time field of the telemetry packet secondary header.

e. Each application process that does not provide the capability to report the status of the spacecraft time reference used when time tagging telemetry packets shall set the spacecraft time reference status field of each telemetry packet it generates to 0.

NOTE For the capability to report the status of the spacecraft time reference, refer to requirement 5.4.2.1l.

f. For each report that it generates, each application process shall set the message type ID field of the corresponding telemetry packet to the message type identifier of that report.

NOTE The structure of the message type ID field is driven by requirement 5.3.4.1c.

g. For each report that it generates, each application process that provides the capability to count the type of generated messages per destination and report the corresponding message type counter shall set the message type counter of the related telemetry packet to the value of the related counter.

NOTE For the capability to count the type of generated messages, refer to requirement 5.4.2.1n.

h. Each application process that does not provide the capability to count the type of generated messages per destination and report the corresponding message type counter shall set the message type counter field of each telemetry packet it generates to 0.

NOTE For the capability to count the type of generated messages, refer to requirement 5.4.2.1n.

i. Each application process shall set the destination APID field of each telemetry packet it generates to the application process identifier of the application process targeted by the related report.

j. The PFC of the time field of telemetry packets shall be declared when specifying the time service used by the spacecraft.

NOTE For the time service, refer to clause 6.9.

k. Each application process shall set the time field of each telemetry packet it generates to the time tag of the related report.

NOTE See requirement 5.4.2.1k.

l. For each application process, the presence and bit-size of the spare field of the telemetry packet secondary header shall be declared when specifying that application process.

m. For each telecommand application data containing fields of absolute time type or relative time type, the PFC of the field shall be declared

n. For each telemetry source data containing fields of absolute time type or relative time type, the PFC of the field shall be declared

7.4.3.2 Telemetry user data field

a. Each telemetry user data field shall have the structure specified in Figure 7-8.

source data	spare	packet error control
deduced	fixed-size bit-string (deduced)	fixed-size bit-string (16 bits)

optional

NOTE 1 The structure and format of the source data is deduced from the message type ID. For each report message type specified in this Standard, the structure and format of the source data is specified in clause 8.

NOTE 2 The spare field is used to constrain the overall packet size to an integral number of words (octets or longer), appropriate to the word size of the application process. Its optional presence is driven by requirement 7.4.3.2c.

NOTE 3 The packet error control field transports an error detection code that is used by the ground system to verify the checksum of the telemetry packet.

Figure 7-8 User data field for telemetry packets

b. The telemetry padding word size used by each application process shall be declared when specifying that application process.

NOTE The telemetry padding word size is the multiple-of-bits number to apply when padding telemetry packets.

c. For each telemetry packet that it generates, each application process shall ensure that the total length of that packet is an integer multiple of the padding word size declared for that application process by including a user data spare field of the minimum bit-size that results in that integer multiple.

d. Whether checksumming telemetry packets is used shall be declared when tailoring this standard to the mission.

NOTE The packet error control field is mandatory in the telemetry user data field. The checksumming calculation to fill the packet error control field is application process dependent.

e. If checksumming telemetry packets is used for the mission, the type of checksum to use, that is either the ISO standard 16-bits checksum or the CRC standard 16-bits, shall be declared when tailoring this standard to the mission.

NOTE 1 For the CRC standard 16-bits checksum algorithm, refer to annex B.1.

NOTE 2 For the ISO standard 16-bits checksum algorithm, refer to annex **Error! Reference source not found.**.

f. If checksumming telemetry packets is used for the mission, for each telemetry packet that it generates, each application process shall:

1. calculate the checksum of that packet, and
2. set the calculated value in the packet error control field of that packet.

NOTE 1 The telemetry packet checksum is calculated when all other fields of the packet are complete, and prior to downloading the packet.

NOTE 2 The telemetry packet checksum is used by the ground system to verify the checksum of the complete telemetry packet.

NOTE 3 Checksumming telemetry packets includes also checksumming large telemetry packets, see clause 6.13.3.

7.4.4 Telecommand packet data field

7.4.4.1 Telecommand packet secondary header

a. With the exception of the CPDU command packet specified in clause **Error! Reference source not found.**, all telecommand packets defined in this Standard shall have a telecommand packet secondary header.

NOTE Missions can also require other types of telecommand packets that do not have telecommand packet secondary header. This is the case of the CCSDS File Delivery Protocol (CFDP) protocol data units or the CCSDS Space Data Link Security (SDLS) extended procedure protocol data units.

b. Each telecommand packet secondary header shall have the structure specified in Figure 7-9.

TC packet PUS version number	acknowledgement flags	message type ID		source APID	spare
		service type ID	message subtype ID		
enumerated (4 bits)	enumerated (4 bits)	enumerated (8 bits)	enumerated (8 bits)	enumerated (16 bits)	fixed-size bit-string
					optional

NOTE The spare field is used to constrain the length of the telecommand packet secondary header to an integral number of words. Its optional presence of is driven by requirement 7.4.4.1g.

Figure 7-9 Packet secondary header for telecommand packets

c. For each request that it issues, each application process shall set the TC packet PUS version number to 2.

NOTE The TC packet PUS version number reflects the different versions of this Standard.

- Version 0 was used by the ESA PUS (ESA PSS-07-101).
- Version 1 corresponds to the ECSS-E-70-41A.

d. For each request that it issues, each application process shall set:

1. the bit 3 of the acknowledgement flags field of the corresponding telecommand packet to:
   (a) 1 if the reporting of the successful acceptance of that request by the destination application process is requested
   (b) 0 otherwise;
2. the bit 2 of the acknowledgement flags field of the corresponding telecommand packet to:
   (a) 1 if successful start of execution of that request by the destination application process is requested;
   (b) 0 otherwise;
3. the bit 1 of the acknowledgement flags field of the corresponding telecommand packet to:
   (a) 1 if the reporting of the successful progresses of execution of that request by the destination application process is requested;
   (b) 0 otherwise;

4. the bit 0 of the acknowledgement flags field of the corresponding telecommand packet to:
   (a) 1 if the reporting of the successful completion of execution of the related request by the destination application process is requested;
   (b) 0 otherwise.

   NOTE 1 For item 1, refer to requirement 5.4.11.2.2a.1.

   NOTE 2 For item 2, refer to requirement 5.4.11.2.2a.2.

   NOTE 3 For item 3, refer to requirement 5.4.11.2.2a.3.

   NOTE 4 For item 4, refer to requirement 5.4.11.2.2a.4.

e. For each request that it issues, each application process shall set the message type ID field of the corresponding telecommand packet to the message type identifier of that request.

   NOTE The structure of the message type ID field is driven by requirement 5.3.4.1c.

f. For each request that it issues, each application process shall set the source APID field to its application process identifier.

g. For each application process that issues requests, the presence and bit-size of the spare field of the telecommand packet secondary header shall be declared when specifying that application process.

7.4.4.2 Telecommand user data field

a. Each telecommand user data field shall have the structure specified in Figure 7-10.

application data	spare	packet error control
deduced	fixed-size bit-string (deduced)	fixed-size bit-string (16 bits)

optional

NOTE 1 The structure and format of the application data is deduced from the message type ID. For each request type specified in this Standard, the structure and format of the application data is specified in clause 6.

NOTE 2 The spare field is used to constrain the overall packet size to an integral number of words (octets or longer), appropriate to the word size of the application process. Its optional presence and deduced size are driven by requirement 7.4.4.2c.

Figure 7-10 User data field for telecommand packets

b. The telecommand padding word size used for each application process shall be declared when specifying that application process.

   NOTE The telecommand padding word size is the multiple-of-bits number to apply when padding telecommand packets.

c. For each telecommand packet that it generates, each application process shall ensure that the total length of that packet is an integer multiple of the padding

word size declared for that application process, by including a user data spare field of the minimum bit-size that results in that integer multiple.

d. The type of checksum to use for checksumming all telecommand packets, which is either the ISO standard 16-bits checksum or the CRC standard 16-bits checksum, shall be declared when tailoring this standard to the mission.

NOTE 1 For the CRC standard 16-bits checksum algorithm, refer to annex B.1.

NOTE 2 For the ISO standard 16-bits checksum algorithm, refer to annex **Error! Reference source not found.**.

e. For each telecommand packet that it generates, each application process shall:

1. calculate the checksum of that packet, and
2. set the calculated value in the packet error control field of that packet.

NOTE 1 The telecommand packet checksum is calculated when all other fields of the packet are complete, and prior to releasing the packet.

NOTE 2 The checksum of each telecommand packet that is received on-board is verified using the checksum that is contained within the packet error control field of the packet. Refer also to requirement 6.1.3.2b.

8 Service type interface requirements

8.1 ST[01] request verification

8.1.1 General

a. Each packet transporting a request verification report shall be of service type 1.

b. The destination ID field of any telemetry packet transporting a request verification report shall be set to the value carried by the source ID field of the related telecommand packet.

NOTE 1 For the destination ID field, refer to requirement 7.4.3.1b.

NOTE 2 For the source ID field, refer to requirement 7.4.4.1b.

NOTE 3 For the corresponding system requirements, refer to requirement 6.1.2.2.1a.

8.1.2 Request and reports

8.1.2.1 TM[1,1] successful acceptance verification report

a. Each telemetry packet transporting a successful acceptance verification report shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.1.4.2.

b. For each telemetry packet transporting a successful acceptance verification report, the source data field shall have the structure specified in Figure 8.1-1.

request ID					
packet version number	packet ID			packet sequence control	
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-1 Successful acceptance verification report

8.1.2.2 TM[1,2] failed acceptance verification report

a. Each telemetry packet transporting a failed acceptance verification report shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.1.4.3.

b. For each telemetry packet transporting a failed acceptance verification report, the source data field shall have the structure specified in Figure 8.1-2.

request ID						failure notice	
packet version number	packet ID			packet sequence control		code	data
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count		
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)	enumerated	deduced

deduced presence

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-2 Failed acceptance verification report

8.1.2.3 TM[1,3] successful start of request execution verification report

a. Each telemetry packet transporting a successful start of request execution verification report shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.1.5.1.1.

b. For each telemetry packet transporting a successful start of request execution verification report, the source data field shall have the structure specified in Figure 8.1-3.

request ID					
packet version number	packet ID			packet sequence control	
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-3 Successful start of request execution verification report

8.1.2.4 TM[1,4] failed start of request execution verification report

a. Each telemetry packet transporting a failed start of request execution verification report shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.1.5.1.2.

b. For each telemetry packet transporting a failed start of request execution verification report, the source data field shall have the structure specified in Figure 8.1-4.

request ID						failure notice	
packet version number	packet ID			packet sequence control		code	data
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count		
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)	enumerated	deduced

deduced presence

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-4 Failed start of request execution verification report

8.1.2.5 TM[1,5] successful progress of request execution verification report

a. Each telemetry packet transporting a successful progress of request execution verification report shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.1.5.2.1.

b. For each telemetry packet transporting a successful progress of request execution verification report, the source data field shall have the structure specified in Figure 8.1-5.

Request ID						step ID
packet version number	packet ID			packet sequence control		
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count	
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)	enumerated

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-5 Successful progress of request execution verification report

8.1.2.6 TM[1,6] failed progress of request execution verification report

a. Each telemetry packet transporting a failed progress of request execution verification report shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.1.5.2.2.

b. For each telemetry packet transporting a failed progress of request execution verification report, the source data field shall have the structure specified in Figure 8.1-6.

Request ID						step ID		
packet version number	packet ID			packet sequence control			code	
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count			
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)	enumerated	enumerated	

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of th the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-6 Failed progress of request execution verification report

8.1.2.7 TM[1,7] successful completion of request execution verification report

a. Each telemetry packet transporting a successful completion of request execution verification report shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.1.5.3.1.

b. For each telemetry packet transporting a successful completion of request execution verification report, the source data field shall have the structure specified in Figure 8.1-7.

request ID					
packet version number	packet ID			packet sequence control	
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-7 Successful completion of request execution verification report

8.1.2.8 TM[1,8] failed completion of request execution verification report

a. Each telemetry packet transporting a failed completion of request execution verification report shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.1.5.3.2.

b. For each telemetry packet transporting a failed completion of request execution verification report, the source data field shall have the structure specified in Figure 8.1-8.

request ID						failure notice	
packet version number	packet ID			packet sequence control		code	data
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count		
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)	enumerated	deduced

deduced presence

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-8 Failed completion of request execution verification report

8.1.2.9 TM[1,10] failed routing verification report

a. Each telemetry packet transporting a failed routing verification report shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.1.3.3.

b. For each telemetry packet transporting a failed routing verification report, the source data field shall have the structure specified in Figure 8.1-9.

request ID						failure notice	
packet version number	packet ID			packet sequence control		code	data
	packet type	secondary header flag	application process ID	sequence flags	packet sequence count		
enumerated (3 bits)	enumerated (1 bit)	Boolean (1 bit)	enumerated (11 bits)	enumerated (2 bits)	unsigned integer (14 bits)	enumerated	deduced

deduced presence

NOTE The request ID field alone cannot be used to identify the request since it does not contain the identifier of the source of that request. That source identifier corresponds to the destination identifier of the secondary header of the related telemetry packet, refer to clause 7.4.3.1.

Figure 8.1-9 Failed routing verification report

8.2 ST[02] device access

8.2.1 General

a. Each packet transporting a device access message shall be of service type 2.

8.2.2 Requests and reports

8.2.2.1 TC[2,1] distribute on/off device commands

a. Each telecommand packet transporting a request to distribute on/off device commands shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.2.4.2.

b. For each telecommand packet transporting a request to distribute on/off device commands, the application data field shall have the structure specified in Figure 8.2-1.

repeated N times

N	on/off device address
unsigned integer	enumerated

Figure 8.2-1 Distribute on/off device commands

8.2.2.2 TC[2,2] distribute register load commands

a. Each telecommand packet transporting a request to distribute register load commands shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.2.5.2.

b. For each telecommand packet transporting a request to distribute register load commands, the application data field shall have the structure specified in Figure 8.2-2.

repeated N times

N	register address	register data
unsigned integer	enumerated	deduced

Figure 8.2-2 Distribute register load commands

8.2.2.3 TC[2,4] distribute CPDU commands

a. Each telecommand packet transporting a request to distribute CPDU commands shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.2.6.2.

b. For each telecommand packet transporting a request to distribute CPDU commands, the application data field shall have the structure specified in Figure 8.2-3.

repeated N1 times

CPDU ID	N1	output line ID	reserved	duration exponential value
enumerated	unsigned integer	enumerated (12 bits)	bit-string (1 bit)	unsigned integer (3 bits)

Figure 8.2-3 Distribute CPDU commands

8.2.2.4 TC[2,5] distribute register dump commands

a. Each telecommand packet transporting a request to *distribute register dump commands* shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.2.5.3.

b. For each telecommand packet transporting a request to *distribute register dump commands*, the application data field shall have the structure specified in Figure 8.2-4.

repeated N times

N	register address
unsigned integer	enumerated

Figure 8.2-4 Distribute register dump commands

8.2.2.5 TM[2,6] register dump report

a. Each telemetry packet transporting a *register dump* report shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.2.5.3.

b. For each telemetry packet transporting a register dump report, the source data field shall have the structure specified in Figure 8.2-5.

repeated N times

N	register address	register data
unsigned integer	enumerated	deduced

Figure 8.2-5 Register dump report

8.2.2.6 TC[2,7] distribute physical device commands

a. Each telecommand packet transporting a request to distribute physical device commands shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.2.7.1.2.

b. For each telecommand packet transporting a request to distribute physical device commands, the application data field shall have the structure specified in Figure 8.2-6.

repeated N times

N	physical device ID	protocol-specific data	command data
unsigned integer	enumerated	deduced	deduced

Figure 8.2-6 Distribute physical device commands

8.2.2.7 TC[2,8] acquire data from physical devices

a. Each telecommand packet transporting a request to acquire data from physical devices shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.2.7.1.3.

b. For each telecommand packet transporting a request to acquire data from physical devices, the application data field shall have the structure specified in Figure 8.2-7.

repeated N times

N	transaction ID	physical device ID	protocol-specific data
unsigned integer	unsigned integer	enumerated	deduced

Figure 8.2-7 Acquire data from physical devices

8.2.2.8 TM[2,9] physical device data report

a. Each telemetry packet transporting a physical device data report shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.2.7.1.3.

b. For each telemetry packet transporting a physical device data report, the source data field shall have the structure specified in Figure 8.2-8.

repeated N times

N	transaction ID	transaction execution status		data block
		data acquisition return code	auxiliary data	
unsigned integer	unsigned integer	enumerated	deduced	deduced

deduced presence

Figure 8.2-8 Physical device data report

8.2.2.9 TC[2,10] distribute logical device commands

a. Each telecommand packet transporting a request to distribute logical device commands shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.2.7.2.2.

b. For each telecommand packet transporting a request to distribute logical device commands, the application data field shall have the structure specified in Figure 8.2-9.

repeated N times

N	logical device ID	command ID	command arguments
unsigned integer	enumerated	deduced	deduced

Figure 8.2-9 Distribute logical device commands

8.2.2.10 TC[2,11] acquire data from logical devices

a. Each telecommand packet transporting a request to acquire data from logical devices shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.2.7.2.3.

b. For each telecommand packet transporting a request to acquire data from logical devices, the application data field shall have the structure specified in Figure 8.2-10.

repeated N times

N	transaction ID	logical device ID	parameter ID
unsigned integer	unsigned integer	enumerated	enumerated

Figure 8.2-10 Acquire data from logical devices

8.2.2.11 TM[2,12] logical device data report

a. Each telemetry packet transporting a logical device data report shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.2.7.2.3.

b. For each telemetry packet transporting a logical device data report, the source data field shall have the structure specified in Figure 8.2-11.

repeated N times

N	transaction ID	transaction execution status		parameter value
		data acquisition return code	auxiliary data	
unsigned integer	unsigned integer	enumerated	deduced	deduced

deduced presence

Figure 8.2-11 Logical device data report

8.3 ST[03] housekeeping

8.3.1 General

a. Each packet transporting a housekeeping message shall be of service type 3.

8.3.2 Requests and reports

8.3.2.1 TC[3,1] create a housekeeping parameter report structure

a. Each telecommand packet transporting a request to create a housekeeping parameter report structure shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.3.3.6.1.

b. For each telecommand packet transporting a request to create a housekeeping parameter report structure, the application data field shall have the structure specified in Figure 8.3-1.

repeated NFA times

repeated N1 times *repeated N2 times*

housekeeping parameter report structure ID	collection interval	N1	parameter ID	NFA	super commutated sample repetition number	N2	parameter ID
enumerated	unsigned integer	unsigned integer	enumerated	unsigned integer	unsigned integer	unsigned integer	enumerated

Figure 8.3-1 Create a housekeeping parameter report structure

8.3.2.2 TC[3,3] delete housekeeping parameter report structures

a. Each telecommand packet transporting a request to delete housekeeping parameter report structures shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.3.3.6.2.

b. For each telecommand packet transporting a request to delete housekeeping parameter report structures, the application data field shall have the structure specified in Figure 8.3-2.

repeated N times

N	housekeeping parameter report structure ID
unsigned integer	enumerated

Figure 8.3-2 Delete housekeeping parameter report structures

8.3.2.3 TC[3,5] enable the periodic generation of housekeeping parameter reports

a. Each telecommand packet transporting a request to enable the periodic generation of housekeeping parameter reports shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.3.3.5.1.

b. For each telecommand packet transporting a request to enable the periodic generation of housekeeping parameter reports, the application data field shall have the structure specified in Figure 8.3-3.

repeated N times

N	housekeeping parameter report structure ID
unsigned integer	enumerated

Figure 8.3-3 Enable the periodic generation of housekeeping parameter reports

8.3.2.4 TC[3,6] disable the periodic generation of housekeeping parameter reports

a. Each telecommand packet transporting a request to disable the periodic generation of housekeeping parameter reports shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.3.3.5.2.

b. For each telecommand packet transporting a request to disable the periodic generation of housekeeping parameter reports, the application

data field shall have the structure specified in Figure **Error! Use the Home tab to apply to the text that you want to appear here.**-4.

repeated N times

N	housekeeping parameter report structure ID
unsigned integer	enumerated

Figure Error! Use the Home tab to apply to the text that you want to appear here.**-4 Disable the periodic generation of housekeeping parameter reports**

c. To disable the periodic generation of all housekeeping parameter reports, N shall be set to 0.

8.3.2.5 TC[3,9] report housekeeping parameter report structures

a. Each telecommand packet transporting a request to report housekeeping parameter report structures shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.3.3.7.

b. For each telecommand packet transporting a request to report housekeeping parameter report structures, the application data field shall have the structure specified in Figure 8.3-5.

repeated N times

N	housekeeping parameter report structure ID
unsigned integer	enumerated

Figure 8.3-5 Report housekeeping parameter report structures

8.3.2.6 TM[3,10] housekeeping parameter report structure report

a. Each telemetry packet transporting a housekeeping parameter report structure report shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.3.3.7.

b. For each telemetry packet transporting a housekeeping parameter report structure report, the source data field shall have the structure specified in Figure 8.3-6.

repeated NFA times

repeated N1 times *repeated N2 times*

housekeeping parameter report structure ID	periodic generation action status	collection interval	N1	parameter ID	NFA	super commutated sample repetition number	N2	parameter ID
enumerated	enumerated	unsigned integer	unsigned integer	enumerated	unsigned integer	unsigned integer	unsigned integer	enumerated

optional

Figure 8.3-6 Housekeeping parameter report structure report

8.3.2.7 TC[3,27] generate a one shot report for housekeeping parameter report structures

a. Each telecommand packet transporting a request to generate a one shot report for housekeeping parameter report structures shall be of message subtype 27.

NOTE For the corresponding system requirements, refer to clause 6.3.3.8.

b. For each telecommand packet transporting a request to generate a one shot report for housekeeping parameter report structures, the application data field shall have the structure specified in Figure 8.3-7.

repeated N times

N	housekeeping parameter report structure ID
unsigned integer	enumerated

Figure 8.3-7 Generate a one shot report for housekeeping parameter report structures

8.3.2.8 TC[3,29] append parameters to a housekeeping parameter report structure

a. Each telecommand packet transporting a request to append parameters to a housekeeping parameter report structure shall be of message subtype 29.

NOTE For the corresponding system requirements, refer to clause 6.3.3.9.

b. For each telecommand packet transporting a request to append parameters to a housekeeping parameter report structure, the application data field shall have the structure specified in Figure 8.3-8.

repeated NFA times

repeated N1 times *repeated N2 times*

housekeeping parameter report structure ID	N1	parameter ID	NFA	super commutated sample repetition number	N2	parameter ID
enumerated	unsigned integer	enumerated	unsigned integer	unsigned integer	unsigned integer	enumerated

Figure 8.3-8 Append parameters to a housekeeping parameter report structure

8.3.2.9 TC[3,31] modify the collection interval of housekeeping parameter report structures

a. Each telecommand packet transporting a request to modify the collection interval of housekeeping parameter report structures shall be of message subtype 31.

NOTE For the corresponding system requirements, refer to clause 6.3.3.10.

b. For each telecommand packet transporting a request to modify the collection interval of housekeeping parameter report structures, the application data field shall have the structure specified in Figure 8.3-9.

repeated N times

N	housekeeping parameter report structure ID	collection interval
unsigned integer	enumerated	unsigned integer

Figure 8.3-9 Modify the collection interval of housekeeping parameter report structures

8.3.2.10 TC[3,33] report the periodic generation properties of housekeeping parameter report structures

a. Each telecommand packet transporting a request to report the periodic generation properties of housekeeping parameter report structures shall be of message subtype 33.

NOTE For the corresponding system requirements, refer to clause 6.3.3.11.

b. For each telecommand packet transporting a request to report the periodic generation properties of housekeeping parameter report structures, the application data field shall have the structure specified in Figure 8.3-10.

repeated N times

N	housekeeping parameter report structure ID
unsigned integer	enumerated

Figure 8.3-10 Report the periodic generation properties of housekeeping parameter report structures

c. To report the periodic generation properties of all housekeeping parameter report structures, N shall be set to 0.

8.3.2.11 TM[3,34] housekeeping parameter report periodic generation properties report

a. Each telemetry packet transporting a housekeeping parameter report periodic generation properties report shall be of message subtype 34.

NOTE For the corresponding system requirements, refer to clause 6.3.3.11.

b. For each telemetry packet transporting a housekeeping parameter report periodic generation properties report, the source data field shall have the structure specified in Figure 8.3-11.

repeated N times

N	housekeeping parameter report structure ID	periodic generation action status	collection interval
unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.3-11 Housekeeping parameter report periodic generation properties report

8.3.2.12 TC[3,37] apply parameter functional reporting configurations

a. Each telecommand packet transporting a request to apply parameter functional reporting configurations shall be of message subtype 37.

NOTE For the corresponding system requirements, refer to clause 6.3.5.3.

b. For each telecommand packet transporting a request to apply parameter functional reporting configurations, the application data field shall have the structure specified in Figure 8.3-12.

repeated N times

configuration execution flag	N	parameter functional reporting definition ID
enumerated	unsigned integer	enumerated

NOTE For the configuration execution flag enumerated values, see requirement 8.3.3a.

Figure 8.3-12 Apply parameter functional reporting configurations

8.3.2.13 TC[3,38] create a parameter functional reporting definition

a. Each telecommand packet transporting a request to create a parameter functional reporting definition shall be of message subtype 38.

NOTE For the corresponding system requirements, refer to clause 6.3.5.4.1.

b. For each telecommand packet transporting a request to create a parameter functional reporting definition, the application data field shall have the structure specified in Figure 8.3-13.

repeated N1 times

repeated N2 times

parameter functional reporting definition ID	N1	application process ID	N2	parameter report structure ID	periodic generation action status	collection interval
enumerated	unsigned integer	enumerated	unsigned integer	Enumerated	enumerated	unsigned integer

optional

NOTE For the parameter report structure type values, see requirement 1.1.1.1.1a.

Figure 8.3-13 Create a parameter functional reporting definition

8.3.2.14 TC[3,39] delete parameter functional reporting definitions

a. Each telecommand packet transporting a request to delete parameter functional reporting definitions shall be of message subtype 39.

NOTE For the corresponding system requirements, refer to clause 6.3.5.4.2.

b. For each telecommand packet transporting a request to delete parameter functional reporting definitions, the application data field shall have the structure specified in Figure 8.3-14.

repeated N times

N	parameter functional reporting definition ID
unsigned integer	enumerated

Figure 8.3-14 Delete parameter functional reporting definitions

8.3.2.15 TC[3,40] report parameter functional reporting definitions

a. Each telecommand packet transporting a request to report parameter functional reporting definitions shall be of message subtype 40.

NOTE For the corresponding system requirements, refer to clause 6.3.5.5.

b. For each telecommand packet transporting a request to report parameter functional reporting definitions, the application data field shall have the structure specified in Figure 8.3-15.

repeated N times

N	parameter functional reporting definition ID
unsigned integer	enumerated

Figure 8.3-15 Report parameter functional reporting definitions

8.3.2.16 TM[3,41] parameter functional reporting definition report

a. Each telemetry packet transporting a parameter functional reporting definition report shall be of message subtype 41.

NOTE For the corresponding system requirements, refer to clause 6.3.5.5.

b. For each telemetry packet transporting a parameter functional reporting definition report, the source data field shall have the structure specified in Figure 8.3-16.

repeated N1 times

repeated N2 times

parameter functional reporting definition ID	N1	application process ID	N2	parameter report structure ID	periodic generation action status	collection interval
enumerated	unsigned integer	enumerated	unsigned integer	enumerated	enumerated	unsigned integer

optional

NOTE 1 The optional presence of the N1 and the application process ID fields is driven by requirement 6.3.5.2b.
NOTE 2 For the parameter report structure type enumerated values, refer to requirement 1.1.1.1.1a.

Figure 8.3-16 Parameter functional reporting definition report

8.3.2.17 TC[3,42] add parameter report definitions to a parameter functional reporting definition

a. Each telecommand packet transporting a request to add parameter report definitions to a parameter functional reporting definition shall be of message subtype 42.

NOTE For the corresponding system requirements, refer to clause 6.3.5.6.1.

b. For each telecommand packet transporting a request to add parameter report definitions to a parameter functional reporting definition, the application data field shall have the structure specified in Figure 8.3-17.

repeated N1 times

repeated N2 times

parameter functional reporting definition ID	N1	application process ID	N2	parameter report structure ID	periodic generation action status	collection interval
enumerated	unsigned integer	enumerated	unsigned integer	enumerated	enumerated	unsigned integer

optional

NOTE For the parameter report structure type values, see requirement 1.1.1.1.1a.

Figure 8.3-17 Add parameter report definitions to a parameter functional reporting definition

8.3.2.18 TC[3,43] remove parameter report definitions from a parameter functional reporting definition

a. Each telecommand packet transporting a request to remove parameter report definitions from a parameter functional reporting definition shall be of message subtype 43.

NOTE For the corresponding system requirements, refer to clause 6.3.5.6.2.

b. For each telecommand packet transporting a request to remove parameter report definitions from a parameter functional reporting definition, the application data field shall have the structure specified in Figure 8.3-18.

repeated N1 times

repeated N2 times

parameter functional reporting definition ID	N1	application process ID	N2	parameter report structure ID
enumerated	unsigned integer	enumerated	unsigned integer	enumerated

optional

NOTE For the parameter report structure type values, see requirement 1.1.1.1.1a.

Figure 8.3-18 Remove parameter report definitions from a parameter functional reporting definition

8.3.2.19 TC[3,44] modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition

a. Each telecommand packet transporting a request to modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition shall be of message subtype 44.

NOTE For the corresponding system requirements, refer to clause 6.3.5.6.3.

b. For each telecommand packet transporting a request to modify the periodic generation properties of parameter report definitions of a

parameter functional reporting definition, the application data field shall have the structure specified in Figure 8.3-21 .

repeated N1 times

repeated N2 times

parameter functional reporting definition ID	N1	application process ID	N2	parameter report structure ID	periodic generation action status	collection interval
enumerated	unsigned integer	enumerated	unsigned integer	enumerated	enumerated	unsigned integer

optional

NOTE For the parameter report structure type values, see requirement 1.1.1.1.1a.

Figure 8.3-21 Modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition

8.3.2.20 TM[3,25] category 1 housekeeping parameter report

a. Each telemetry packet transporting a category 1 housekeeping parameter report shall be of message subtype 25.

NOTE For the corresponding system requirements, refer to clause 6.3.3.4.

b. For each telemetry packet transporting a category 1 housekeeping parameter report, the source data field shall have the structure specified in .

deduced repeated number of times

housekeeping parameter report structure ID	parameter value
unsigned integer	deduced

Figure 8.3-19 Category 1 housekeeping parameter report

8.3.2.21 TM[3,26] category 2 housekeeping parameter report

a. Each telemetry packet transporting a category 2 housekeeping parameter report shall be of message subtype 26.

NOTE For the corresponding system requirements, refer to clause 6.3.3.4.

b. For each telemetry packet transporting a category 2 housekeeping parameter report, the source data field shall have the structure specified in .

deduced repeated number of times

housekeeping parameter report structure ID	parameter value
unsigned integer	deduced

Figure 8.3-20 Category 2 housekeeping parameter report

8.3.2.22 TM[3,50] category 3 housekeeping parameter report

a. Each telemetry packet transporting a category 3 housekeeping parameter report shall be of message subtype 50.

NOTE For the corresponding system requirements, refer to clause 6.3.3.4.

b. For each telemetry packet transporting a category 3 housekeeping parameter report, the source data field shall have the structure specified in Figure 8.3-21.

deduced repeated number of times

housekeeping parameter report structure ID	parameter value
unsigned integer	deduced

Figure 8.3-21 Category 3 housekeeping parameter report

8.3.2.23 TM[3,51] category 4 housekeeping parameter report

a. Each telemetry packet transporting a category 4 housekeeping parameter report shall be of message subtype 51.

NOTE For the corresponding system requirements, refer to clause 6.3.3.4.

b. For each telemetry packet transporting a category 4 housekeeping parameter report, the source data field shall have the structure specified in Figure 8.3-3.

deduced repeated number of times

housekeeping parameter report structure ID	parameter value
unsigned integer	deduced

Figure 8.3-23 Category 4 housekeeping parameter report

8.3.2.24 TM[3,52] category 5 housekeeping parameter report

a. Each telemetry packet transporting a category 5 housekeeping parameter report shall be of message subtype 52.

NOTE For the corresponding system requirements, refer to clause 6.3.3.4.

b. For each telemetry packet transporting a category 5 housekeeping parameter report, the source data field shall have the structure specified in Figure 8.3-22.

deduced repeated number of times

housekeeping parameter report structure ID	parameter value
unsigned integer	deduced

Figure 8.3-22 Category 5 housekeeping parameter report

8.3.2.25 TM[3,53] category 6 housekeeping parameter report

a. Each telemetry packet transporting a category 6 housekeeping parameter report shall be of message subtype 53.

NOTE For the corresponding system requirements, refer to clause 6.3.3.4.

b. For each telemetry packet transporting a category 6 housekeeping parameter report, the source data field shall have the structure specified in Figure 8.3-23.

deduced repeated number of times

housekeeping parameter report structure ID	parameter value
unsigned integer	deduced

Figure 8.3-23 Category 6 housekeeping parameter report

8.3.3 Enumeration

a. The values of the configuration execution flag shall be as specified in Table 8.3-1.

Table 8.3-1 Service 3 configuration execution flag

engineering value	raw value
"non-exclusive"	0
"exclusive"	1

8.4 ST[04] parameter statistics reporting

8.4.1 General

a. Each packet transporting a parameter statistics reporting message shall be of service type 4.

8.4.2 Requests and reports

8.4.2.1 TC[4,1] report the parameter statistics

a. Each telecommand packet transporting a request to report the parameter statistics shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.4.5.2.

b. For each telecommand packet transporting a request to report the parameter statistics, the application data field shall have the structure specified in Figure 8.4-1.

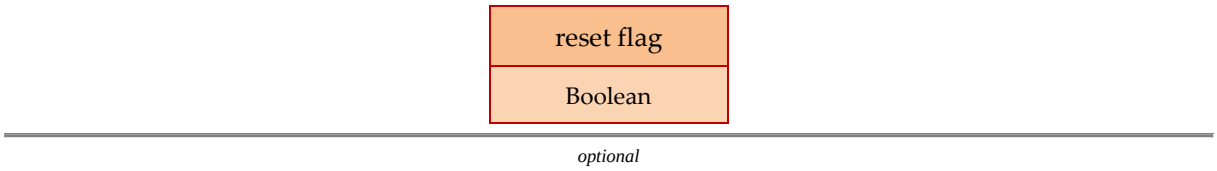

Figure 8.4-1 Report the parameter statistics

8.4.2.2 TM[4,2] parameter statistics report

a. Each telemetry packet transporting a parameter statistics report shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.4.5.3.

b. For each telemetry packet transporting a parameter statistics report, the source data field shall have the structure specified in Figure 8.4-2.

repeated N times

start time	end time	N	parameter ID	number of samples	maximum		minimum		mean value	standard deviation value
					value	time	value	time		
absolute time	absolute time	unsigned integer	enumerated	unsigned integer	deduced	absolute time	deduced	absolute time	deduced	deduced

optional

NOTE The formats of the max value field, the min value field, the mean value field and the standard deviation value field are specific to the parameter identified by the associated parameter ID field.

Figure 8.4-2 Parameter statistics report

8.4.2.3 TC[4,3] reset the parameter statistics

a. Each telecommand packet transporting a request to reset the parameter statistics shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.4.4.

b. For each telecommand packet transporting a request to reset the parameter statistics, the application data field shall be omitted.

8.4.2.4 TC[4,4] enable the periodic parameter statistics reporting

a. Each telecommand packet transporting a request to enable the periodic parameter statistics reporting shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.4.6.2.

b. For each telecommand packet transporting a request to enable the periodic parameter statistics reporting, the application data field shall have the structure specified in Figure 8.4-3.

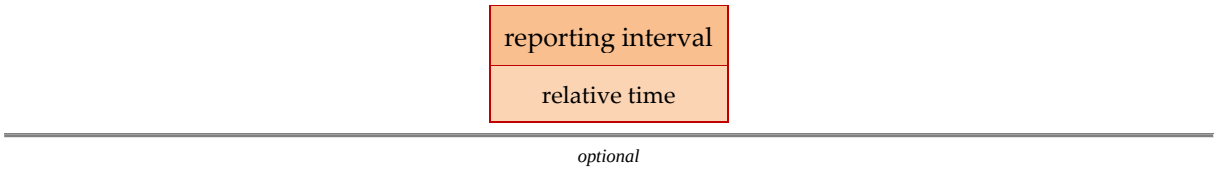

Figure 8.4-3 Enable the periodic parameter statistics reporting

8.4.2.5 TC[4,5] disable the periodic parameter statistics reporting

a. Each telecommand packet transporting a request to disable the periodic parameter statistics reporting shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.4.6.3.

b. For each telecommand packet transporting a request to disable the periodic parameter statistics reporting, the application data field shall be omitted.

8.4.2.6 TC[4,6] add or update parameter statistics definitions

a. Each telecommand packet transporting a request to add or update parameter statistics definitions shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.4.7.1.

b. For each telecommand packet transporting a request to add or update parameter statistics definitions, the application data field shall have the structure specified in Figure 8.4-4.

repeated N times

N	parameter ID	sampling interval
unsigned integer	enumerated	relative time

optional

Figure 8.4-4 Add or update parameter statistics definitions

8.4.2.7 TC[4,7] delete parameter statistics definitions

a. Each telecommand packet transporting a request to delete parameter statistics definitions shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.4.7.2.

b. For each telecommand packet transporting a request to delete parameter statistics definitions, the application data field shall have the structure specified in Figure 8.4-5.

repeated N times

N	parameter ID
unsigned integer	enumerated

Figure 8.4-5 Delete parameter statistics definitions

c. To delete all parameter statistics definitions, N shall be set to 0.

8.4.2.8 TC[4,8] report the parameter statistics definitions

a. Each telecommand packet transporting a request to report the parameter statistics definitions shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.4.7.3.

b. For each telecommand packet transporting a request to report the parameter statistics definitions, the application data field shall be omitted.

8.4.2.9 TM[4,9] parameter statistics definition report

a. Each telemetry packet transporting a parameter statistics definition report shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.4.7.3.

b. For each telemetry packet transporting a parameter statistics definition report, the source data field shall have the structure specified in Figure 8.4-6.

repeated N times

reporting interval	N	parameter ID	sampling interval
relative time	unsigned integer	enumerated	relative time

optional *optional*

Figure 8.4-6 Parameter statistics definition report

c. Whenever a parameter statistics definition report is generated, if the reporting interval field is present and the periodic reporting is not enabled, the reporting interval field value shall be set to zero seconds.

8.5 ST[05] event reporting

8.5.1 General

a. Each packet transporting an event reporting message shall be of service type 5.

8.5.2 Requests and reports

8.5.2.1 TM[5,1] informative event report

a. Each telemetry packet transporting an informative event report shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.5.4.

b. For each telemetry packet transporting an informative event report, the source data field shall have the structure specified in Figure 8.5-1.

event definition ID	auxiliary data
enumerated	deduced

deduced presence

NOTE The event definition ID, together with the application process ID, identifies an event definition and as such the presence and structure of the auxiliary data field.

Figure 8.5-1 Informative event report

8.5.2.2 TM[5,2] low severity anomaly report

a. Each telemetry packet transporting a low severity anomaly report shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.5.4.

b. For each telemetry packet transporting a low severity anomaly report, the source data field shall have the structure specified in Figure 8.5-2.

event definition ID	auxiliary data
enumerated	deduced

deduced presence

NOTE The event definition ID, together with the application process ID, identifies an event definition and as such the presence and structure of the auxiliary data field.

Figure 8.5-2 Low severity anomaly report

8.5.2.3 TM[5,3] medium severity anomaly report

a. Each telemetry packet transporting a medium severity anomaly report shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.5.4.

b. For each telemetry packet transporting a medium severity anomaly report, the source data field shall have the structure specified in Figure 8.5-3.

event definition ID	auxiliary data
enumerated	deduced

deduced presence

NOTE The event definition ID, together with the application process ID, identifies an event definition and as such the presence and structure of the auxiliary data field.

Figure 8.5-3 Medium severity anomaly report

8.5.2.4 TM[5,4] High severity anomaly report

a. Each telemetry packet transporting a high severity anomaly report shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.5.4.

b. For each telemetry packet transporting a high severity anomaly report, the source data field shall have the structure specified in Figure 8.5-4.

event definition ID	auxiliary data
enumerated	deduced

deduced presence

NOTE The event definition ID, together with the application process ID, identifies an event definition and as such the presence and structure of the auxiliary data field.

Figure 8.5-4 High severity anomaly report

8.5.2.5 TC[5,5] enable the report generation of event definitions

a. Each telecommand packet transporting a request to enable the report generation of event definitions shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause b.

b. For each telecommand packet transporting a request to enable the report generation of event definitions, the application data field shall have the structure specified in Figure 8.5-5.

repeated N times

N	event definition ID
unsigned integer	enumerated

Figure 8.5-5 Enable the report generation of event definitions

8.5.2.6 TC[5,6] disable the report generation of event definitions

a. Each telecommand packet transporting a request to disable the report generation of event definitions shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.5.5.3.

b. For each telecommand packet transporting a request to disable the report generation of event definitions, the application data field shall have the structure specified in Figure 8.5-6.

repeated N times

N	event definition ID
unsigned integer	enumerated

Figure 8.5-6 Disable the report generation of event definitions

8.5.2.7 TC[5,7] report the list of disabled event definitions

a. Each telecommand packet transporting a request to report the list of disabled event definitions shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.5.5.4.

b. For each telecommand packet transporting a request to report the list of disabled event definitions, the application data field shall be omitted.

8.5.2.8 TM[5,8] disabled event definitions list report

a. Each telemetry packet transporting a disabled event definitions list report shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.5.5.4.

b. For each telemetry packet transporting a disabled event definitions list report, the source data field shall have the structure specified in Figure 8.5-7.

repeated N times

N	event definition ID
unsigned integer	enumerated

Figure 8.5-7 Disabled event definitions list report

c. To report that the list of disabled event definitions is empty, N shall be set to 0.

8.5.2.9 TC[5,9] generate event report

a. Each telecommand packet transporting a request to generate an event report shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.5.5.4.

b. For each telecommand packet transporting a request to generate an event report, the application data field shall have the structure specified in Figure 8.5-5.

event APID	event severity	event definition ID	event data	
			length	data
enumerated	enumerated	enumerated	variable octet-string	

Figure 8.5-8 Generate event report

8.6 ST[06] memory management

8.6.1 General

a. Each packet transporting a memory management message shall be of service type 6.

b. Whether the memory management service supports multiple instructions within memory management related requests shall be declared when specifying that service.

8.6.2 Requests and reports

8.6.2.1 TC[6,1] load object memory data

a. Each telecommand packet transporting a request to load object memory data shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.6.4.4.

b. For each telecommand packet transporting a request to load object memory data, the application data field shall have the structure specified in Figure 8.6-1.

repeated N times

memory ID	base	N	offset	data to load	checksum
enumerated	deduced	unsigned integer	unsigned integer	variable-length octet-string	bit-string (16 bits)

optional *optional*

NOTE The PFC of the length field of the data to load is driven by requirement 7.3.8d.

Figure 8.6-1 Load object memory data

8.6.2.2 TC[6,2] load raw memory data areas

a. Each telecommand packet transporting a request to load raw memory data areas shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.6.3.3.1.

b. For each telecommand packet transporting a request to load raw memory data areas, the application data field shall have the structure specified in Figure 8.6-2.

repeated N times

memory ID	N	start address	data to load	checksum
enumerated	unsigned integer	unsigned integer	variable-length octet-string	bit-string (16 bits)

optional *optional*

NOTE The PFC of the length field of the data to load is driven by requirement 7.3.8d.

Figure 8.6-2 Load raw memory data areas

8.6.2.3 TC[6,3] dump object memory data

a. Each telecommand packet transporting a request to dump object memory data shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.6.4.5.

b. For each telecommand packet transporting a request to dump object memory data, the application data field shall have the structure specified in Figure 8.6-3.

repeated N times

memory ID	base	N	offset	length
enumerated	deduced	unsigned integer	unsigned integer	unsigned integer

optional

Figure 8.6-3 Dump object memory data

8.6.2.4 TM[6,4] dumped object memory data report

a. Each telemetry packet transporting a dumped object memory data report shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.6.4.5.

b. For each telemetry packet transporting a dumped object memory data report, the source data field shall have the structure specified in Figure 8.6-4.

repeated N times

memory ID	base	N	offset	dumped data	checksum
enumerated	deduced	unsigned integer	unsigned integer	variable-length octet-string	bit-string (16 bits)

optional *optional*

NOTE The PFC of the length field of the dumped data is driven by requirement 7.3.8d.

Figure 8.6-4 Dumped object memory data report

8.6.2.5 TC[6,5] dump raw memory data

a. Each telecommand packet transporting a request to dump raw memory data shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.6.3.4.

b. For each telecommand packet transporting a request to dump raw memory data, the application data field shall have the structure specified in Figure 8.6-5.

repeated N times

memory ID	N	start address	length
enumerated	unsigned integer	unsigned integer	unsigned integer

optional

Figure 8.6-5 Dump raw memory data

8.6.2.6 TM[6,6] dumped raw memory data report

a. Each telemetry packet transporting a dumped raw memory data report shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.6.3.4.

b. For each telemetry packet transporting a dumped raw memory data report, the source data field shall have the structure specified in Figure 8.6-6.

repeated N times

memory ID	N	start address	dumped data	checksum
enumerated	unsigned integer	unsigned integer	variable-length octet-string	bit-string (16 bits)

optional *optional*

NOTE The PFC of the length field of the dumped data is driven by requirement 7.3.8d.

Figure 8.6-6 Dumped raw memory data report

8.6.2.7 TC[6,7] check object memory data

a. Each telecommand packet transporting a request to check object memory data shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.6.4.6.

b. For each telecommand packet transporting a request to check object memory data, the application data field shall have the structure specified in Figure 8.6-7.

repeated N times

memory ID	base	N	offset	length
enumerated	deduced	unsigned integer	unsigned integer	unsigned integer

optional

Figure 8.6-7 Check object memory data

8.6.2.8 TM[6,8] checked object memory data report

a. Each telemetry packet transporting a checked object memory data report shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.6.4.6.

b. For each telemetry packet transporting a checked object memory data report, the source data field shall have the structure specified in Figure 8.6-8.

repeated N times

memory ID	base	N	offset	length	checksum
enumerated	deduced	unsigned integer	unsigned integer	unsigned integer	bit-string (16 bits)

optional

Figure 8.6-8 Checked object memory data report

8.6.2.9 TC[6,9] check raw memory data

a. Each telecommand packet transporting a request to check raw memory data shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.6.3.5.

b. For each telecommand packet transporting a request to check raw memory data, the application data field shall have the structure specified in Figure 8.6-9.

repeated N times

memory ID	N	start address	length
enumerated	unsigned integer	unsigned integer	unsigned integer

optional

Figure 8.6-9 Check raw memory data

8.6.2.10 TM[6,10] checked raw memory data report

a. Each telemetry packet transporting a checked raw memory data report shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.6.3.5.

b. For each telemetry packet transporting a checked raw memory data report, the source data field shall have the structure specified in Figure 8.6-10.

repeated N times

memory ID	N	start address	length	checksum
enumerated	unsigned integer	unsigned integer	unsigned integer	bit-string (16 bits)

optional

Figure 8.6-10 Checked raw memory data report

8.6.2.11 TC[6,11] load a raw memory atomic data area in a non-interruptible transaction

a. Each telecommand packet transporting a request to load a raw memory atomic data area in a non-interruptible transaction shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.6.3.3.2.

b. For each telecommand packet transporting a request to load a raw memory atomic data area in a non-interruptible transaction, the application data field shall have the structure specified in Figure 8.6-11.

memory ID	start address	bit mask	data to load
enumerated	unsigned integer	fixed octet-string (deduced size)	fixed octet-string (deduced size)

optional

NOTE The deduced size of the bit mask field and of the data to load field is driven by requirement 5.4.3.4.1c.1. The size of each of these fields is equal to the size of the memory access alignment constraint defined by the memory ID.

Figure 8.6-11 Load a raw memory atomic data area in a non-interruptible transaction

8.6.2.12 TC[6,12] abort all memory dumps

a. Each telecommand packet transporting a request to abort all memory dumps shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.6.5.1.

b. For each telecommand packet transporting a request to abort all memory dumps, the application data field shall be omitted.

8.6.2.13 TC[6,13] enable the scrubbing of a memory

a. Each telecommand packet transporting a request to enable the scrubbing of a memory shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.6.6.1.4.

b. For each telecommand packet transporting a request to enable the scrubbing of a memory, the application data field shall have the structure specified in Figure 8.6-12.

memory ID
enumerated

optional

Figure 8.6-12 Enable the scrubbing of a memory

8.6.2.14 TC[6,14] disable the scrubbing of a memory

a. Each telecommand packet transporting a request to disable the scrubbing of a memory shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.6.6.1.5.

b. For each telecommand packet transporting a request to disable the scrubbing of a memory, the application data field shall have the structure specified in Figure 8.6-13.

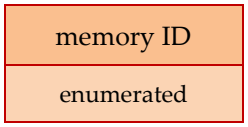

Figure 8.6-13 Disable the scrubbing of a memory

8.6.2.15 TC[6,15] enable the write protection of a memory

a. Each telecommand packet transporting a request to enable the write protection of a memory shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.6.6.2.4.

b. For each telecommand packet transporting a request to enable the write protection of a memory, the application data field shall have the structure specified in Figure 8.6-14.

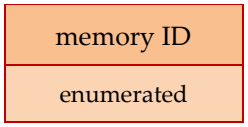

Figure 8.6-14 Enable the write protection of a memory

8.6.2.16 TC[6,16] disable the write protection of a memory

a. Each telecommand packet transporting a request to disable the write protection of a memory shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.6.6.2.5.

b. For each telecommand packet transporting a request to disable the write protection of a memory, the application data field shall have the structure specified in Figure 8.6-15.

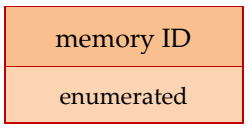

Figure 8.6-15 Disable the write protection of a memory

8.6.2.17 TC[6,17] check an object memory object

a. Each telecommand packet transporting a request to check an object memory object shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.6.4.7.

b. For each telecommand packet transporting a request to check an object memory object, the application data field shall have the structure specified in Figure 8.6-16.

memory ID	base
enumerated	deduced

optional

Figure 8.6-16 Check an object memory object

8.6.2.18 TM[6,18] checked object memory object report

a. Each telemetry packet transporting a checked object memory object report shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.6.4.7.

b. For each telemetry packet transporting a checked object memory object report, the source data field shall have the structure specified in Figure 8.6-17.

memory ID	base	length	checksum
enumerated	deduced	unsigned integer	bit-string (16 bits)

optional

Figure 8.6-17 Checked object memory object report

8.6.2.19 TC[6,19] load raw memory data areas by reference

a. Each telecommand packet transporting a request to load raw memory data areas by reference shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.6.3.6.

b. For each telecommand packet transporting a request to load raw memory data areas, the application data field shall have the structure specified in Figure 8.6-18.

repeated N times

memory ID	file path		N	start address	offset in file	length	checksum
	repository path	file name					
enumerated	variable-length character-string	variable-length character-string	unsigned integer	unsigned integer	unsigned integer	unsigned integer	bit-string (16 bits)

optional *optional*

Figure 8.6-18 Load raw memory data areas by reference

8.6.2.20 TC[6,20] dump raw memory data areas to file

a. Each telecommand packet transporting a request to dump raw memory data areas to file shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.6.3.7.

b. For each telecommand packet transporting a request to dump raw memory data areas to file, the application data field shall have the structure specified in Figure 8.6-19.

repeated N times

memory ID	file path		N	start address	length
	repository path	file name			
enumerated	variable-length character-string	variable-length character-string	unsigned integer	unsigned integer	unsigned integer

optional

Figure 8.6-19 Dump raw memory data areas to file

8.6.2.21 TC[6,21] load object memory data areas by reference

a. Each telecommand packet transporting a request to load object memory data areas by reference shall be of message subtype 21.

NOTE For the corresponding system requirements, refer to clause 6.6.4.8.

b. For each telecommand packet transporting a request to load object memory data areas, the application data field shall have the structure specified in Figure 8.6-20.

repeated N times

memory ID	base	file path		N	destination offset	offset in file	length	checksum
		repository path	file name					
enumerated	deduced	variable-length character-string	variable-length character-string	unsigned integer	unsigned integer	unsigned integer	unsigned integer	bit-string (16 bits)

optional *optional*

Figure 8.6-20 Load object memory data areas by reference

8.6.2.22 TC[6,22] dump object memory data areas to file

a. Each telecommand packet transporting a request to dump object memory data areas to file shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.6.4.9.

b. For each telecommand packet transporting a request to dump object memory data areas to file, the application data field shall have the structure specified in Figure 8.6-21.

repeated N times

memory ID	base	file path		N	offset	length
		repository path	file name			
enumerated	deduced	variable-length character-string	variable-length character-string	unsigned integer	unsigned integer	unsigned integer

optional

Figure 8.6-21 Dump object memory data areas to file

8.7 ST[07] (reserved)

8.8 ST[08] (reserved)

8.9 ST[09] time management

8.9.1 General

a. Each packet transporting a time management message shall be of service type 9.

NOTE The time reports generated by the time reporting subservice are spacecraft time packets. A spacecraft time packet does not carry the message type, consisting of the service type and message subtype. Nevertheless, the message type is associated to the time report and can be used in PUS services: for example, by the real-time forwarding control subservice specified in clause 6.14.2.

b. The spacecraft time packets shall not have any packet secondary header field.

NOTE The spacecraft time packets are specified clauses 6.9.4.2 and 6.9.4.3. See also requirement 7.4.3.1a.

c. For each spacecraft time packet, the secondary header flag in its packet primary header shall be set to 0.

NOTE Setting the secondary header flag to 0 indicates that the packet secondary header field is not present in the packet.

d. For each spacecraft time packet, the application process identifier in its packet primary header shall be set to zero.

NOTE For the application process identifier, the value zero is reserved for use in spacecraft time packets.

8.9.2 Requests and reports

8.9.2.1 TC[9,1] set the time report generation rate

a. Each telecommand packet transporting a request to set the time report generation rate shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.9.5.1.1.

b. For each telecommand packet transporting a request to set the time report generation rate, the application data field shall have the structure specified in Figure 8.9-1.

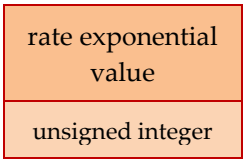

Figure 8.9-1 Set the time report generation rate

8.9.2.2 TM[9,2] CUC time report

a. Each telemetry packet transporting a CUC time report shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.9.4.2.

b. For each telemetry packet transporting a CUC time report, the source data field shall have the structure specified in Figure 8.9-2.

rate exponential value	spacecraft time	spacecraft time reference status
unsigned integer	absolute time	deduced
optional		*optional*

NOTE The spacecraft time field is formatted according to the CUC time code format, refer to requirement 6.9.4.2d.

Figure 8.9-2 CUC time report

8.9.2.3 TM[9,3] CDS time report

a. Each telemetry packet transporting a CDS time report shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.9.4.3.

b. For each telemetry packet transporting a CDS time report, the source data field shall have the structure specified in Figure 8.9-3.

rate exponential value	spacecraft time	spacecraft time reference status
unsigned integer	absolute time	deduced
optional		*optional*

NOTE The spacecraft time field is formatted according to the CDS time code format, refer to requirement 6.9.4.3d.

Figure 8.9-3 CDS time report

8.9.2.4 TC[9,4] set the reference time with absolute time

a. Each telecommand packet transporting a request to set the reference time with absolute time shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.9.6.1.1.

b. For each telecommand packet transporting a request to set the reference time with absolute time, the application data field shall have the structure specified in Figure 8.9-4.

reference time	spacecraft time reference status
absolute time	deduced

optional

Figure 8.9-4 Set reference time with absolute time

8.9.2.5 TC[9,5] set the reference time with relative time

a. Each telecommand packet transporting a request to set the reference time with relative time shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.9.6.1.2.

b. For each telecommand packet transporting a request to set the reference time with relative time, the application data field shall have the structure specified in Figure 8.9-5.

reference time	spacecraft time reference status
relative time	deduced

optional

Figure 8.9-5 Set reference time with relative time

8.9.2.6 TC[9,6] start time distribution to on-board users

a. Each telecommand packet transporting a request to start time distribution to on-board users shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.9.7.2.1.

b. For each telecommand packet transporting a request to start time distribution to on-board users, the application data field shall have the structure specified in Figure 8.9-6.

repeated N times

N	distribution interval	application process ID
unsigned integer	unsigned integer	enumerated

Figure 8.9-6 Start time distribution to on-board users

c. For a one-shot time distribution, the distribution interval shall be set to 0.

8.9.2.7 TC[9,7] stop time distribution to on-board users

a. Each telecommand packet transporting a request to stop time distribution to on-board users shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.9.7.2.2.

b. For each telecommand packet transporting a request to stop time distribution to on-board users, the application data field shall have the structure specified in Figure 8.9-7.

repeated N times

N	application process ID
unsigned integer	enumerated

Figure 8.9-7 Stop time distribution to on-board users

8.10 ST[10] (reserved)

8.11 ST[11] time-based scheduling

8.11.1 General

a. Each packet transporting a time-based scheduling message shall be of service type 11.

8.11.2 Requests and reports

8.11.2.1 TC[11,1] enable the time-based schedule execution function

a. Each telecommand packet transporting a request to enable the time-based schedule execution function shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.11.4.3.2.

b. For each telecommand packet transporting a request to enable the time-based schedule execution function, the application data field shall be omitted.

8.11.2.2 TC[11,2] disable the time-based schedule execution function

a. Each telecommand packet transporting a request to disable the time-based schedule execution function shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.11.4.3.3.

b. For each telecommand packet transporting a request to disable the time-based schedule execution function, the application data field shall be omitted.

8.11.2.3 TC[11,3] reset the time-based schedule

a. Each telecommand packet transporting a request to reset the time-based schedule shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.11.4.4.

b. For each telecommand packet transporting a request to reset the time-based schedule, the application data field shall be omitted.

8.11.2.4 TC[11,4] insert activities into the time-based schedule

a. Each telecommand packet transporting a request to insert activities into the time-based schedule shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.11.4.5.

b. For each telecommand packet transporting a request to insert activities into the time-based schedule, the application data field shall have the structure specified in Figure 8.11-1.

repeated N times

sub-schedule ID	N	group ID	release time	request
enumerated	unsigned integer	enumerated	absolute time	TC packet

optional *optional*

Figure 8.11-1 Insert activities into the time-based schedule

8.11.2.5 TC[11,5] delete time-based scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to delete time-based scheduled activities identified by request identifier shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.11.9.2.

b. For each telecommand packet transporting a request to delete time-based scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.11-2.

repeated N times

N	request ID		
	source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.11-2 Delete time-based scheduled activities identified by request identifier

8.11.2.6 TC[11,6] delete the time-based scheduled activities identified by a filter

a. Each telecommand packet transporting a request to delete the time-based scheduled activities identified by a filter shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.11.10.3.

b. For each telecommand packet transporting a request to delete the time-based scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.11-3.

repeated N1 times *repeated N2 times*

time window			N1	sub-schedule ID	N2	group ID
type	time tag 1	time tag 2				
enumerated	absolute time	absolute time	unsigned integer	enumerated	unsigned integer	enumerated
	deduced presence	*deduced presence*	*optional*		*optional*	

NOTE For the type enumerated values, refer to requirement 8.11.3c.

Figure 8.11-3 Delete the time-based scheduled activities identified by a filter

8.11.2.7 TC[11,7] time-shift scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to time-shift scheduled activities identified by request identifier shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.11.9.3.

b. For each telecommand packet transporting a request to time-shift scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.11-4.

repeated N times

time offset	N	request ID		
		source ID	application process ID	sequence count
relative time	unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.11-4 Time-shift scheduled activities identified by request identifier

8.11.2.8 TC[11,8] time-shift the scheduled activities identified by a filter

a. Each telecommand packet transporting a request to time-shift the scheduled activities identified by a filter shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.11.10.4.

b. For each telecommand packet transporting a request to time-shift the scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.11-5.

				repeated N1 times		*repeated N2 times*	
time offset	time window			N1	sub-schedule ID	N2	group ID
	type	time tag 1	time tag 2				
relative time	enumerated	absolute time	absolute time	unsigned integer	enumerated	unsigned integer	enumerated
		deduced presence	*deduced presence*	*optional*		*optional*	

NOTE For the type enumerated values, refer to requirement 8.11.3c.

Figure 8.11-5 Time-shift the scheduled activities identified by a filter

8.11.2.9 TC[11,9] detail-report time-based scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to detail-report time-based scheduled activities identified by request identifier shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.11.9.5.

b. For each telecommand packet transporting a request to detail-report time-based scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.11-6.

	repeated N times		
N	request ID		
	source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.11-6 Detail-report time-based scheduled activities identified by request identifier

8.11.2.10 TM[11,10] time-based schedule detail report

a. Each telemetry packet transporting a time-based schedule detail report shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.11.7.2.

b. For each telemetry packet transporting a time-based schedule detail report, the source data field shall have the structure specified in Figure 8.11-7.

repeated N times

N	sub-schedule ID	group ID	release time	request
unsigned integer	enumerated	enumerated	absolute time	TC packet

optional *optional*

Figure 8.11-7 Time-based schedule detail report

8.11.2.11 TC[11,11] detail-report the time-based scheduled activities identified by a filter

a. Each telecommand packet transporting a request to detail-report the time-based scheduled activities identified by a filter shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.11.10.6.

b. For each telecommand packet transporting a request to detail-report the time-based scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.11-8.

repeated N1 times *repeated N2 times*

time window			N1	sub-schedule ID	N2	group ID
type	time tag 1	time tag 2				
enumerated	absolute time	absolute time	unsigned integer	enumerated	unsigned integer	enumerated

deduced presence *deduced presence* *optional* *optional*

NOTE For the type enumerated values, refer to requirement 8.11.3c.

Figure 8.11-8 Detail-report the time-based scheduled activities identified by a filter

8.11.2.12 TC[11,12] summary-report time-based scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to summary-report time-based scheduled activities identified by request identifier shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.11.9.4.

b. For each telecommand packet transporting a request to summary-report time-based scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.11-9.

repeated N times

N	request ID		
	source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.11-9 Summary-report time-based scheduled activities identified by request identifier

8.11.2.13 TM[11,13] time-based schedule summary report

a. Each telemetry packet transporting a time-based schedule summary report shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.11.7.1.

b. For each telemetry packet transporting a time-based schedule summary report, the source data field shall have the structure specified in Figure 8.11-10.

repeated N times

N	sub-schedule ID	group ID	release time	request ID		
				source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	absolute time	enumerated	enumerated	unsigned integer

optional *optional*

Figure 8.11-10 Time-based schedule summary report

8.11.2.14 TC[11,14] summary-report the time-based scheduled activities identified by a filter

a. Each telecommand packet transporting a request to summary-report the time-based scheduled activities identified by a filter shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.11.10.5.

b. For each telecommand packet transporting a request to summary-report the time-based scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.11-11.

				repeated N1 times		repeated N2 times
time window			N1	sub-schedule ID	N2	group ID
type	time tag 1	time tag 2				
enumerated	absolute time	absolute time	unsigned integer	enumerated	unsigned integer	enumerated
	deduced presence	deduced presence		optional		optional

NOTE For the type enumerated values, refer to requirement 8.11.3c.

Figure 8.11-11 Summary-report the time-based scheduled activities identified by a filter

8.11.2.15 TC[11,15] time-shift all scheduled activities

a. Each telecommand packet transporting a request to time-shift all scheduled activities shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.11.8.1.

b. For each telecommand packet transporting a request to time-shift all scheduled activities, the application data field shall have the structure specified in Figure 8.11-12.

time offset
relative time

Figure 8.11-12 Time-shift all scheduled activities

8.11.2.16 TC[11,16] detail-report all time-based scheduled activities

a. Each telecommand packet transporting a request to detail-report all time-based scheduled activities shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.11.8.3.

b. For each telecommand packet transporting a request to detail-report all time-based scheduled activities, the application data field shall be omitted.

8.11.2.17 TC[11,17] summary-report all time-based scheduled activities

a. Each telecommand packet transporting a request to summary-report all time-based scheduled activities shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.11.8.2.

b. For each telecommand packet transporting a request to summary-report all time-based scheduled activities, the application data field shall be omitted.

8.11.2.18 TC[11,18] report the status of each time-based sub-schedule

a. Each telecommand packet transporting a request to report the status of each time-based sub-schedule shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.11.5.2.3.

b. For each telecommand packet transporting a request to report the status of each time-based sub-schedule, the application data field shall be omitted.

8.11.2.19 TM[11,19] time-based sub-schedule status report

a. Each telemetry packet transporting a time-based sub-schedule status report shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.11.5.2.3.

b. For each telemetry packet transporting a time-based sub-schedule status report, the source data field shall have the structure specified in Figure 8.11-13.

repeated N times

N	sub-schedule ID	sub-schedule status
unsigned integer	enumerated	enumerated
NOTE For the sub-schedule status enumerated values, refer to requirement 8.11.3a.		

Figure 8.11-13 Time-based sub-schedule status report

8.11.2.20 TC[11,20] enable time-based sub-schedules

a. Each telecommand packet transporting a request to enable time-based sub-schedules shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.11.5.2.1.

b. For each telecommand packet transporting a request to enable time-based sub-schedules, the application data field shall have the structure specified in Figure 8.11-14.

repeated N times

N	sub-schedule ID
unsigned integer	enumerated

Figure 8.11-14 Enable time-based sub-schedules

c. To enable all time-based sub-schedules, N shall be set to 0.

8.11.2.21 TC[11,21] disable time-based sub-schedules

a. Each telecommand packet transporting a request to disable time-based sub-schedules shall be of message subtype 21.

NOTE For the corresponding system requirements, refer to clause 6.11.5.2.2.

b. For each telecommand packet transporting a request to disable time-based sub-schedules, the application data field shall have the structure specified in Figure 8.11-15.

repeated N times

N	sub-schedule ID
unsigned integer	enumerated

Figure 8.11-15 Disable time-based sub-schedules

c. To disable all time-based sub-schedules, N shall be set to 0.

8.11.2.22 TC[11,22] create time-based scheduling groups

a. Each telecommand packet transporting a request to create time-based scheduling groups shall be of message subtype 22.

NOTE For the corresponding system requirements, refer to clause 6.11.6.2.1.

b. For each telecommand packet transporting a request to create time-based scheduling groups, the application data field shall have the structure specified in Figure 8.11-16.

repeated N times

N	group ID	group status
unsigned integer	enumerated	enumerated
NOTE For the group status enumerated values, refer to requirement 8.11.3b.		

Figure 8.11-16 Create time-based scheduling groups

8.11.2.23 TC[11,23] delete time-based scheduling groups

a. Each telecommand packet transporting a request to delete time-based scheduling groups shall be of message subtype 23.

NOTE For the corresponding system requirements, refer to clause 6.11.6.2.2.

b. For each telecommand packet transporting a request to delete time-based scheduling groups, the application data field shall have the structure specified in Figure 8.11-17.

repeated N times

N	group ID
unsigned integer	enumerated

Figure 8.11-17 Delete time-based scheduling groups

c. To delete all time-based scheduling groups, N shall be set to 0.

8.11.2.24 TC[11,24] enable time-based scheduling groups

a. Each telecommand packet transporting a request to enable time-based scheduling groups shall be of message subtype 24.

NOTE For the corresponding system requirements, refer to clause 6.11.6.3.1.

b. For each telecommand packet transporting a request to enable time-based scheduling groups, the application data field shall have the structure specified in Figure 8.11-18.

repeated N times

N	group ID
unsigned integer	enumerated

Figure 8.11-18 Enable time-based scheduling groups

c. To enable all time-based scheduling groups, N shall be set to 0.

8.11.2.25 TC[11,25] disable time-based scheduling groups

a. Each telecommand packet transporting a request to disable time-based scheduling groups shall be of message subtype 25.

NOTE For the corresponding system requirements, refer to clause 6.11.6.3.2.

b. For each telecommand packet transporting a request to disable time-based scheduling groups, the application data field shall have the structure specified in Figure 8.11-19.

repeated N times

N	group ID
unsigned integer	enumerated

Figure 8.11-19 Disable time-based scheduling groups

c. To disable all time-based scheduling groups, N shall be set to 0.

8.11.2.26 TC[11,26] report the status of each time-based scheduling group

a. Each telecommand packet transporting a request to report the status of each time-based scheduling group shall be of message subtype 26.

NOTE For the corresponding system requirements, refer to clause 6.11.6.3.3.

b. For each telecommand packet transporting a request to report the status of each time-based scheduling group, the application data field shall be omitted.

8.11.2.27 TM[11,27] time-based scheduling group status report

a. Each telemetry packet transporting a time-based scheduling group status report shall be of message subtype 27.

NOTE For the corresponding system requirements, refer to clause 6.11.6.3.3.

b. For each telemetry packet transporting a time-based scheduling group status report, the source data field shall have the structure specified in Figure 8.11-20.

repeated N times

N	group ID	group status
unsigned integer	enumerated	enumerated

NOTE For the group status enumerated values, refer to requirement 8.11.3b.

Figure 8.11-20 Time-based scheduling group status report

8.11.3 Enumeration

a. The values of the sub-schedule status shall be as specified in Table 8.11-1.

Table 8.11-1 Service 11 sub-schedule status

engineering value	raw value
"disabled"	0
"enabled"	1

b. The values of the group status shall be as specified in Table 8.11-2.

Table 8.11-2 Service 11 group status

engineering value	raw value
"disabled"	0
"enabled"	1

c. The values of the type of time window shall be as specified in Table 8.11-3.

Table 8.11-3 Service 11 type of time window

engineering value	raw value
"select all"	0
"from time tag to time tag"	1
"from time tag"	2
"to time tag"	3

8.12 ST[12] on-board monitoring

8.12.1 General

a. Each packet transporting an on-board monitoring message shall be of service type 12.

8.12.2 Requests and reports

8.12.2.1 TC[12,1] enable parameter monitoring definitions

a. Each telecommand packet transporting a request to enable parameter monitoring definitions shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.12.3.6.1.

b. For each telecommand packet transporting a request to enable parameter monitoring definitions, the application data field shall have the structure specified in Figure 8.12-1.

repeated N times

N	PMON ID
unsigned integer	enumerated

Figure 8.12-1 Enable parameter monitoring definitions

8.12.2.2 TC[12,2] disable parameter monitoring definitions

a. Each telecommand packet transporting a request to disable parameter monitoring definitions shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.12.3.6.2.

b. For each telecommand packet transporting a request to disable parameter monitoring definitions, the application data field shall have the structure specified in Figure 8.12-2.

repeated N times

N	PMON ID
unsigned integer	enumerated

Figure 8.12-2 Disable parameter monitoring definitions

8.12.2.3 TC[12,3] change the maximum transition reporting delay

a. Each telecommand packet transporting a request to change the maximum transition reporting delay shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.12.3.7.2.

b. For each telecommand packet transporting a request to change the maximum transition reporting delay, the application data field shall have the structure specified in Figure 8.12-3.

max. reporting delay
unsigned integer

Figure 8.12-3 Change the maximum transition reporting delay

8.12.2.4 TC[12,4] delete all parameter monitoring definitions

a. Each telecommand packet transporting a request to delete all parameter monitoring definitions shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.12.3.8.2.

b. For each telecommand packet transporting a request to delete all parameter monitoring definitions, the application data field shall be omitted.

8.12.2.5 8.TC[12,5] add parameter monitoring definitions

a. Each telecommand packet transporting a request to add parameter monitoring definitions shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.12.3.8.1.

b. For each telecommand packet transporting a request to add parameter monitoring definitions, the application data field shall have the structure specified in Figure 8.12-4.

repeated N times

N	PMON ID	monitored parameter ID	check validity condition			monitoring interval	repetition number	check type	check type dependent criteria (see below)
			validity parameter ID	mask	expected value				
unsigned integer	enumerated	enumerated	enumerated	bit-string (deduced size)	deduced	unsigned integer	unsigned integer	enumerated	

optional *optional*

NOTE 1 For the check type enumerated values, refer to requirement 8.12.3.1a.
NOTE 2 In the check validity condition field, the size of the mask field and the format of the expected value field are specific to the validity parameter identified by its parameter ID field.
NOTE 3 The structure of the check type dependent criteria field is driven by requirement 8.12.2.5c for expected-value-checking, requirement 8.12.2.5e for limit-checking and requirement 8.12.2.5f for delta-checking.

Figure 8.12-4 Add parameter monitoring definitions

c. For expected-value-checking, the check type dependent criteria field of the add parameter monitoring definitions request shall have the structure specified in Figure 8.12-5.

mask	spare	expected value	event definition ID
bit-string (deduced size)	bit-string (of event definition ID field size)	deduced	enumerated

optional

NOTE 1 The size of the mask field and the structure and format of the expected value field are derived from the monitored parameter identified by the monitored parameter ID field (refer to Figure 8.12-4).

NOTE 2 The spare field can be used for harmonising the size of all check types.

NOTE 3 The value 0 for in the event definition ID field means "no event report to generate".

Figure 8.12-5 Add parameter monitoring definitions: expected-value-checking definition fields

d. For expected-value-checking, the presence of the spare field in the expected-value-checking definition fields of the requests to add parameter monitoring definitions shall be declared when specifying the parameter monitoring subservice.

e. For limit-checking, the check type dependent criteria field of the add parameter monitoring definitions request shall have the structure specified in Figure 8.12-6.12-6.

low limit	event definition ID	high limit	event definition ID
deduced	enumerated	deduced	enumerated

NOTE 1 The structure and format of the low limit and the high limit fields are derived from the monitored parameter identified by the monitored parameter ID field (refer to Figure 8.12-4).

NOTE 2 The value 0 for in the event definition ID field means "no event report to generate".

Figure 8.12-6 Add parameter monitoring definitions: limit-checking definition fields

f. For delta-checking, the check type dependent criteria field of the add parameter monitoring definitions request shall have the structure specified in Figure 8.12-7.

low delta threshold	event definition ID	high delta threshold	event definition ID	number of consecutive delta values
deduced	enumerated	deduced	enumerated	unsigned integer

NOTE 1 The structure and format of the low delta threshold and high delta threshold are derived from the monitored parameter identified by the monitored parameter ID field (refer to Figure 8.12-4).

NOTE 2 The value 0 for in the event definition ID field means "no event report to generate".

Figure 8.12-7 Add parameter monitoring definitions: delta-checking definition fields

8.12.2.6 TC[12,6] delete parameter monitoring definitions

a. Each telecommand packet transporting a request to delete parameter monitoring definitions shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.12.3.8.3.

b. For each telecommand packet transporting a request to delete parameter monitoring definitions, the application data field shall have the structure specified in Figure 8.12-8.

repeated N times

N	PMON ID
unsigned integer	enumerated

Figure 8.12-8 Delete parameter monitoring definitions

8.12.2.7 TC[12,7] modify parameter monitoring definitions

a. Each telecommand packet transporting a request to modify parameter monitoring definitions shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.12.3.8.4.

b. For each telecommand packet transporting a request to modify parameter monitoring definitions, the application data field shall have the structure specified in Figure 8.12-9.

repeated N times

N	PMON ID	monitored parameter ID	repetition number	check type	check type dependent criteria (see below)
unsigned integer	enumerated	enumerated	unsigned integer	enumerated	

NOTE 1 For the check type enumerated values, refer to requirement 8.12.3.1a.
NOTE 2 The structure of the check type dependent criteria field is driven by requirement 8.12.2.7d for expected-value-checking, requirement 8.12.2.7f for limit-checking and requirement 8.12.2.7g for delta-checking.

Figure 8.12-9 Modify parameter monitoring definitions

c. The parameter monitoring subservice shall reject any instruction contained within a modify parameter monitoring definitions request if:

1. that instruction refers to a check type that is different from the original check type specified for that parameter monitoring definition.

NOTE This interface constraint completes requirement 6.12.3.8.4d.

d. For expected-value-checking, the check type dependent criteria field of the modify parameter monitoring definitions request shall have the structure specified in Figure 8.12-10.

mask	spare	expected value	event definition ID
bit-string (deduced size)	bit-string (of event definition ID field size)	deduced	enumerated

optional

NOTE 1 The size of the mask field and the structure and format of the expected value field are derived from the monitored parameter identified by the monitored parameter ID field (refer to Figure 8.12-9).
NOTE 2 The spare field can be used for harmonising the size of all check types.
NOTE 3 The value 0 for in the event definition ID field means "no event report to generate".

Figure 8.12-10 Modify parameter monitoring definitions: expected-value-checking definition fields

e. For expected-value-checking, the presence of the spare field in the expected-value-checking definition fields of the requests to modify parameter monitoring definitions shall be declared when specifying the parameter monitoring subservice.

f. For limit-checking, the check type dependent criteria field of the modify parameter monitoring definitions request shall have the structure specified in Figure 8.12-11.

low limit	event definition ID	high limit	event definition ID
deduced	enumerated	deduced	enumerated

NOTE 1 The structure and format of the low limit and the high limit fields are derived from the monitored parameter identified by the monitored parameter ID field (refer to Figure 8.12-9).

NOTE 2 The value 0 for in the event definition ID field means "no event report to generate".

Figure 8.12-11 Modify parameter monitoring definitions: limit-checking definition fields

NOTE 1 The structure and format of the low limit and the high limit fields are derived from the monitored parameter identified by the monitored parameter ID field (refer to Figure 8.12-9).

NOTE 2 The value 0 for in the event definition ID field means "no event report to generate".

g. For delta-checking, the check type dependent criteria field of the modify parameter monitoring definitions request shall have the structure specified in Figure 8.12-12.

low delta threshold	event definition ID	high delta threshold	event definition ID	number of consecutive delta values
deduced	enumerated	deduced	enumerated	unsigned integer

NOTE 1 The structure and format of the low delta threshold and high delta threshold are derived from the monitored parameter identified by the monitored parameter ID field (refer to Figure 8.12-9).

NOTE 2 The value 0 for in the event definition ID field means "no event report to generate".

Figure 8.12-12 Modify parameter monitoring definitions: limit-checking definition fields

8.12.2.8 TC[12,8] report parameter monitoring definitions

a. Each telecommand packet transporting a request to report parameter monitoring definitions shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.12.3.9.

b. For each telecommand packet transporting a request to report parameter monitoring definitions, the application data field shall have the structure specified in Figure 8.12-13.

repeated N times

N	PMON ID
unsigned integer	enumerated

Figure 8.12-13 Report parameter monitoring definitions

c. To report all parameter monitoring definitions, N shall be set to 0.

8.12.2.9 TM[12,9] parameter monitoring definition report

a. Each telemetry packet transporting a parameter monitoring definition report shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.12.3.9.

b. For each telemetry packet transporting a parameter monitoring definition report, the source data field shall have the structure specified in Figure 8.12-14.

repeated NOE times …

max. transition reporting delay	NOE	PMON ID	monitored parameter ID	check validity condition		
				validity parameter ID	mask	expected value
unsigned integer	unsigned integer	enumerated	enumerated	enumerated	deduced	deduced

optional

… repeated NOE times

monitoring interval	PMON status	repetition number	check type	mask	expected value	event definition ID
unsigned integer	enumerated	unsigned integer	enumerated	deduced	deduced	enumerated

optional

repeated NOL times…

NOL	PMON ID	monitored parameter ID	check validity condition		
			validity parameter ID	mask	expected value
unsigned integer	enumerated	enumerated	enumerated	deduced	deduced

… repeated NOL times

monitoring interval	PMON status	repetition number	check type	low limit	event definition ID	high limit	event definition ID
unsigned integer	enumerated	unsigned integer	enumerated	deduced	enumerated	deduced	enumerated

optional

repeated NOD times…

NOD	PMON ID	monitored parameter ID	check validity condition		
			validity parameter ID	mask	expected value
unsigned integer	enumerated	enumerated	enumerated	deduced	deduced

… repeated NOD times

monitoring interval	PMON status	repetition number	check type	low delta threshold	event definition ID	high delta threshold	event definition ID	number of consecutive delta values
unsigned integer	enumerated	unsigned integer	enumerated	deduced	enumerated	deduced	enumerated	unsigned integer

optional

NOTE 1 NOE represents the Number Of Expected Value Checks
NOTE 2 NOL represents the Number Of Limit Checks
NOTE 3 NOD represents the Number Of Delta Checks
NOTE 4 For the check type enumerated values, refer to requirement 8.12.3.1a.
NOTE 5 The size of the mask field and the structure and format of the expected value field are derived from the monitored parameter identified by the monitored parameter ID field
NOTE 6 The structure and format of the low limit and the high limit fields are derived from the monitored parameter identified by the monitored parameter ID field.
NOTE 7 The structure and format of the low delta threshold and high delta threshold are derived from the monitored parameter identified by the monitored parameter ID field
NOTE 8 The value 0 in the event definition ID field means "no event report to generate"

Figure 8.12-14 Parameter monitoring definition report

8.12.2.10 TC[12,10] report the current out-of-limits

a. Each telecommand packet transporting a request to report the current out-of-limits shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.12.3.11.

b. For each telecommand packet transporting a request to report the current out-of-limits, the application data field shall be omitted.

8.12.2.11 TM[12,11] current out-of-limits report

a. Each telemetry packet transporting a current out-of-limits report shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.12.3.11.

b. For each telemetry packet transporting a current out-of-limits report, the source data field shall have the structure specified in Figure 8.12-14.

repeated N times

N	PMON ID	monitored parameter ID	check type	parameter value	limit crossed	previous PMON checking status	current PMON checking status	transition time
unsigned integer	enumerated	enumerated	enumerated	deduced	deduced	enumerated	enumerated	absolute time

NOTE 1 For the check type enumerated values, refer to requirement 8.12.3.1a
NOTE 2 The format of the parameter value field and limit crossed field is specific to the monitored parameter identified by its parameter ID field.
NOTE 3 For the PMON checking status enumerated values, refer to requirement 8.12.3.1b.

Figure 8.12-15 Current out-of-limits report

8.12.2.12 TM[12,12] check transition report

a. Each telemetry packet transporting a check transition report shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.12.3.7.

b. For each telemetry packet transporting a check transition report, the source data field shall have the structure specified in Figure 8.12-16..

repeated N times

N	PMON ID	monitored parameter ID	check type	parameter value	limit crossed	previous PMON checking status	current PMON checking status	transition time
unsigned integer	enumerated	enumerated	enumerated	deduced	deduced	enumerated	enumerated	absolute time

NOTE 1 For the check type enumerated values, refer to requirement 8.12.3.1a.
NOTE 2 The format of the parameter value field and limit crossed field is specific to the monitored parameter identified by its parameter ID field.
NOTE 3 For the PMON checking status enumerated values, refer to requirement 8.12.3.1b.

Figure 8.12-16 Check transition report

8.12.2.13 TC[12,13] report the status of each parameter monitoring definition

a. Each telecommand packet transporting a request to report the status of each parameter monitoring definition shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.12.3.10.

b. For each telecommand packet transporting a request to report the status of each parameter monitoring definition, the application data field shall be omitted.

8.12.2.14 TM[12,14] parameter monitoring definition status report

a. Each telemetry packet transporting a parameter monitoring definition status report shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.12.3.10.

b. For each telemetry packet transporting a parameter monitoring definition status report, the source data field shall have the structure specified in Figure 8.12-17.

repeated N times

N	PMON ID	PMON status	check type	PMON checking Status
unsigned integer	enumerated	enumerated	enumerated	enumerated

NOTE 1 For the PMON status enumerated values, refer to requirement 8.12.3.1c.
NOTE 2 For the check type enumerated values, refer to requirement 8.12.3.1a.
NOTE 3 For the PMON checking status enumerated values, refer to requirement 8.12.3.1b.

Figure 8.12-17 Parameter monitoring definition status report

8.12.2.15 TC[12,15] enable the parameter monitoring function

a. Each telecommand packet transporting a request to enable the parameter monitoring function shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.12.3.5.1.

b. For each telecommand packet transporting a request to enable the parameter monitoring function, the application data field shall be omitted.

8.12.2.16 TC[12,16] disable the parameter monitoring function

a. Each telecommand packet transporting a request to disable the parameter monitoring function shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.12.3.5.2.

b. For each telecommand packet transporting a request to disable the parameter monitoring function, the application data field shall be omitted.

8.12.2.17 TC[12,17] enable the functional monitoring function

a. Each telecommand packet transporting a request to enable the functional monitoring function shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.12.4.4.1.

b. For each telecommand packet transporting a request to enable the functional monitoring function, the application data field shall be omitted.

8.12.2.18 TC[12,18] disable the functional monitoring function

a. Each telecommand packet transporting a request to disable the functional monitoring function shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.12.4.4.2.

b. For each telecommand packet transporting a request to disable the functional monitoring function, the application data field shall be omitted.

8.12.2.19 TC[12,19] enable functional monitoring definitions

a. Each telecommand packet transporting a request to enable functional monitoring definitions shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.12.4.5.2.

b. For each telecommand packet transporting a request to enable functional monitoring definitions, the application data field shall have the structure specified in Figure 8.12-18.

repeated N times

N	FMON ID
unsigned integer	enumerated

Figure 8.12-18 Enable functional monitoring definitions

8.12.2.20 TC[12,20] disable functional monitoring definitions

a. Each telecommand packet transporting a request to disable functional monitoring definitions shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.12.4.5.3.

b. For each telecommand packet transporting a request to disable functional monitoring definitions, the application data field shall have the structure specified in Figure 8.12-19.

repeated N times

N	FMON ID
unsigned integer	enumerated

Figure 8.12-19 Disable functional monitoring definitions

8.12.2.21 TC[12,21] protect functional monitoring definitions

a. Each telecommand packet transporting a request to protect functional monitoring definitions shall be of message subtype 21.

NOTE For the corresponding system requirements, refer to clause 6.12.4.6.1.

b. For each telecommand packet transporting a request to protect functional monitoring definitions, the application data field shall have the structure specified in Figure 8.12-20.

repeated N times

N	FMON ID
unsigned integer	enumerated

Figure 8.12-20 Protect functional monitoring definitions

8.12.2.22 TC[12,22] unprotect functional monitoring definitions

a. Each telecommand packet transporting a request to unprotect functional monitoring definitions shall be of message subtype 22.

NOTE For the corresponding system requirements, refer to clause 6.12.4.6.2.

b. For each telecommand packet transporting a request to unprotect functional monitoring definitions, the application data field shall have the structure specified in Figure 8.12-21.

repeated N times

N	FMON ID
unsigned integer	enumerated

Figure 8.12-21 Unprotect functional monitoring definitions

8.12.2.23 TC[12,23] add functional monitoring definitions

a. Each telecommand packet transporting a request to add functional monitoring definitions shall be of message subtype 23.

NOTE For the corresponding system requirements, refer to clause 6.12.4.7.1.

b. For each telecommand packet transporting a request to add functional monitoring definitions, the application data field shall have the structure specified in Figure 8.12-22.

repeated N1 times

repeated N2 times

N1	FMON ID	check validity condition			event definition ID	minimum PMON failing number	N2	PMON ID
		validity parameter ID	mask	expected value				
unsigned integer	enumerated	enumerated	bit-string (deduced size)	deduced	enumerated	unsigned integer	unsigned integer	enumerated

optional *optional*

NOTE In the check validity condition field, the size of the mask field and the format of the expected value field are specific to the validity parameter identified by its parameter ID field.

Figure 8.12-22 Add functional monitoring definitions

8.12.2.24 TC[12,24] delete functional monitoring definitions

a. Each telecommand packet transporting a request to delete functional monitoring definitions shall be of message subtype 24.

NOTE For the corresponding system requirements, refer to clause 6.12.4.7.2.

b. For each telecommand packet transporting a request to delete functional monitoring definitions, the application data field shall have the structure specified in Figure 8.12-23.

repeated N times

N	FMON ID
unsigned integer	enumerated

Figure 8.12-23 Delete functional monitoring definitions

8.12.2.25 TC[12,25] report functional monitoring definitions

a. Each telecommand packet transporting a request to report functional monitoring definitions shall be of message subtype 25.

NOTE For the corresponding system requirements, refer to clause 6.12.4.8.

b. For each telecommand packet transporting a request to report functional monitoring definitions, the application data field shall have the structure specified in Figure 8.12-24.

repeated N times

N	FMON ID
unsigned integer	enumerated

Figure 8.12-24 Report functional monitoring definitions

c. To report all functional monitoring definitions, N shall be set to 0.

8.12.2.26 TM[12,26] functional monitoring definition report

a. Each telemetry packet transporting a functional monitoring definition report shall be of message subtype 26.

NOTE For the corresponding system requirements, refer to clause 6.12.4.8.

b. For each telemetry packet transporting a functional monitoring definition report, the source data field shall have the structure specified in Figure 8.12-25.

repeated N1 times

repeated N2 times

N1	FMON ID	check validity condition			FMON protection status	FMON status	event definition ID	minimum PMON failing number	N2	PMON ID
		validity parameter ID	mask	expected value						
unsigned integer	enumerated	enumerated	bit-string (deduced size)	deduced	enumerated	enumerated	enumerated	unsigned integer	unsigned integer	enumerated

optional *optional* *optional*

NOTE 1 In the check validity condition field, the size of the mask field and the format of the expected value field are specific to the validity parameter identified by its parameter ID field.
NOTE 2 For the FMON protection status enumerated values, refer to requirement 8.12.3.2a.
NOTE 3 For the FMON status enumerated values, refer to requirement 8.12.3.2b.

Figure 8.12-25 Functional monitoring definition report

8.12.2.27 TC[12,27] report the status of each functional monitoring definition

a. Each telecommand packet transporting a request to report the status of each functional monitoring definition shall be of message subtype 27.

NOTE For the corresponding system requirements, refer to clause 6.12.4.9.

b. For each telecommand packet transporting a request to report the status of each functional monitoring definition, the application data field shall be omitted.

8.12.2.28 TM[12,28] functional monitoring definition status report

a. Each telemetry packet transporting a functional monitoring definition status report shall be of message subtype 28.

NOTE For the corresponding system requirements, refer to clause 6.12.4.9.

b. For each telemetry packet transporting a functional monitoring definition status report, the source data field shall have the structure specified in Figure 8.12-26.

repeated N times

N	FMON ID	FMON protection status	FMON status	FMON checking status
unsigned integer	enumerated	enumerated	enumerated	enumerated

optional

NOTE 1 For the FMON protection status enumerated values, see requirement 8.12.3.2a.
NOTE 2 For the FMON status enumerated values, see requirement 8.12.3.2b.
NOTE 3 For the FMON checking status enumerated values, see requirement 8.12.3.2c.

Figure 8.12-26 Functional monitoring definition status report

8.12.2.29 TC[12,29] enable check transition list logging

a. Each telecommand packet transporting a request to enable the check transition list logging shall be of message subtype 29.

NOTE For the corresponding system requirements, refer to clause 6.12.3.7.3.

b. For telecommand packet transporting a request to enable the check transition list logging, the application data field shall be omitted.

8.12.2.30 TC[12,30] disable check transition list logging

a. Each telecommand packet transporting a request to disable the the check transition list logging shall be of message subtype 30.

NOTE For the corresponding system requirements, refer to clause 6.12.3.7.4.

b. For telecommand packet transporting a request to disable the check transition list logging, the application data field shall be omitted.

8.12.3 Enumeration

8.12.3.1 Parameter monitoring

a. The values of the check type shall be as specified in Table 8.12-1.

Table 8.12-1 Service 12 check type

engineering value	raw value
"expected-value-checking"	0
"limit-checking"	1
"delta-checking"	2

b. The values of the PMON checking status shall be:

1. for expected-value-checking, as specified in Table 8.12-2.

Table 8.12-2 Service 12 PMON checking status for expected-value-checking

engineering value	raw value
"expected value"	0
"unchecked"	1
"invalid"	2
"unexpected value"	3

2. for limit-checking, as specified in Table 8.12-3.

Table 8.12-3 Service 12 PMON checking status for limit-checking

engineering value	raw value
"within limits"	0
"unchecked"	1
"invalid"	2
"below low limit"	3
"above high limit"	4

3. for delta-checking, as specified in Table 8.12-4.

Table 8.12-4 Service 12 PMON checking status for delta-checking

engineering value	raw value
"within thresholds"	0
"unchecked"	1
"invalid"	2
"below low threshold"	3
"above high threshold"	4

c. The values of the PMON status shall be as specified in Table 8.12-5.

Table 8.12-5 Service 12 PMON status

engineering value	raw value
"disabled"	0
"enabled"	1

8.12.3.2 Functional monitoring

a. The values of the FMON protection status shall be as specified in Table 8.12-6.

Table 8.12-6 Service 12 FMON protection status

engineering value	raw value
"unprotected"	0
"protected"	1

b. The values of the FMON status shall be as specified in Table 8.12-7.

Table 8.12-7 Service 12 FMON status

engineering value	raw value
"disabled"	0
"enabled"	1

c. The values of the FMON checking status shall be as specified in Table 8.12-8.

Table 8.12-8 Service 12 FMON checking status

engineering value	raw value
"unchecked"	0
"running"	1
"invalid"	2

engineering value	raw value
"failed"	3

8.13 ST[13] large packet transfer

8.13.1 General

a. Each packet transporting a large packet transfer message shall be of service type 13.

8.13.2 Requests and reports

8.13.2.1 TM[13,1] first downlink part report

a. Each telemetry packet transporting a first downlink part report shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.13.3.3.1.

b. For each telemetry packet transporting a first downlink part report, the source data field shall have the structure specified in Figure 8.13-1.

large message transaction identifier	part sequence number	part
unsigned integer	unsigned integer	fixed octet-string

Figure 8.13-1 First downlink part report

8.13.2.2 TM[13,2] intermediate downlink part report

a. Each telemetry packet transporting an intermediate downlink part report shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.13.3.3.1.

b. For each telemetry packet transporting an intermediate downlink part report, the source data field shall have the structure specified in Figure 8.13-1.

large message transaction identifier	part sequence number	part
unsigned integer	unsigned integer	fixed octet-string

Figure 8.13-2 Intermediate downlink part report

8.13.2.3 TM[13,3] last downlink part report

a. Each telemetry packet transporting a last downlink part report shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.13.3.3.1.

b. For each telemetry packet transporting a last downlink part report, the source data field shall have the structure specified in Figure 8.13-1.

large message transaction identifier	part sequence number	part
unsigned integer	unsigned integer	fixed octet-string of deduced size
NOTE The size of the part field is deduced from the size of the telemetry packet that is transported.		

Figure 8.13-3 Last downlink part report

8.13.2.4 TC[13,9] uplink the first part

a. Each telecommand packet transporting an uplink the first part shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.13.4.3.1.

b. For each telecommand packet transporting an uplink the first part, the application data field shall have the structure specified in Figure 8.13-4.

large message transaction identifier	part sequence number	part
unsigned integer	unsigned integer	fixed octet-string

Figure 8.13-4 Uplink the first part

8.13.2.5 TC[13,10] uplink an intermediate part

a. Each telecommand packet transporting an uplink an intermediate part shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.13.4.3.1.

b. For each telecommand packet transporting an uplink an intermediate part, the application data field shall have the structure specified in Figure 8.13-4.

large message transaction identifier	part sequence number	part
unsigned integer	unsigned integer	fixed octet-string

Figure 8.13-5 Uplink an intermediate part

8.13.2.6 TC[13,11] uplink the last part

a. Each telecommand packet transporting an uplink the last part shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.13.4.3.1.

b. For each telecommand packet transporting an uplink the last part, the application data field shall have the structure specified in Figure 8.13-4.

large message transaction identifier	part sequence number	part
unsigned integer	unsigned integer	fixed octet-string of deduced size

NOTE The size of the part field is deduced from the size of the large telecommand packet that is transported.

Figure 8.13-6 Uplink the last part

8.13.2.7 TM[13,16] large packet uplink abortion report

a. Each telemetry packet transporting a large packet uplink abortion report shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.13.4.3.3.

b. For each telemetry packet transporting a large packet uplink abortion report, the source data field shall have the structure specified in Figure 8.13-7.

large message transaction identifier	failure reason
unsigned integer	enumerated

Figure 8.13-7 Large packet uplink abortion report

8.14 ST[14] real-time forwarding control

8.14.1 General

a. Each packet transporting a real-time forwarding control message shall be of service type 14.

8.14.2 Requests and reports

8.14.2.1 TC[14,1] add report types to the application process forward-control configuration

a. Each telecommand packet transporting a request to add report types to the application process forward-control configuration shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.14.3.5.1.

b. For each telecommand packet transporting a request to add report types to the application process forward-control configuration, the application data field shall have the structure specified in Figure 8.14-1.

repeated N1 times

repeated N2 times

repeated N3 times

N1	application process ID	N2	service type	N3	message subtype
unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.14-1 Add report types to the application process forward-control configuration

c. To add all report types of an application process to the application process forward-control configuration, N2 shall be set to 0.

d. To add all report types of a service type to the application process forward-control configuration, N3 shall be set to 0.

8.14.2.2 TC[14,2] delete report types from the application process forward-control configuration

a. Each telecommand packet transporting a request to delete report types from the application process forward-control configuration shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.14.3.5.2.

b. For each telecommand packet transporting a request to delete report types from the application process forward-control configuration, the application data field shall have the structure specified in Figure 8.14-2.

repeated N1 times

repeated N2 times

repeated N3 times

N1	application process ID	N2	service type	N3	message subtype
unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.14-2 Delete report types from the application process forward-control configuration

c. To empty the application process forward-control configuration, N1 shall be set to 0.

d. To delete an application process from the application process forward-control configuration, N2 shall be set to 0.

e. To delete a service type from the application process forward-control configuration, N3 shall be set to 0.

8.14.2.3 TC[14,3] report the content of the application process forward-control configuration

a. Each telecommand packet transporting a request to report the content of the application process forward-control configuration shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.14.3.5.3.

b. For each telecommand packet transporting a request to report the content of the application process forward-control configuration, the application data field shall be omitted.

8.14.2.4 TM[14,4] application process forward-control configuration content report

a. Each telemetry packet transporting an application process forward-control configuration content report shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.14.3.5.3.

b. For each telemetry packet transporting an application process forward-control configuration content report, the source data field shall have the structure specified in Figure 8.14-3.

repeated N1 times

repeated N2 times

repeated N3 times

N1	application process ID	N2	service type	N3	message subtype
unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.14-3 Application process forward-control configuration content report

c. To report that the application process forward-control configuration is empty, N1 shall be set to 0.

d. To report that no service type of the related application process is in the application process forward-control configuration, N2 shall be set to 0.

e. To report that no message type of the related application process and service type is in the application process forward-control configuration, N3 shall be set to 0.

8.14.2.5 TC[14,5] add structure identifiers to the housekeeping parameter report forward-control configuration

a. Each telecommand packet transporting a request to add structure identifiers to the housekeeping parameter report forward-control configuration shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.14.3.6.1.

b. For each telecommand packet transporting a request to add structure identifiers to the housekeeping parameter report forward-control configuration, the application data field shall have the structure specified in Figure 8.14-4.

repeated N1 times

repeated N2 times

N1	application process ID	N2	housekeeping parameter report structure ID	subsampling rate
unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer

optional

Figure 8.14-4 Add structure identifiers to the housekeeping parameter report forward-control configuration

c. To add all structure identifiers of an application process to the housekeeping parameter report forward-control configuration, N2 shall be set to 0.

8.14.2.6 TC[14,6] delete structure identifiers from the housekeeping parameter report forward-control configuration

a. Each telecommand packet transporting a request to delete structure identifiers from the housekeeping parameter report forward-control configuration shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.14.3.6.2.

b. For each telecommand packet transporting a request to delete structure identifiers from the housekeeping parameter report forward-control configuration, the application data field shall have the structure specified in Figure 8.14-5.

repeated N1 times

repeated N2 times

N1	application process ID	N2	housekeeping parameter report structure ID
unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.14-5 Delete structure identifiers from the housekeeping parameter report forward-control configuration

c. To empty the housekeeping parameter report forward-control configuration, N1 shall be set to 0.

d. To delete an application process from the housekeeping parameter report forward-control configuration, N2 shall be set to 0.

8.14.2.7 TC[14,7] report the content of the housekeeping parameter report forward-control configuration

a. Each telecommand packet transporting a request to report the content of the housekeeping parameter report forward-control configuration shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.14.3.6.3.

b. For each telecommand packet transporting a request to report the content of the housekeeping parameter report forward-control configuration, the application data field shall be omitted.

8.14.2.8 TM[14,8] housekeeping parameter report forward-control configuration content report

a. Each telemetry packet transporting a housekeeping parameter report forward-control configuration content report shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.14.3.6.3.

b. For each telemetry packet transporting a housekeeping parameter report forward-control configuration content report, the source data field shall have the structure specified in Figure 8.14-6.

repeated N1 times

repeated N2 times

N1	application process ID	N2	housekeeping parameter report structure ID	subsampling rate
unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer

optional

Figure 8.14-6 Housekeeping parameter report forward-control configuration content report

c. To report that the housekeeping parameter report forward-control configuration is empty, N1 shall be set to 0.

d. To report that no housekeeping parameter report type of the related application process is in the housekeeping parameter report forward-control configuration, N2 shall be set to 0.

8.14.2.9 TC[14,13] delete event definition identifiers from the event report blocking forward-control configuration

a. Each telecommand packet transporting a request to delete event definition identifiers from the event report blocking forward-control configuration shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.14.3.7.1.

b. For each telecommand packet transporting a request to delete event definition identifiers from the event report blocking forward-control

configuration, the application data field shall have the structure specified in Figure 8.14-7.

repeated N1 times

repeated N2 times

N1	application process ID	N2	event definition ID
unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.14-7 Delete event definition identifiers from the event report blocking forward-control configuration

c. To delete an application process from the event report blocking forward-control configuration, N2 shall be set to 0.

8.14.2.10 TC[14,14] add event definition identifiers to the event report blocking forward-control configuration

a. Each telecommand packet transporting a request to add event definition identifiers to the event report blocking forward-control configuration shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.14.3.7.2.

b. For each telecommand packet transporting a request to add event definition identifiers to the event report blocking forward-control configuration, the application data field shall have the structure specified in Figure 8.14-8.

repeated N1 times

repeated N2 times

N1	application process ID	N2	event definition ID
unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.14-8 Add event definition identifiers to the event report blocking forward-control configuration

c. To add all event definition identifiers of an application process to the event report blocking forward-control configuration, N2 shall be set to 0.

8.14.2.11 TC[14,15] report the content of the event report blocking forward-control configuration

a. Each telecommand packet transporting a request to report the content of the event report blocking forward-control configuration shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.14.3.7.3.

b. For each telecommand packet transporting a request to report the content of the event report blocking forward-control configuration, the application data field shall be omitted.

8.14.2.12 TM[14,16] event report blocking forward-control configuration content report

a. Each telemetry packet transporting an event report blocking forward-control configuration content report shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.14.3.7.3.

b. For each telemetry packet transporting an event report blocking forward-control configuration content report, the source data field shall have the structure specified in Figure 8.14-9.

repeated N1 times

repeated N2 times

N1	application process ID	N2	event definition ID
unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.14-9 Event report blocking forward-control configuration content report

c. To report that the event report blocking forward-control configuration is empty, N1 shall be set to 0.

d. To report that no event definition for the related application process is in the event report blocking forward-control configuration, N2 shall be set to 0.

8.15 ST[15] on-board storage and retrieval

8.15.1 General

a. Each packet transporting an on-board storage and retrieval message shall be of message service type 15.

8.15.2 Requests and reports

8.15.2.1 TC[15,1] enable the storage function of packet stores

a. Each telecommand packet transporting a request to enable the storage function of packet stores shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.15.3.4.2.

b. For each telecommand packet transporting a request to enable the storage function of packet stores, the application data field shall have the structure specified in Figure 8.15-1.

repeated N times

N	packet store ID
unsigned integer	fixed character-string

Figure 8.15-1 Enable the storage function of packet stores

c. To enable the storage function of all packet stores, N shall be set to 0.

8.15.2.2 TC[15,2] disable the storage function of packet stores

a. Each telecommand packet transporting a request to disable the storage function of packet stores shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.15.3.4.3.

b. For each telecommand packet transporting a request to disable the storage function of packet stores, the application data field shall have the structure specified in Figure 8.15-2.

repeated N times

N	packet store ID
unsigned integer	fixed character-string

Figure 8.15-2 Disable the storage function of packet stores

c. To disable the storage function of all packet stores, N shall be set to 0.

8.15.2.3 TC[15,3] add report types to the application process storage-control configuration

a. Each telecommand packet transporting a request to add report types to the application process storage-control configuration shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.15.4.4.1.

b. For each telecommand packet transporting a request to add report types to the application process storage-control configuration, the application data field shall have the structure specified in Figure 8.15-3.

repeated N1 times

repeated N2 times

repeated N3 times

packet store ID	N1	application process ID	N2	service type	N3	message subtype
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

optional

Figure 8.15-3 Add report types to the application process storage-control configuration

c. To add all report types of an application process to the application process storage-control configuration, N2 shall be set to 0.

d. To add all report types of a service type to the application process storage-control configuration, N3 shall be set to 0.

8.15.2.4 TC[15,4] delete report types from the application process storage-control configuration

a. Each telecommand packet transporting a request to delete report types from the application process storage-control configuration shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.15.4.4.2.

b. For each telecommand packet transporting a request to delete report types from the application process storage-control configuration, the application data field shall have the structure specified in Figure 8.15-4.

repeated N1 times

repeated N2 times

repeated N3 times

packet store ID	N1	application process ID	N2	service type	N3	message subtype
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

optional

Figure 8.15-4 Delete report types from the application process storage-control configuration

c. To empty the application process storage-control configuration, N1 shall be set to 0.

d. To delete an application process from the application process storage-control configuration, N2 shall be set to 0.

e. To delete a service type from the application process storage-control configuration, N3 shall be set to 0.

8.15.2.5 TC[15,5] report the content of the application process storage-control configuration

a. Each telecommand packet transporting a request to report the content of the application process storage-control configuration shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.15.4.4.3.

b. For each telecommand packet transporting a request to report the content of the application process storage-control configuration, the application data field shall have the structure specified in Figure 8.15-5.

packet store ID
fixed character-string

Figure 8.15-5 Report the content of the application process storage-control configuration

8.15.2.6 TM[15,6] application process storage-control configuration content report

a. Each telemetry packet transporting an application process storage-control configuration content report shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.15.4.4.3.

b. For each telemetry packet transporting an application process storage-control configuration content report, the source data field shall have the structure specified in Figure 8.15-6.

repeated N1 times

repeated N2 times

repeated N3 times

packet store ID	N1	application process ID	N2	service type	N3	message subtype
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

optional

Figure 8.15-6 Application process storage-control configuration content report

8.15.2.7 TC[15,9] start the by-time-range retrieval of packet stores

a. Each telecommand packet transporting a request to start the by-time-range retrieval of packet stores shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.15.3.6.2.

b. For each telecommand packet transporting a request to start the by-time-range retrieval of packet stores, the application data field shall have the structure specified in Figure 8.15-7.

repeated N times

N	packet store ID	from time	to time	retrieval priority
unsigned integer	fixed character-string	absolute time	absolute time	enumerated

optional

Figure 8.15-7 Start the by-time-range retrieval of packet stores

c. To start the by-time-range retrieval with the default retrieval priority, retrieval priority shall be set to 0.

8.15.2.8 TC[15,10] Abort all requests to copy the packets contained in a packet store selected by time window

a. Each telecommand packet transporting a request to abort all requests to copy the packets contained in a packet store selected by time window shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.15.3.8.5.

b. For each telecommand packet transporting a request to abort all requests to copy the packets contained in a packet store selected by time window, the application data field shall be omitted.

8.15.2.9 TC[15,11] delete the content of packet stores up to the specified time

a. Each telecommand packet transporting a request to delete the content of packet stores up to the specified time shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.15.3.8.1.

b. For each telecommand packet transporting a request to delete the content of packet stores up to the specified time, the application data field shall have the structure specified in Figure 8.15-8.

repeated N times

storage time	N	packet store ID
absolute time	unsigned integer	fixed character-string

Figure 8.15-8 Delete the content of packet stores up to the specified time

c. To delete the content of all packet stores up to the specified time, N shall be set to 0.

8.15.2.10 TC[15,12] summary-report the content of packet stores

a. Each telecommand packet transporting a request to summary-report the content of packet stores shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.15.3.11.1.

b. For each telecommand packet transporting a request to summary-report the content of packet stores, the application data field shall have the structure specified in Figure 8.15-9.

repeated N times

N	packet store ID
unsigned integer	fixed character-string

Figure 8.15-9 Summary-report the content of packet stores

c. To summary-report the content of each packet store, N shall be set to 0.

8.15.2.11 TM[15,13] packet store content summary report

a. Each telemetry packet transporting a packet store content summary report shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.15.3.11.1.

b. For each telemetry packet transporting a packet store content summary report, the source data field shall have the structure specified in Figure 8.15-10.

repeated N times

N	packet store ID	oldest stored packet time	newest stored packet time	current open retrieval start time tag	percentage filled	from open retrieval start time tag percentage filled
unsigned integer	fixed character-string	absolute time	absolute time	absolute time	unsigned integer	unsigned integer

Figure 8.15-10 Packet store content summary report

8.15.2.12 TC[15,14] change the open retrieval start time tag of packet stores

a. Each telecommand packet transporting a request to change the open retrieval start time tag of packet stores shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.15.3.5.2.

b. For each telecommand packet transporting a request to change the open retrieval start time tag of packet stores, the application data field shall have the structure specified in Figure 8.15-11.

repeated N times

open retrieval start time tag	N	packet store ID
absolute time	unsigned integer	fixed character-string

Figure 8.15-11 Change the open retrieval start time tag of packet stores

c. To change the open retrieval start time tag of all packet stores, N shall be set to 0.

8.15.2.13 TC[15,15] resume the open retrieval of packet stores

a. Each telecommand packet transporting a request to resume the open retrieval of packet stores shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.15.3.5.3.

b. For each telecommand packet transporting a request to resume the open retrieval of packet stores, the application data field shall have the structure specified in Figure 8.15-12.

repeated N times

N	packet store ID	open retrieval policy	retrieval priority
unsigned integer	fixed character-string	enumerated	enumerated

optional

Figure 8.15-12 Resume the open retrieval of packet stores

c. To resume the open retrieval of all packet stores, N shall be set to 0.

d. To resume the open retrieval with the default retrieval priority, retrieval priority shall be set to 0.

8.15.2.14 TC[15,16] suspend the open retrieval of packet stores

a. Each telecommand packet transporting a request to suspend the open retrieval of packet stores shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.15.3.5.4.

b. For each telecommand packet transporting a request to suspend the open retrieval of packet stores, the application data field shall have the structure specified in Figure 8.15-13.

repeated N times

N	packet store ID
unsigned integer	fixed character-string

Figure 8.15-13 Suspend the open retrieval of packet stores

c. To suspend the open retrieval of all packet stores, N shall be set to 0.

8.15.2.15 TC[15,17] abort the by-time-range retrieval of packet stores

a. Each telecommand packet transporting a request to abort the by-time-range retrieval of packet stores shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.15.3.6.3.

b. For each telecommand packet transporting a request to abort the by-time-range retrieval of packet stores, the application data field shall have the structure specified in Figure 8.15-14.

repeated N times

N	packet store ID
unsigned integer	fixed character-string

Figure 8.15-14 Abort the by-time-range retrieval of packet stores

c. To abort the by-time-range retrieval of all packet stores, N shall be set to 0.

8.15.2.16 TC[15,18] report the status of each packet store

a. Each telecommand packet transporting a request to report the status of each packet store shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.15.3.7.

b. For each telecommand packet transporting a request to report the status of each packet store, the application data field shall be omitted.

8.15.2.17 TM[15,19] packet store status report

a. Each telemetry packet transporting a packet store status report shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.15.3.7.

b. For each telemetry packet transporting a packet store status report, the source data field shall have the structure specified in Figure 8.15-15.

repeated N times

N	packet store ID	packet store status	packet store open retrieval status	packet store by-time-range retrieval status
unsigned integer	fixed character-string	enumerated	enumerated	enumerated

optional

Figure 8.15-15 Packet store status report

8.15.2.18 TC[15,20] create packet stores

a. Each telecommand packet transporting a request to create packet stores shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.15.3.9.1.

b. For each telecommand packet transporting a request to create packet stores, the application data field shall have the structure specified in Figure 8.15-16.

repeated N times

N	packet store ID	packet store size	default open retrieval policy	packet store type	packet store virtual channel	default retrieval priority
unsigned integer	fixed character-string	unsigned integer	enumerated	enumerated	enumerated	enumerated

optional *optional* *optional*

Figure 8.15-16 Create packet stores

8.15.2.19 TC[15,21] delete packet stores

a. Each telecommand packet transporting a request to delete packet stores shall be of message subtype 21.

NOTE For the corresponding system requirements, refer to clause 6.15.3.9.2.

b. For each telecommand packet transporting a request to delete packet stores, the application data field shall have the structure specified in Figure 8.15-17.

repeated N times

N	packet store ID
unsigned integer	fixed character-string

Figure 8.15-17 Delete packet stores

c. To delete all packet stores, N shall be set to 0.

8.15.2.20 TC[15,22] report the configuration of each packet store

a. Each telecommand packet transporting a request to report the configuration of each packet store shall be of message subtype 22.

NOTE For the corresponding system requirements, refer to clause 6.15.3.9.3.

b. For each telecommand packet transporting a request to report the configuration of each packet store, the application data field shall be omitted.

8.15.2.21 TM[15,23] packet store configuration report

a. Each telemetry packet transporting a packet store configuration report shall be of message subtype 23.

NOTE For the corresponding system requirements, refer to clause 6.15.3.9.3.

b. For each telemetry packet transporting a packet store configuration report, the source data field shall have the structure specified in Figure 8.15-18.

repeated N times

N	packet store ID	packet store size	default open retrieval policy	packet store type	packet store virtual channel	default retrieval priority
unsigned integer	fixed character-string	unsigned integer	enumerated	enumerated	enumerated	enumerated

optional *optional* *optional*

Figure 8.15-18 Packet store configuration report

8.15.2.22 TC[15,24] copy the packets contained in a packet store selected by time window

a. Each telecommand packet transporting a request to copy the packets contained in a packet store selected by time window shall be of message subtype 24.

NOTE For the corresponding system requirements, refer to clause 6.15.3.9.4.

b. For each telecommand packet transporting a request to copy the packets contained in a packet store selected by time window, the application data field shall have the structure specified in Figure 8.15-19.

time window			from packet store ID	to packet store ID
type	time tag 1	time tag 2		
enumerated	absolute time	absolute time	fixed character-string	fixed character-string
	deduced presence	*deduced presence*		

Figure 8.15-19 Copy the packets contained in a packet store selected by time window

8.15.2.23 TC[15,25] resize packet stores

a. Each telecommand packet transporting a request to resize packet stores shall be of message subtype 25.

NOTE For the corresponding system requirements, refer to clause 6.15.3.10.1.

b. For each telecommand packet transporting a request to resize packet stores, the application data field shall have the structure specified in Figure 8.15-20.

repeated N times

N	packet store ID	packet store size
unsigned integer	fixed character-string	unsigned integer

Figure 8.15-20 Resize packet stores

8.15.2.24 TC[15,26] change a packet store type to circular

a. Each telecommand packet transporting a request to change a packet store type to circular shall be of message subtype 26.

NOTE For the corresponding system requirements, refer to clause 6.15.3.10.2.

b. For each telecommand packet transporting a request to change a packet store type to circular, the application data field shall have the structure specified in Figure 8.15-21.

packet store ID
fixed character-string

Figure 8.15-21 Change a packet store type to circular

8.15.2.25 TC[15,27] change a packet store type to bounded

a. Each telecommand packet transporting a request to change a packet store type to bounded shall be of message subtype 27.

NOTE For the corresponding system requirements, refer to clause 6.15.3.10.3.

b. For each telecommand packet transporting a request to change a packet store type to bounded, the application data field shall have the structure specified in Figure 8.15-22.

packet store ID
fixed character-string

Figure 8.15-22 Change a packet store type to bounded

8.15.2.26 TC[15,28] change the virtual channel used by a packet store

a. Each telecommand packet transporting a request to change the virtual channel used by a packet store shall be of message subtype 28.

NOTE For the corresponding system requirements, refer to clause 6.15.3.10.4.

b. For each telecommand packet transporting a request to change the virtual channel used by a packet store, the application data field shall have the structure specified in Figure 8.15-23.

packet store ID	packet store virtual channel
fixed character-string	enumerated

Figure 8.15-23 Change the virtual channel used by a packet store

8.15.2.27 TC[15,29] add structure identifiers to the housekeeping parameter report storage-control configuration

a. Each telecommand packet transporting a request to add structure identifiers to the housekeeping parameter report storage-control configuration shall be of message subtype 29.

NOTE For the corresponding system requirements, refer to clause 6.15.4.5.1.

b. For each telecommand packet transporting a request to add structure identifiers to the housekeeping parameter report storage-control

configuration, the application data field shall have the structure specified in Figure 8.15-24.

repeated N1 times

repeated N2 times

packet store ID	N1	application process ID	N2	housekeeping parameter report structure ID	subsampling rate
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer

optional *optional*

Figure 8.15-24 Add structure identifiers to the housekeeping parameter report storage-control configuration

c. To add all structure identifiers to the housekeeping parameter report storage-control configuration, N2 shall be set to 0.

8.15.2.28 TC[15,30] delete structure identifiers from the housekeeping parameter report storage-control configuration

a. Each telecommand packet transporting a request to delete structure identifiers from the housekeeping parameter report storage-control configuration shall be of message subtype 30.

NOTE For the corresponding system requirements, refer to clause 6.15.4.5.2.

b. For each telecommand packet transporting a request to delete structure identifiers from the housekeeping parameter report storage-control configuration, the application data field shall have the structure specified in Figure 8.15-25.

repeated N1 times

repeated N2 times

packet store ID	N1	application process ID	N2	housekeeping parameter report structure ID
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated

optional

Figure 8.15-25 Delete structure identifiers from the housekeeping parameter report storage-control configuration

c. To empty the housekeeping parameter report storage-control configuration, N1 shall be set to 0.

d. To delete an application process from the housekeeping parameter report storage-control configuration, N2 shall be set to 0.

8.15.2.29 TC[15,33] delete event definition identifiers from the event report blocking storage-control configuration

a. Each telecommand packet transporting a request to delete event definition identifiers from the event report blocking storage-control configuration shall be of message subtype 33.

NOTE For the corresponding system requirements, refer to clause 6.15.4.6.2.

b. For each telecommand packet transporting a request to delete event definition identifiers from the event report blocking storage-control configuration, the application data field shall have the structure specified in Figure 8.15-26.

repeated N1 times

repeated N2 times

packet store ID	N1	application process ID	N2	event definition ID
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated

optional

Figure 8.15-26 Delete event definition identifiers from the event report blocking storage-control configuration

c. To empty empty the event report blocking storage-control configuration, N1 shall be set to 0.

d. To delete an application process from the event report blocking storage-control configuration, N2 shall be set to 0.

8.15.2.30 TC[15,34] add event definition identifiers to the event report blocking storage-control configuration

a. Each telecommand packet transporting a request to add event definition identifiers to the event report blocking storage-control configuration shall be of message subtype 34.

NOTE For the corresponding system requirements, refer to clause 6.15.4.6.1.

b. For each telecommand packet transporting a request to add event definition identifiers to the event report blocking storage-control

configuration, the application data field shall have the structure specified in Figure 8.15-27.

repeated N1 times

repeated N2 times

packet store ID	N1	application process ID	N2	event definition ID
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated

optional

Figure 8.15-27 Add event definition identifiers to the event report blocking storage-control configuration

c. To add all event definition identifiers to the event report blocking storage-control configuration, N2 shall be set to 0.

8.15.2.31 TC[15,35] report the content of the housekeeping parameter report storage-control configuration

a. Each telecommand packet transporting a request to report the content of the housekeeping parameter report storage-control configuration shall be of message subtype 35.

NOTE For the corresponding system requirements, refer to clause 6.15.4.5.3.

b. For each telecommand packet transporting a request to report the content of the housekeeping parameter report storage-control configuration, the application data field shall have the structure specified in Figure 8.15-28.

packet store ID
fixed character-string

Figure 8.15-28 Report the content of the housekeeping parameter report storage-control configuration

8.15.2.32 TM[15,36] housekeeping parameter report storage-control configuration content report

a. Each telemetry packet transporting a housekeeping parameter report storage-control configuration content report shall be of message subtype 36.

NOTE For the corresponding system requirements, refer to clause 6.15.4.5.3.

b. For each telemetry packet transporting a housekeeping parameter report storage-control configuration content report, the source data field shall have the structure specified in Figure 8.15-29.

repeated N1 times

repeated N2 times

packet store ID	N1	application process ID	N2	housekeeping parameter report structure ID	subsampling rate
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer

optional *optional*

Figure 8.15-29 Housekeeping parameter report storage-control configuration content report

8.15.2.33 TC[15,39] report the content of the event report blocking storage-control configuration

a. Each telecommand packet transporting a request to report the content of the event report blocking storage-control configuration shall be of message subtype 39.

NOTE For the corresponding system requirements, refer to clause 6.15.4.6.3.

b. For each telecommand packet transporting a request to report the content of the event report blocking storage-control configuration, the application data field shall have the structure specified in Figure 8.15-30.

packet store ID
fixed character-string

Figure 8.15-30 Report the content of the event report blocking storage-control configuration

8.15.2.34 TM[15,40] event report blocking storage-control configuration content report

a. Each telemetry packet transporting an event report blocking storage-control configuration content report shall be of message subtype 40.

NOTE For the corresponding system requirements, refer to clause 6.15.4.6.3.

b. For each telemetry packet transporting an event report blocking storage-control configuration content report, the source data field shall have the structure specified in Figure 8.15-31.

repeated N1 times

repeated N2 times

packet store ID	N1	application process ID	N2	event definition ID
fixed character-string	unsigned integer	enumerated	unsigned integer	enumerated

optional

Figure 8.15-31 Event report blocking storage-control configuration content report

8.15.2.35 TC[15,41] change the default retrieval priority of a packet store

a. Each telecommand packet transporting a request to change the default retrieval priority of a packet store shall be of message subtype 41.

NOTE For the corresponding system requirements, refer to clause 6.15.3.10.5.

b. For each telecommand packet transporting a request to change the default retrieval priority of a packet store, the application data field shall have the structure specified in Figure 8.15-7.

packet store ID	default retrieval priority
fixed character-string	enumerated

Figure 8.15-32 Change default retrieval priority of a packet store

8.15.3 Enumeration

a. The values of the packet store type shall be as specified in Table 8.15-1.

Table 8.15-1 Service 15 packet store type

engineering value	raw value
"circular type"	0
"bounded type"	1

b. The values of the packet store time ranged retrieval status shall be as specified in Table 8.15-2.

Table 8.15-2 Service 15 packet store time range retrieval status

engineering value	raw value
"disabled"	0
"enabled"	1

c. The values of the packet store open retrieval status shall be as specified in Table 8.15-3.

Table 8.15-3 Service 15 packet store open retrieval status

engineering value	raw value
"suspended"	0
"in progress"	1

d. The values of the packet store open retrieval policy shall be as specified in Table 8.15-4.

Table 8.15-4 Service 15 packet store open retrieval policy

engineering value	raw value
"suspend"	0
"stay open"	1

8.16 ST[16] (reserved)

8.17 ST[17] test

8.17.1 General

a. Each packet transporting a test message shall be of service type 17.

8.17.2 Requests and reports

8.17.2.1 TC[17,1] perform an are-you-alive connection test

a. Each telecommand packet transporting a request to perform an are-you-alive connection test shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.17.3.

b. For each telecommand packet transporting a request to perform an are-you-alive connection test, the application data field shall be omitted.

8.17.2.2 TM[17,2] are-you-alive connection test report

a. Each telemetry packet transporting an are-you-alive connection test report shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.17.3.

b. For each telemetry packet transporting an are-you-alive connection test report, the source data field shall be omitted.

8.17.2.3 TC[17,3] perform an on-board connection test

a. Each telecommand packet transporting a request to perform an on-board connection test shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.17.4.2.

b. For each telecommand packet transporting a request to perform an on-board connection test, the application data field shall have the structure specified in Figure 8.17-1.

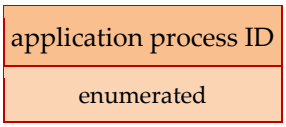

Figure 8.17-1 Perform an on-board connection test

8.17.2.4 TM[17,4] on-board connection test report

a. Each telemetry packet transporting an on-board connection test report shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.17.4.2.

b. For each telemetry packet transporting an on-board connection test report, the source data field shall have the structure specified in Figure 8.17-2.

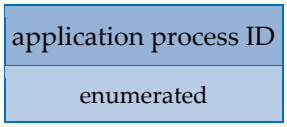

Figure 8.17-2 On-board connection test report

8.17.2.5 TC[17,5] perform a variable size are-you-alive connection test

a. Each telecommand packet transporting a request to perform a variable size are-you-alive connection test shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.17.3.

b. For each telecommand packet transporting a request to perform a variable size are-you-alive connection test, the application data field shall have the structure specified in Figure 8-17-3.

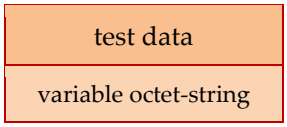

Figure 8.17-3 Perform a variable size are-you-alive connection test

8.17.2.6 TM[17,6] are-you-alive variable size connection test report

a. Each telemetry packet transporting a variable size are-you-alive connection test report shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.17.3.

b. For each telemetry packet transporting a variable size are-you-alive connection test report, the source data field shall have the structure specified in Figure 8-17-4.

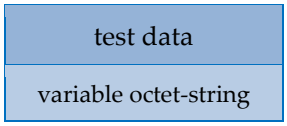

Figure 8.17-4 Variable size are-you-alive connection test report

8.18 ST[18] on-board control procedure

8.18.1 General

a. Each packet transporting an on-board control procedure message shall be of service type 18.

8.18.2 Requests and reports

8.18.2.1 TC[18,1] direct-load an OBCP

a. Each telecommand packet transporting a request to direct-load an OBCP shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.2.

b. For each telecommand packet transporting a request to direct-load an OBCP, the application data field shall have the structure specified in Figure 8.18-1.

OBCP ID	OBCP code	checksum
fixed character-string	variable-length octet-string	bit-string (16 bits)

optional

NOTE The PFC of the length field of the OBCP code is driven by requirement 7.3.8d.

Figure 8.18-1 Direct-load an OBCP

8.18.2.2 TC[18,2] unload OBCPs

a. Each telecommand packet transporting a request to unload OBCPs shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.4.

b. For each telecommand packet transporting a request to unload OBCPs, the application data field shall have the structure specified in Figure 8.18-2.

N	OBCP ID
unsigned integer	fixed character-string

Figure 8.18-2 Unload OBCPs

c. To unload all OBCPs, N shall be set to 0.

8.18.2.3 TC[18,3] activate an OBCP

a. Each telecommand packet transporting a request to activate an OBCP shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.5.

b. For each telecommand packet transporting a request to activate an OBCP, the application data field shall have the structure specified in Figure 8.18-3.

OBCP ID	observability level	argument values
fixed character-string	enumerated	deduced
	optional	*deduced presence*

NOTE 1 For the observability level enumerated values, refer to requirement 8.18.3.1b.

NOTE 2 The presence and structure of the argument values field is driven by the definition of the OBCP indicated by the OBCP ID.

Figure 8.18-3 Activate an OBCP

8.18.2.4 TC[18,4] stop an OBCP

a. Each telecommand packet transporting a request to stop an OBCP shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.7.

b. For each telecommand packet transporting a request to stop an OBCP, the application data field shall have the structure specified in Figure 8.18-4.

OBCP ID	step ID
fixed character-string	enumerated

Figure 8.18-4 Stop an OBCP

c. To stop an OBCP at the end of current step, the step ID field shall be set to 0.

8.18.2.5 TC[18,5] suspend an OBCP

a. Each telecommand packet transporting a request to suspend an OBCP shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.18.4.6.1.

b. For each telecommand packet transporting a request to suspend an OBCP, the application data field shall have the structure specified in Figure 8.18-5.

OBCP ID	step ID
fixed character-string	enumerated

Figure 8.18-5 Suspend an OBCP

c. To suspend an OBCP at the end of current step, the step ID field shall be set to 0.

8.18.2.6 TC[18,6] resume an OBCP

a. Each telecommand packet transporting a request to resume an OBCP shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.18.4.6.2.

b. For each telecommand packet transporting a request to resume an OBCP, the application data field shall have the structure specified in Figure 8.18-6.

OBCP ID
fixed character-string

Figure 8.18-6 Resume an OBCP

8.18.2.7 TC[18,7] communicate parameters to an OBCP

a. Each telecommand packet transporting a request to communicate parameters to an OBCP shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.18.4.7.1.

b. For each telecommand packet transporting a request to communicate parameters to an OBCP, the application data field shall have the structure specified in Figure 8.18-7.

OBCP ID	parameter values
fixed character-string	deduced
NOTE The structure of the argument values field is driven by the definition of the OBCP indicated by the OBCP ID.	

Figure 8.18-7 Communicate parameters to an OBCP

8.18.2.8 TC[18,8] report the execution status of each OBCP

a. Each telecommand packet transporting a request to report the execution status of each OBCP shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.18.4.5.1.

b. For each telecommand packet transporting a request to report the execution status of each OBCP, the application data field shall be omitted.

8.18.2.9 TM[18,9] OBCP execution status report

a. Each telemetry packet transporting an OBCP execution status report shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.18.4.5.1.

b. For each telemetry packet transporting an OBCP execution status report, the source data field shall have the structure specified in Figure 8.18-8.

repeated N times

N	OBCP ID	OBCP checksum	OBCP execution status	step ID
unsigned integer	fixed character-string	bit-string (16 bits)	enumerated	enumerated

optional

NOTE For the OBCP execution status enumerated values, refer to requirement 8.18.3.1a.

Figure 8.18-8 OBCP execution status report

8.18.2.10 TC[18,12] abort an OBCP

a. Each telecommand packet transporting a request to abort an OBCP shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.9.

b. For each telecommand packet transporting a request to abort an OBCP, the application data field shall have the structure specified in Figure 8.18-9.

OBCP ID
fixed character-string

Figure 8.18-9 Abort an OBCP

8.18.2.11 TC[18,13] load an OBCP by reference

a. Each telecommand packet transporting a request to load an OBCP by reference shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.3.

b. For each telecommand packet transporting a request to load an OBCP by reference, the application data field shall have the structure specified in Figure 8.18-10.

OBCP ID	file path	
	repository path	file name
fixed character-string	variable-length character-string	variable-length character-string

optional

Figure 8.18-10 Load an OBCP by reference

8.18.2.12 TC[18,14] activate and execute one OBCP step

a. Each telecommand packet transporting a request to activate and execute one OBCP step shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.18.4.6.3.

b. For each telecommand packet transporting a request to activate and execute one OBCP step, the application data field shall have the structure specified in Figure 8.18-11.

OBCP ID	observability level	argument values
fixed character-string	enumerated	deduced

optional *deduced presence*

NOTE 1 For the observability level enumerated values, refer to requirement 8.18.3.1b.

NOTE 2 The presence and structure of the argument values field is driven by the definition of the OBCP indicated by the OBCP ID.

Figure 8.18-11 Activate and execute one OBCP step

8.18.2.13 TC[18,15] resume and execute one OBCP step

a. Each telecommand packet transporting a request to resume and execute one OBCP step shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.18.4.6.4.

b. For each telecommand packet transporting a request to resume and execute one OBCP step, the application data field shall have the structure specified in Figure 8.18-12.

OBCP ID
fixed character-string

Figure 8.18-12 Resume and execute one OBCP step

8.18.2.14 TC[18,16] set the observability level of OBCPs

a. Each telecommand packet transporting a request to set the observability level of OBCPs shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.18.4.8.1.

b. For each telecommand packet transporting a request to set the observability level of OBCPs, the application data field shall have the structure specified in Figure 8.18-13.

repeated N times

N	OBCP ID	observability level
unsigned integer	fixed character-string	enumerated
NOTE For the observability level enumerated values, refer to requirement 8.18.3.1b.		

Figure 8.18-13 Set the observability level of OBCPs

8.18.2.15 TC[18,17] abort all OBCPs and report

a. Each telecommand packet transporting a request to abort all OBCPs and report shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.10.

b. For each telecommand packet transporting a request to abort all OBCPs and report, the application data field shall be omitted.

8.18.2.16 TM[18,18] aborted OBCP report

a. Each telemetry packet transporting an aborted OBCP report shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.10.

b. For each telemetry packet transporting an aborted OBCP report, the source data field shall have the structure specified in Figure 8.18-14.

repeated N times

N	OBCP ID
unsigned integer	fixed character-string

Figure 8.18-14 Aborted OBCP report

8.18.2.17 TC[18,19] load by reference and activate an OBCP

a. Each telecommand packet transporting a request to load by reference and activate an OBCP shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.6.

b. For each telecommand packet transporting a request to load by reference and activate an OBCP, the application data field shall have the structure specified in Figure 8.18-15.

OBCP ID	file path		observability level	argument values
	repository path	file name		
fixed character-string	variable-length character-string	variable-length character-string	enumerated	deduced
	optional		*optional*	*deduced presence*

NOTE 1 For the observability level enumerated values, refer to requirement 8.18.3.1b.

NOTE 2 The presence and structure of the argument values field is driven by the definition of the OBCP indicated by the OBCP ID.

Figure 8.18-15 Load by reference and activate an OBCP

8.18.2.18 TC[18,20] stop and unload an OBCP

a. Each telecommand packet transporting a request to stop and unload an OBCP shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.18.4.4.8.

b. For each telecommand packet transporting a request to stop and unload an OBCP, the application data field shall have the structure specified in Figure 8.18-16.

OBCP ID	step ID
fixed character-string	enumerated

Figure 8.18-16 Stop and unload an OBCP

c. To stop and unload an OBCP at the end of current step, the step ID field shall be set to 0.

8.18.2.19 TC[18,21] start the OBCP engine

a. Each telecommand packet transporting a request to start the OBCP engine shall be of message subtype 21.

NOTE For the corresponding system requirements, refer to clause 6.18.5.2.1.

b. For each telecommand packet transporting a request to start the OBCP engine, the application data field shall be omitted.

8.18.2.20 TC[18,22] stop the OBCP engine

a. Each telecommand packet transporting a request to stop the OBCP engine shall be of message subtype 22.

NOTE For the corresponding system requirements, refer to clause 6.18.5.2.2.

b. For each telecommand packet transporting a request to stop the OBCP engine, the application data field shall be omitted.

8.18.3 Enumeration

8.18.3.1 OBCP management

a. The OBCP execution status values shall be as specified in Table 8.18-1

Table 8.18-1 Service 18 OBCP execution status

engineering value	raw value
"inactive"	0
"active and running"	1
"active and held"	2

b. The observability level values shall be as specified in Table 8.18-2.

Table 8.18-2 Service 18 Observability level

engineering value	raw value
"no-observability"	0
"at-procedure-level"	1
"at-step-level"	2
"at-detailed-level"	3

NOTE For the meaning of the observability levels, refer to clause 6.18.4.2.

8.19 ST[19] event-action

8.19.1 General

a. Each packet transporting an event-action message shall be of service type 19.

8.19.2 Requests and reports

8.19.2.1 TC[19,1] add event-action definitions

a. Each telecommand packet transporting a request to add event-action definitions shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.19.8.1.

b. For each telecommand packet transporting a request to add event-action definitions, the application data field shall have the structure specified in Figure 8.19-1.

repeated N times

N	event definition system ID		request
	application process ID	event definition ID	
unsigned integer	enumerated	enumerated	TC packet

optional

Figure 8.19-1 Add event-action definitions

8.19.2.2 TC[19,2] delete event-action definitions

a. Each telecommand packet transporting a request to delete event-action definitions shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.19.8.3.

b. For each telecommand packet transporting a request to delete event-action definitions, the application data field shall have the structure specified in Figure 8.19-2.

repeated N times

N	event definition system ID	
	application process ID	event definition ID
unsigned integer	enumerated	enumerated

optional

Figure 8.19-2 Delete event-action definitions

8.19.2.3 TC[19,3] delete all event-action definitions

a. Each telecommand packet transporting a request to delete all event-action definitions shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.19.8.4.

b. For each telecommand packet transporting a request to delete all event-action definitions the application data field shall be omitted.

8.19.2.4 TC[19,4] enable event-action definitions

a. Each telecommand packet transporting a request to enable event-action definitions shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.19.7.1.

b. For each telecommand packet transporting a request to enable event-action definitions, the application data field shall have the structure specified in Figure 8.19-3.

repeated N times

N	event definition system ID	
	application process ID	event definition ID
unsigned integer	enumerated	enumerated

optional

Figure 8.19-3 Enable event-action definitions

c. To enable all event-action definitions, N shall be set to 0.

8.19.2.5 TC[19,5] disable event-action definitions

a. Each telecommand packet transporting a request to disable event-action definitions shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.19.7.2.

b. For each telecommand packet transporting a request to disable event-action definitions, the application data field shall have the structure specified in Figure 8.19-4.

repeated N times

N	event definition system ID	
	application process ID	event definition ID
unsigned integer	enumerated	enumerated

optional

Figure 8.19-4 Disable event-action definitions

c. To disable all event-action definitions, N shall be set to 0.

8.19.2.6 TC[19,6] report the status of each event-action definition

a. Each telecommand packet transporting a request to report the status of each event-action definition shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.19.8.5.

b. For each telecommand packet transporting a request to report the status of each event-action definition, the application data field shall be omitted.

8.19.2.7 TM[19,7] event-action status report

a. Each telemetry packet transporting an event-action status report shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.19.8.5.

b. For each telemetry packet transporting an event-action status report, the source data field shall contain the structure specified in Figure 8.19-5.

repeated N times

N	event definition system ID		event-action status
	application process ID	event definition ID	
unsigned integer	enumerated	enumerated	enumerated

optional

NOTE For the event-action status enumerated values, refer to requirement 8.19.3b.

Figure 8.19-5 Event-action status report

8.19.2.8 TC[19,8] enable the event-action function

a. Each telecommand packet transporting a request to enable the event-action function shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.19.6.1.

b. For each telecommand packet transporting a request to enable the event-action function, the application data field shall be omitted.

8.19.2.9 TC[19,9] disable the event-action function

a. Each telecommand packet transporting a request to disable the event-action function shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.19.6.2.

b. For each telecommand packet transporting a request to disable the event-action function, the application data field shall be omitted.

8.19.2.10 TC[19,10] report event-action definitions

a. Each telecommand packet transporting a request to report event-action definitions shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.19.8.6.

b. For each telecommand packet transporting a request to report event-action definitions, the application data field shall contain the structure specified in Figure 8.19-6.

repeated N times

N	event definition system ID	
	application process ID	event definition ID
unsigned integer	enumerated	enumerated

optional

Figure 8.19-6 Report event-action definitions

c. To report all event-action definitions, N shall be set to 0.

8.19.2.11 TM[19,11] event-action definition report

a. Each telemetry packet transporting an event-action status report shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.19.8.6.

b. For each telemetry packet transporting an event-action status report, the source data field shall contain the structure specified in Figure 8.19-7.

repeated N times

N	event definition system ID		event-action status	request
	application process ID	event definition ID		
unsigned integer	enumerated	enumerated	enumerated	TC packet

optional

NOTE For the event-action status enumerated values, refer to requirement 8.19.3b.

Figure 8.19-7 Event-action definition report

8.19.3 Enumeration

a. The values of the event-action function status shall be as specified in Table 8.19-1.

Table 8.19-1 Service 19 event-action function status

engineering value	raw value
"disabled"	0
"enabled"	1

b. The values of the event-action status shall be as specified in Table 8.19-2.

Table 8.19-2 Service 19 event-action status

engineering value	raw value
"disabled"	0
"enabled"	1

8.20 ST[20] on-board parameter management

8.20.1 General

a. Each packet transporting a parameter management message shall be of service type 20.

8.20.2 Requests and reports

8.20.2.1 TC[20,1] report parameter values

a. Each telecommand packet transporting a request to report parameter values shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.20.4.1.

b. For each telecommand packet transporting a request to report parameter values, the application data field shall have the structure specified in Figure 8.20-1.

repeated N times

N	parameter ID
unsigned integer	enumerated

Figure 8.20-1 Report parameter values

8.20.2.2 TM[20,2] parameter value report

a. Each telemetry packet transporting a parameter value report shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.20.4.1.

b. For each telemetry packet transporting a parameter value report, the source data field shall have the structure specified in Figure 8.20-2.

repeated N times

N	parameter ID	value
unsigned integer	enumerated	deduced
NOTE The format of the value field is specific to the parameter identified by the associated parameter ID field.		

Figure 8.20-2 Parameter value report

8.20.2.3 TC[20,3] set parameter values

a. Each telecommand packet transporting a request to set parameter values shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.20.4.2.

b. For each telecommand packet transporting a request to set parameter values, the application data field shall have the structure specified in Figure 8.20-3.

repeated N times

N	parameter ID	value
unsigned integer	enumerated	deduced
NOTE The format of the value field is specific to the parameter identified by the associated parameter ID field.		

Figure 8.20-3 Set parameter values

8.20.2.4 TC[20,4] change raw memory parameter definitions

a. Each telecommand packet transporting a request to change raw memory parameter definitions shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.20.5.2.

b. For each telecommand packet transporting a request to change raw memory parameter definitions, the application data field shall have the structure specified in Figure 8.20-4.

repeated N times

N	parameter ID	memory ID	absolute address	PTC	PFC
unsigned integer	enumerated	enumerated	unsigned integer	enumerated	enumerated

optional

Figure 8.20-4 Change raw memory parameter definitions

8.20.2.5 TC[20,5] change object memory parameter definitions

a. Each telecommand packet transporting a request to change object memory parameter definitions shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.20.5.3.

b. For each telecommand packet transporting a request to change object memory parameter definitions, the application data field shall have the structure specified in Figure 8.20-4.

repeated N times

N	parameter ID	memory ID	base	offset	PTC	PFC
unsigned integer	enumerated	enumerated	deduced	unsigned integer	enumerated	enumerated

optional

Figure 8.20-5 Change object memory parameter definitions

8.20.2.6 TC[20,6] report parameter definitions

a. Each telecommand packet transporting a request to report parameter definitions shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.20.5.4.

b. For each telecommand packet transporting a request to report parameter definitions, the application data field shall have the structure specified in Figure 8.20-6.

repeated N times

N	parameter ID
unsigned integer	enumerated

Figure 8.20-6 Report parameter definitions

8.20.2.7 TM[20,7] parameter definition report

a. Each telemetry packet transporting a parameter definition report shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.20.5.4.

b. For each telemetry packet transporting a parameter definition report, the source data field shall have the structure specified in Figure 8.20-7.

repeated N times

N	parameter ID	memory ID	*addressing scheme*	*addressing scheme dependent address (see below)*	PTC	PFC
unsigned integer	enumerated	enumerated	enumerated		enumerated	enumerated

optional *optional*

NOTE For the addressing scheme enumerated values, refer to requirement 8.20.3a.

Figure 8.20-7 Parameter definition report

c. For absolute address addressing scheme, the addressing scheme dependent address field of the parameter definition report shall have the structure specified in Figure 8.20-8.

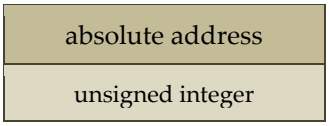

Figure 8.20-8 Parameter definition report: absolute address addressing scheme field

d. For base plus offset addressing scheme, the addressing scheme dependent address field of the parameter definition report shall have the structure specified in Figure 8.20-9.

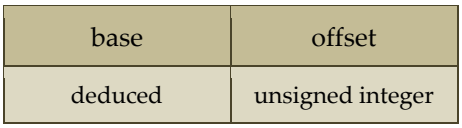

Figure 8.20-9 Parameter definition report: base plus offset addressing scheme field

8.20.3 Enumeration

a. The values of the addressing scheme shall be as specified in Table 8.20-1.

Table 8.20-1 Service 20 addressing scheme

engineering value	raw value
"absolute addressing scheme"	0
"base plus offset addressing scheme"	1

8.21 ST[21] request sequencing

8.21.1 General

a. Each packet transporting a request sequencing message shall be of service type 21.

8.21.2 Requests and reports

8.21.2.1 TC[21,1] direct-load a request sequence

a. Each telecommand packet transporting a request to direct-load a request sequence shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.21.6.2.

b. For each telecommand packet transporting a request to direct-load a request sequence, the application data field shall have the structure specified in Figure 8.21-1.

repeated N times

request sequence ID	N	request	delay
enumerated	unsigned integer	TC packet	relative time

Figure 8.21-1 Direct-load a request sequence

8.21.2.2 TC[21,2] load a request sequence by reference

a. Each telecommand packet transporting a request to load a request sequence by reference shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.21.6.3.

b. For each telecommand packet transporting a request to load a request sequence by reference, the application data field shall have the structure specified in Figure 8.21-2.

request sequence ID	file path	
	repository path	file name
enumerated	variable-length character-string	variable-length character-string

optional

Figure 8.21-2 Load a request sequence by reference

8.21.2.3 TC[21,3] unload a request sequence

a. Each telecommand packet transporting a request to unload a request sequence shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.21.6.4.

b. For each telecommand packet transporting a request to unload a request sequence, the application data field shall have the structure specified in Figure 8.21-3.

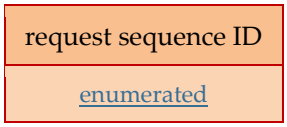

Figure 8.21-3 Unload a request sequence

8.21.2.4 TC[21,4] activate a request sequence

a. Each telecommand packet transporting a request to activate a request sequence shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.21.6.5.

b. For each telecommand packet transporting a request to activate a request sequence, the application data field shall have the structure specified in Figure 8.21-4.

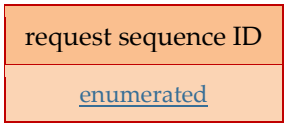

Figure 8.21-4 Activate a request sequence

8.21.2.5 TC[21,5] abort a request sequence

a. Each telecommand packet transporting a request to abort a request sequence shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.21.6.7.

b. For each telecommand packet transporting a request to abort a request sequence, the application data field shall have the structure specified in Figure 8.21-5.

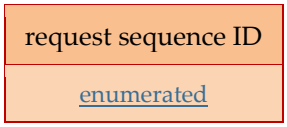

Figure 8.21-5 Abort a request sequence

8.21.2.6 TC[21,6] report the execution status of each request sequence

a. Each telecommand packet transporting a request to report the execution status of each request sequence shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.21.7.

b. For each telecommand packet transporting a request to report the execution status of each request sequence, the application data field shall be omitted.

8.21.2.7 TM[21,7] request sequence execution status report

a. Each telemetry packet transporting a request sequence execution status report shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.21.7.

b. For each telemetry packet transporting a request sequence execution status report, the source data field shall have the structure specified in Figure 8.21-6.

repeated N times

N	request sequence ID	execution status
unsigned integer	enumerated	enumerated
NOTE For the execution status enumerated values, refer to requirement 8.21.3a.		

Figure 8.21-6 Request sequence execution status report

8.21.2.8 TC[21,8] load by reference and activate a request sequence

a. Each telecommand packet transporting a request to load by reference and activate a request sequence shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.21.6.6.

b. For each telecommand packet transporting a request to load by reference and activate a request sequence, the application data field shall have the structure specified in Figure 8.21-7.

request sequence ID	file path	
	repository path	file name
enumerated	variable-length character-string	variable-length character-string

optional

Figure 8.21-7 Load by reference and activate a request sequence

c. To indicate that the next free request sequence identifier will be used, N shall be set to 0

8.21.2.9 TC[21,9] checksum a request sequence

a. Each telecommand packet transporting a request to checksum a request sequence shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.21.8.

b. For each telecommand packet transporting a request to checksum a request sequence, the application data field shall have the structure specified in Figure 8.21-8.

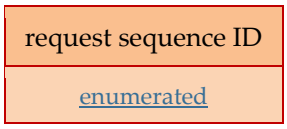

Figure 8.21-8 Checksum a request sequence

8.21.2.10 TM[21,10] request sequence checksum report

a. Each telemetry packet transporting a request sequence checksum report shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.21.8.

b. For each telemetry packet transporting a request sequence checksum report, the source data field shall have the structure specified in Figure 8.21-9.

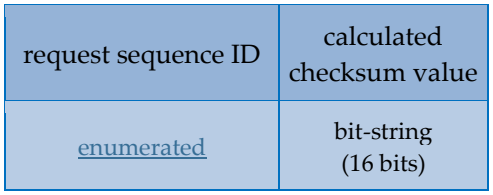

Figure 8.21-9 Request sequence checksum report

8.21.2.11 TC[21,11] report the content of a request sequence

a. Each telecommand packet transporting a request to report the content of a request sequence shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.21.9.

b. For each telecommand packet transporting a request to report the content of a request sequence, the application data field shall have the structure specified in Figure 8.21-10.

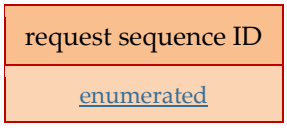

Figure 8.21-10 Report the content of a request sequence

8.21.2.12 TM[21,12] request sequence content report

a. Each telemetry packet transporting a request sequence content report shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.21.9.

b. For each telemetry packet transporting a request sequence content report, the source data field shall have the structure specified in Figure 8.21-11.

repeated N times

request sequence ID	N	request	delay
enumerated	unsigned integer	TC packet	relative time

Figure 8.21-11 Request sequence content report

8.21.2.13 TC[21,13] abort all request sequences and report

a. Each telecommand packet transporting a request to abort all request sequences and report shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.21.6.8.

b. For each telecommand packet transporting a request to abort all request sequences and report, the application data field shall be omitted.

8.21.2.14 TM[21,14] aborted request sequence report

a. Each telemetry packet transporting an aborted request sequence report shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.21.6.8.

b. For each telemetry packet transporting an aborted request sequence report, the source data field shall have the structure specified in Figure 8.21-12.

repeated N times

N	request sequence ID
unsigned integer	enumerated

Figure 8.21-12 Aborted request sequence report

8.21.2.15 TC[21,15] change request sequence directory attributes

a. Each telecommand packet transporting a request to change the attributes of a request sequence directory shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.21.10.

b. For each telecommand packet transporting a request to change the attributes of a request sequence directory, the application data field shall have the structure specified in Figure 8.15-7.

directory path	immediate execution after creation	immediate deletion after execution
variable-length character-string	boolean	boolean

Figure 8.21-13 Change request sequence directory attributes

8.21.2.16 TC[21,16] report the attributes of a request sequence directory

a. Each telecommand packet transporting a request to report the attributes of a request sequence directory shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.21.911.

b. For each telecommand packet transporting a request to report the attributes of a request sequence directory, the application data field shall have the structure specified in Figure 8.21-10.

directory path
variable-length character-string

Figure 8.21-14 Report the attributes of a request sequence directory

8.21.2.17 TM[21,17] request sequence directory attributes report

a. Each telemetry packet transporting a request sequence directory attributes report shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.21.911.

b. For each telemetry packet transporting a request sequence directory attributes report, the source data field shall have the structure specified in Figure 8.21-11.

directory path	immediate execution after creation	immediate deletion after execution
variable-length character-string	boolean	boolean

Figure 8.21-15 Request sequence directory report

8.21.3 Enumeration

a. The values of the request sequence execution status shall be as specified in Table 8.21-11-1.

Table 8.21-1 Service 21 execution status of the request sequence

engineering value	raw value
"inactive"	0
"under execution"	1

8.22 ST[22] position-based scheduling

8.22.1 General

a. Each packet transporting a position-based scheduling message shall be of service type 22.

b. The structure and format of the fields that contain an orbit position shall be declared when specifying the position-based scheduling subservice.

NOTE Refer to clause 6.22.4.

c. The structure and format of the fields that contain a delta position shall be declared when specifying the position-based scheduling subservice.

NOTE Refer to clause 6.22.4.

8.22.2 Requests and reports

8.22.2.1 TC[22,1] enable the position-based schedule execution function

a. Each telecommand packet transporting a request to enable the position-based schedule execution function shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.22.6.3.2.

b. For each telecommand packet transporting a request to enable the position-based schedule execution function, the application data field shall be omitted.

8.22.2.2 TC[22,2] disable the position-based schedule execution function

a. Each telecommand packet transporting a request to disable the position-based schedule execution function shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.22.6.3.3.

b. For each telecommand packet transporting a request to disable the position-based schedule execution function, the application data field shall be omitted.

8.22.2.3 TC[22,3] reset the position-based schedule

a. Each telecommand packet transporting a request to reset the position-based schedule shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.22.6.5.

b. For each telecommand packet transporting a request to reset the position-based schedule, the application data field shall be omitted.

8.22.2.4 TC[22,4] insert activities into the position-based schedule

a. Each telecommand packet transporting a request to insert activities into the position-based schedule shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.22.6.6.

b. For each telecommand packet transporting a request to insert activities into the position-based schedule, the application data field shall have the structure specified in Figure 8.22-1.

repeated N times

sub-schedule ID	N	group ID	position tag	activity persistency status	persistent activity periodicity	request
enumerated	unsigned integer	enumerated	deduced	enumerated	unsigned integer	TC packet

deduced presence

optional *optional* *optional*

NOTE 1 The structure of the position tag field is driven by requirement 8.22.1b.
NOTE 2 For the activity persistency status enumerated values, refer to requirement 8.22.3d.

Figure 8.22-1 Insert activities into the position-based schedule

8.22.2.5 TC[22,5] delete position-based scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to delete position-based scheduled activities identified by request identifier shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.22.11.2.

b. For each telecommand packet transporting a request to delete position-based scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.22-2.

repeated N times

N	request ID		
	source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.22-2 Delete position-based scheduled activities identified by request identifier

8.22.2.6 TC[22,6] delete the position-based scheduled activities identified by a filter

a. Each telecommand packet transporting a request to delete the position-based scheduled activities identified by a filter shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.22.12.3.

b. For each telecommand packet transporting a request to delete the position-based scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.22-3.

repeated N1 times *repeated N2 times*

position window			N1	sub-schedule ID	N2	group ID
type	tag 1	tag 2				
enumerated	deduced	deduced	unsigned integer	enumerated	unsigned integer	enumerated
	deduced presence	*deduced presence*		*optional*		*optional*

NOTE 1 For the type enumerated values, refer to requirement 8.22.3c.
NOTE 2 The structure of the position tag fields is driven by requirement 8.22.1b.

Figure 8.22-3 Delete the position-based scheduled activities identified by a filter

8.22.2.7 TC[22,7] position-shift scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to position-shift scheduled activities identified by request identifier shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.22.11.3.

b. For each telecommand packet transporting a request to position-shift scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.22-4.

repeated N times

delta position	N	request ID		
		source ID	application process ID	sequence count
deduced	unsigned integer	enumerated	enumerated	unsigned integer

NOTE The structure and format of the delta position field are driven by requirement 8.22.1c.

Figure 8.22-4 Position-shift scheduled activities identified by request identifier

8.22.2.8 TC[22,8] position-shift the scheduled activities identified by a filter

a. Each telecommand packet transporting a request to position-shift the scheduled activities identified by a filter shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.22.12.4.

b. For each telecommand packet transporting a request to position-shift the scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.22-5.

repeated N1 times *repeated N2 times*

delta position	position window			N1	sub-schedule ID	N2	group ID
	type	tag 1	tag 2				
deduced	enumerated	deduced	deduced	unsigned integer	enumerated	unsigned integer	enumerated
		deduced presence	*deduced presence*				
				optional		*optional*	

NOTE 1 The structure and format of the delta position field are driven by requirement 8.22.1c.
NOTE 2 For the type enumerated values, refer to requirement 8.22.3c.
NOTE 3 The structure of the position tag fields is driven by requirement 8.22.1b.

Figure 8.22-5 Position-shift the scheduled activities identified by a filter

8.22.2.9 TC[22,9] detail-report position-based scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to detail-report position-based scheduled activities identified by request identifier shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.22.11.5.

b. For each telecommand packet transporting a request to detail-report position-based scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.22-6.

repeated N times

N	request ID		
	source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.22-6 Detail-report position-based scheduled activities identified by request identifier

8.22.2.10 TM[22,10] position-based schedule detail report

a. The telemetry packet transporting a position-based schedule detail report shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.22.9.2.

b. For each telemetry packet transporting a position-based schedule detail report, the source data field shall have the structure specified in Figure 8.22-7.

repeated N times

N	sub-schedule ID	group ID	position tag	activity persistency status	persistent activity periodicity	request
enumerated	enumerated	enumerated	deduced	enumerated	unsigned integer	TC packet
			deduced presence			
	optional	*optional*		*optional*		

NOTE 1 The structure of the position tag field is driven by requirement 8.22.1b.
NOTE 2 For the activity persistency status enumerated values, refer to requirement 8.22.3d.

Figure 8.22-7 Position-based schedule detail report

8.22.2.11 TC[22,11] detail-report the position-based scheduled activities identified by a filter

a. Each telecommand packet transporting a request to detail-report the position-based scheduled activities identified by a filter shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.22.12.6.

b. For each telecommand packet transporting a request to detail-report the position-based scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.22-8.

repeated N1 times *repeated N2 times*

position window			N1	sub-schedule ID	N2	group ID
type	tag 1	tag 2				
enumerated	deduced	deduced	unsigned integer	enumerated	unsigned integer	enumerated
	deduced presence	*deduced presence*	*optional*		*optional*	

NOTE 1 For the type enumerated values, refer to requirement 8.22.3c.
NOTE 2 The structure of the position tag fields is driven by requirement 8.22.1b.

Figure 8.22-8 Detail-report the position-based scheduled activities identified by a filter

8.22.2.12 TC[22,12] summary-report position-based scheduled activities identified by request identifier

a. Each telecommand packet transporting a request to summary-report position-based scheduled activities identified by request identifier shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.22.11.4.

b. For each telecommand packet transporting a request to summary-report position-based scheduled activities identified by request identifier, the application data field shall have the structure specified in Figure 8.22-9.

repeated N times

N	request ID		
	source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	unsigned integer

Figure 8.22-9 Summary-report position-based scheduled activities identified by request identifier

8.22.2.13 TM[22,13] position-based schedule summary report

a. The telemetry packet transporting a position-based schedule summary report shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.22.9.1.

b. For each telemetry packet transporting a position-based schedule summary report, the source data field shall have the structure specified in Figure 8.22-10.

repeated N times

N	sub-schedule ID	group ID	position tag	persistency status	persistent activity periodicity	request ID		
						source ID	application process ID	sequence count
unsigned integer	enumerated	enumerated	deduced	enumerated	unsigned integer	enumerated	enumerated	unsigned integer
			deduced presence					
	optional	*optional*			*optional*			

NOTE 1 The structure of the position tag field is driven by requirement 8.22.1b.
NOTE 2 For the activity persistency status enumerated values, refer to requirement 8.22.3d.

Figure 8.22-10 Position-based schedule summary report

8.22.2.14 TC[22,14] summary-report the position-based scheduled activities identified by a filter

a. Each telecommand packet transporting a request to summary-report the position-based scheduled activities identified by a filter shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.22.12.5.

b. For each telecommand packet transporting a request to summary-report the position-based scheduled activities identified by a filter, the application data field shall have the structure specified in Figure 8.22-11.

repeated N1 times *repeated N2 times*

position window			N1	sub-schedule ID	N2	group ID
type	tag 1	tag 2				
enumerated	deduced	deduced	unsigned integer	enumerated	unsigned integer	enumerated
	deduced presence	*deduced presence*				
			optional		*optional*	

NOTE 1 For the type enumerated values, refer to requirement 8.22.3c.
NOTE 2 The structure of the position tag fields is driven by requirement 8.22.1b.

Figure 8.22-11 Summary-report the position-based scheduled activities identified by a filter

8.22.2.15 TC[22,15] position-shift all scheduled activities

a. Each telecommand packet transporting a request to position-shift all scheduled activities shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.22.10.2.

b. For each telecommand packet transporting a request to position-shift all scheduled activities, the application data field shall have the structure specified in Figure 8.22-12.

delta position
deduced
NOTE The structure and format of the delta position are driven requirement 8.22.1c

Figure 8.22-12 Position-shift all scheduled activities

8.22.2.16 TC[22,16] detail-report all position-based scheduled activities

a. Each telecommand packet transporting a request to detail-report all position-based scheduled activities shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.22.10.4.

b. For each telecommand packet transporting a request to detail-report all position-based scheduled activities, the application data field shall be omitted.

8.22.2.17 TC[22,17] summary-report all position-based scheduled activities

a. Each telecommand packet transporting a request to summary-report all position-based scheduled activities shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.22.10.3.

b. For each telecommand packet transporting a request to summary-report all position-based scheduled activities, the application data field shall be omitted.

8.22.2.18 TC[22,18] report the status of each position-based sub-schedule

a. Each telecommand packet transporting a request to report the status of each position-based sub-schedule shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.22.7.2.3.

b. For each telecommand packet transporting a request to report the status of each position-based sub-schedule, the application data field shall be omitted.

8.22.2.19 TM[22,19] position-based sub-schedule status report

a. Each telemetry packet transporting a position-based sub-schedule status report shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.22.7.2.3.

b. For each telemetry packet transporting a position-based sub-schedule status report, the source data field shall have the structure specified in Figure 8.22-13.

repeated N times

N	sub-schedule ID	sub-schedule status
unsigned integer	enumerated	enumerated
NOTE For the sub-schedule status values, refer to requirement 8.22.3a.		

Figure 8.22-13 Position-based sub-schedule status report

8.22.2.20 TC[22,20] enable position-based sub-schedules

a. Each telecommand packet transporting a request to enable position-based sub-schedules shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.22.7.2.1.

b. For each telecommand packet transporting a request to enable position-based sub-schedules, the application data field shall have the structure specified in Figure 8.22-14.

repeated N times

N	sub-schedule ID
unsigned integer	enumerated

Figure 8.22-14 Enable position-based sub-schedules

c. To enable all position-based sub-schedules, N shall be set to 0.

8.22.2.21 TC[22,21] disable position-based sub-schedules

a. Each telecommand packet transporting a request to disable position-based sub-schedules shall be of message subtype 21.

NOTE For the corresponding system requirements, refer to clause 6.22.7.2.2.

b. For each telecommand packet transporting a request to disable position-based sub-schedules, the application data field shall have the structure specified in Figure 8.22-15.

repeated N times

N	sub-schedule ID
unsigned integer	enumerated

Figure 8.22-15 Disable position-based sub-schedules

c. To disable all position-based sub-schedules, N shall be set to 0.

8.22.2.22 TC[22,22] create position-based scheduling groups

a. Each telecommand packet transporting a request to create position-based scheduling groups shall be of message subtype 22.

NOTE For the corresponding system requirements, refer to clause 6.22.8.2.1.

b. For each telecommand packet transporting a request to create position-based scheduling groups, the application data field shall have the structure specified in Figure 8.22-16.

repeated N times

N	group ID	group status
unsigned integer	enumerated	enumerated
NOTE For the group status values, refer to requirement 8.22.3b.		

Figure 8.22-16 Create position-based scheduling groups

8.22.2.23 TC[22,23] delete position-based scheduling groups

a. Each telecommand packet transporting a request to delete position-based scheduling groups shall be of message subtype 23.

NOTE For the corresponding system requirements, refer to clause 6.22.8.2.2.

b. For each telecommand packet transporting a request to delete position-based scheduling groups, the application data field shall have the structure specified in Figure 8.22-17.

repeated N times

N	group ID
unsigned integer	enumerated

Figure 8.22-17 Delete position-based scheduling groups

c. To delete all position-based scheduling groups, N shall be set to 0.

8.22.2.24 TC[22,24] enable position-based scheduling groups

a. Each telecommand packet transporting a request to enable position-based scheduling groups shall be of message subtype 24.

NOTE For the corresponding system requirements, refer to clause 6.22.8.3.1.

b. For each telecommand packet transporting a request to enable position-based scheduling groups, the application data field shall have the structure specified in Figure 8.22-18.

repeated N times

N	group ID
unsigned integer	enumerated

Figure 8.22-18 Enable position-based scheduling groups

c. To enable all position-based scheduling groups, N shall be set to 0.

8.22.2.25 TC[22,25] disable position-based scheduling groups

a. Each telecommand packet transporting a request to disable position-based scheduling groups shall be of message subtype 25.

NOTE For the corresponding system requirements, refer to clause 6.22.8.3.2.

b. For each telecommand packet transporting a request to disable position-based scheduling groups, the application data field shall have the structure specified in Figure 8.22-19.

repeated N times

N	group ID
unsigned integer	enumerated

Figure 8.22-19 Disable position-based scheduling groups

c. To disable all position-based scheduling groups, N shall be set to 0.

8.22.2.26 TC[22,26] report the status of each position-based scheduling group

a. Each telecommand packet transporting a request to report the status of each position-based scheduling group shall be of message subtype 26.

NOTE For the corresponding system requirements, refer to clause 6.22.8.3.3.

b. For each telecommand packet transporting a request to report the status of each position-based scheduling group, the application data field shall be omitted.

8.22.2.27 TM[22,27] position-based scheduling group status report

a. Each telemetry packet transporting a position-based scheduling group status report shall be of message subtype 27.

NOTE For the corresponding system requirements, refer to clause 6.22.8.3.3.

b. For each telemetry packet transporting a position-based scheduling group status report, the source data field shall have the structure specified in Figure 8.22-20.

repeated N times

N	group ID	group status
unsigned integer	enumerated	enumerated

NOTE For the group status enumerated values, refer to requirement 8.22.3b.

Figure 8.22-20 Position-based scheduling group status report

8.22.2.28 TC[22,28] set the orbit number

a. Each telecommand packet transporting a request to set the orbit number shall be of message subtype 28.

NOTE For the corresponding system requirements, refer to clause 6.22.6.4.

b. For each telecommand packet transporting a request to set the orbit number, the application data field shall have the structure specified in Figure 8.22-21.

orbit number
unsigned integer

Figure 8.22-21 set the orbit number

8.22.3 Enumeration

a. The values of the sub-schedule status shall be as specified in Table 8.22-1.

Table 8.22-1 Service 22 sub-schedule status

engineering value	raw value
"disabled"	0
"enabled"	1

b. The values of the group status shall be as specified in Table 8.22-1.

Table 8.22-2 Service 22 group status

engineering value	raw value
"disabled"	0
"enabled"	1

c. The values of the position window type shall be as specified in Table 8.22-1.

Table 8.22-3 Service 22 position window type

engineering value	raw value
"all"	0
"between 2 position tags"	1
"from position tag"	2
"to position tag"	3

d. The values of the activity persistency status shall be as specified in Table 8.22-1:

Table 8.22-4 Service 22 activity persistency status

engineering value	raw value
"non-persistent"	0
"persistent"	1

8.23 ST[23] file management

8.23.1 General

a. Each packet transporting a file management message shall be of service type 23.

8.23.2 Requests and reports

8.23.2.1 TC[23,1] create a file

a. Each telecommand packet transporting a request to create a file shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.23.4.1.1.

b. For each telecommand packet transporting a request to create a file, the application data field shall have the structure specified in Figure 8.23-1.

file path		maximum size	file locked status	additional file attributes
repository path	file name			
variable-length character-string	variable-length character-string	unsigned integer	Boolean	deduced
	optional		*optional*	

Figure 8.23-1 Create a file

c. If the size of the file to create is not bounded, the maximum size shall be set to 0.

NOTE The concept of bounded file size is driven by requirement 5.4.5c.

8.23.2.2 TC[23,2] delete a file

a. Each telecommand packet transporting a request to delete a file shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.23.4.1.2.

b. For each telecommand packet transporting a request to delete a file, the application data field shall have the structure specified in Figure 8.23-2.

file path	
repository path	file name
variable-length character-string	variable-length character-string

Figure 8.23-2 Delete a file

8.23.2.3 TC[23,3] report the attributes of a file

a. Each telecommand packet transporting a request to report the attributes of a file shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.23.4.2.

b. For each telecommand packet transporting a request to report the attributes of a file, the application data field shall have the structure specified in Figure 8.23-3.

file path	
repository path	file name
variable-length character-string	variable-length character-string

Figure 8.23-3 Report the attributes of a file

8.23.2.4 TM[23,4] file attribute report

a. Each telemetry packet transporting a file attribute report shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.23.4.2.

b. For each telemetry packet transporting a file attribute report, the source data field shall have the structure specified in Figure 8.23-4.

file path		file size	file locked status	additional file attributes
repository path	file name			
variable-length character-string	variable-length character-string	unsigned integer	Boolean	deduced
			optional	*optional*

Figure 8.23-4 File attribute report

8.23.2.5 TC[23,5] lock a file

a. Each telecommand packet transporting a request to lock a file shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.23.4.3.1.

b. For each telecommand packet transporting a request to lock a file, the application data field shall have the structure specified in Figure 8.23-5.

file path	
repository path	file name
variable-length character-string	variable-length character-string

Figure 8.23-5 Lock a file

8.23.2.6 TC[23,6] unlock a file

a. Each telecommand packet transporting a request to unlock a file shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.23.4.3.2.

b. For each telecommand packet transporting a request to unlock a file, the application data field shall have the structure specified in Figure 8.23-6.

file path	
repository path	file name
variable-length character-string	variable-length character-string

Figure 8.23-6 Unlock a file

8.23.2.7 TC[23,7] find files

a. Each telecommand packet transporting a request to find files shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.23.4.4.

b. For each telecommand packet transporting a request to find files, the application data field shall have the structure specified in Figure 8.23-7.

repository path	search pattern
variable-length character-string	variable-length character-string

Figure 8.23-7 Find files

8.23.2.8 TM[23,8] found files report

a. Each telemetry packet transporting a found files report shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.23.4.4.

b. For each telemetry packet transporting a found files report, the source data field shall have the structure specified in Figure 8.23-8.

repeated N times

repository path	search pattern	N	matching file path
variable-length character-string	variable-length character-string	unsigned integer	variable-length character-string

Figure 8.23-8 Found files report

8.23.2.9 TC[23,9] create a directory

a. Each telecommand packet transporting a request to create a directory shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.23.4.6.1.

b. For each telecommand packet transporting a request to create a directory, the application data field shall have the structure specified in Figure 8.23-9.

directory path	
repository path	directory name
variable-length character-string	variable-length character-string

Figure 8.23-9 Create a directory

8.23.2.10 TC[23,10] delete a directory

a. Each telecommand packet transporting a request to delete a directory shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.23.4.6.2.

b. For each telecommand packet transporting a request to delete a directory, the application data field shall have the structure specified in Figure 8.23-10.

directory path	
repository path	directory name
variable-length character-string	variable-length character-string

Figure 8.23-10 Delete a directory

8.23.2.11 TC[23,11] rename a directory

a. Each telecommand packet transporting a request to rename a directory shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.23.4.6.3.

b. For each telecommand packet transporting a request to rename a directory, the application data field shall have the structure specified in Figure 8.23-11.

repository path	old directory name	new directory name
variable-length character-string	variable-length character-string	variable-length character-string

Figure 8.23-11 Rename a directory

8.23.2.12 TC[23,12] report the content of a repository

a. Each telecommand packet transporting a request to report the content of a repository shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.23.4.7.

b. For each telecommand packet transporting a request to report the content of a repository, the application data field shall have the structure specified in Figure 8.23-12.

repository path
variable-length character-string

Figure 8.23-12 Summary-report the content of a repository

8.23.2.13 TM[23,13] repository content report

a. Each telemetry packet transporting a repository content report shall be of message subtype 13.

b. For the corresponding system requirements, refer to clause 6.23.4.7.

NOTE For each telemetry packet transporting a repository content report, the source data field shall have the structure specified in Figure 8.23-13

repeated N times

repository path	N	object type	object name	file size	file locked status	additional attributes
variable-length character-string	unsigned integer	enumerated	variable-length character-string	unsigned integer	boolean	any

optional

Figure 8.23-13 Repository content summary report

NOTE 1 For the object type enumerated values, refer to requirement 8.23.3b.

NOTE 2 File attributes only apply when object type is "file"

8.23.2.14 TC[23,14] copy a file

a. Each telecommand packet transporting a request to copy a file shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.23.5.2.2.

b. For each telecommand packet transporting a request to copy a file, the application data field shall have the structure specified in Figure 8.23-14.

operation ID	source file path		target file path	
	repository path	file name	repository path	file name
unsigned integer	variable-length character-string	variable-length character-string	variable-length character-string	variable-length character-string

Figure 8.23-14 Copy a file

NOTE The target file name is ignored when the source file name contains wildcards, as for each file copied the corresponding source file name is used

8.23.2.15 TC[23,15] move a file

a. Each telecommand packet transporting a request to move a file shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.23.5.2.3.

b. For each telecommand packet transporting a request to move a file, the application data field shall have the structure specified in Figure 8.23-15.

operation ID	source file path		target file path	
	repository path	file name	repository path	file name
unsigned integer	variable-length character-string	variable-length character-string	variable-length character-string	variable-length character-string

Figure 8.23-15 Move a file

NOTE The target file name is ignored when the source file name contains wildcards, as for each file moved the corresponding source file name is used

8.23.2.16 TC[23,16] suspend file copy operations

a. Each telecommand packet transporting a request to suspend file copy operation shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.23.5.3.1.

b. For each telecommand packet transporting a request to suspend file copy operation, the application data field shall have the structure specified in Figure 8.23-16.

repeated N times

N	operation ID
unsigned integer	unsigned integer

Figure 8.23-16 Suspend file copy operation

8.23.2.17 TC[23,17] resume file copy operations

a. Each telecommand packet transporting a request to resume file copy operation shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.23.5.3.2.

b. For each telecommand packet transporting a request to resume file copy operation, the application data field shall have the structure specified in Figure 8.23-17.

repeated N times

N	operation ID
unsigned integer	unsigned integer

Figure 8.23-17 Resume file copy operation

8.23.2.18 TC[23,18] abort file copy operations

a. Each telecommand packet transporting a request to abort file copy operations shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.23.5.4.1.

b. For each telecommand packet transporting a request to abort file copy operations, the application data field shall have the structure specified in Figure 8.23-18.

repeated N times

N	operation ID
unsigned integer	unsigned integer

Figure 8.23-18 Abort file copy operations

8.23.2.19 TC[23,19] suspend all file copy operations involving a repository path

a. Each telecommand packet transporting a request to suspend all file copy operations involving a repository path shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.23.5.3.3.

b. For each telecommand packet transporting a request to suspend all file copy operations involving a repository path, the application data field shall have the structure specified in Figure 8.23-19.

repository path
variable-length character-string

Figure 8.23-19 Suspend all file copy operations involving a repository path

8.23.2.20 TC[23,20] resume all file copy operations involving a repository path

a. Each telecommand packet transporting a request to resume all file copy operations involving a repository path shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.23.5.3.4.

b. For each telecommand packet transporting a request to resume all file copy operations involving a repository path, the application data field shall have the structure specified in Figure 8.23-20.

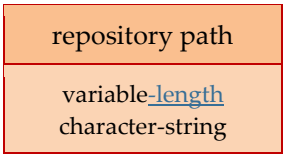

Figure 8.23-20 Resume all file copy operations involving a repository path

8.23.2.21 TC[23,21] abort all file copy operations involving a repository path

a. Each telecommand packet transporting a request to abort all file copy operations involving a repository path shall be of message subtype 21.

NOTE For the corresponding system requirements, refer to clause 6.23.5.4.2.

b. For each telecommand packet transporting a request to abort all file copy operations involving a repository path, the application data field shall have the structure specified in Figure 8.23-21.

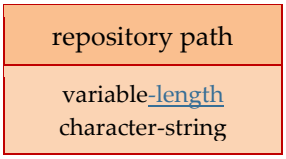

Figure 8.23-21 Suspend all file copy operations involving a repository path

8.23.2.22 TC[23,22] enable the periodic reporting of the file copy status

a. Each telecommand packet transporting a request to enable the periodic reporting of the file copy status shall be of message subtype 22.

NOTE For the corresponding system requirements, refer to clause 6.23.5.5.2.

b. For each telecommand packet transporting a request to enable the periodic reporting of the file copy status, the application data field shall have the structure specified in Figure 8.23-22.

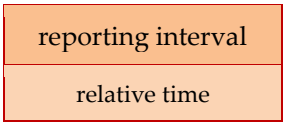

Figure 8.23-22 Enable the periodic reporting of the file copy status

8.23.2.23 TM[23,23] file copy status report

a. Each telemetry packet transporting a file copy status report shall be of message subtype 23.

NOTE For the corresponding system requirements, refer to clause 6.23.5.5.4.

b. For each telemetry packet transporting a file copy status report, the source data field shall have the structure specified in Figure 8.23-23.

repeated N times

N	operation ID	operation status	progress indicator
unsigned integer	unsigned integer	enumerated	unsigned integer

optional

Figure 8.23-23 File copy status report

8.23.2.24 TC[23,24] disable the periodic reporting of the file copy status

a. Each telecommand packet transporting a request to disable the periodic reporting of the file copy status shall be of message subtype 24.

NOTE For the corresponding system requirements, refer to clause 6.23.5.5.3.

b. For each telecommand packet transporting a request to disable the periodic reporting of the file copy status, the application data field shall be omitted.

8.23.2.25 TC[23,25] checksum a file

a. Each telecommand packet transporting a request to checksum a file shall be of message subtype 25.

NOTE For the corresponding system requirements, refer to clause 6.23.5.4.5.

b. For each telecommand packet transporting a request to checksum a file, the application data field shall have the structure specified in Figure 8.23-5.

file path	
repository path	file name
variable-length character-string	variable-length character-string

Figure 8.23-24 Checksum a file

8.23.2.26 TC[23,26] checksum report of a file

a. Each telemetry packet transporting a checksum report of a file shall be of message subtype 26.

NOTE For the corresponding system requirements, refer to clause 6.23.5.4.5.

b. For each telemetry packet transporting a checksum report of a file, the application data field shall have the structure specified in Figure 8.23-5.

file path		checksum
repository path	file name	
variable-length character-string	variable-length character-string	bit-string (16-bits)

Figure 8.23-25 Checksum report of a file

8.23.2.27 TC[23,27] enable directory protection

a. Each telecommand packet transporting a request to enable directory protection shall be of message subtype 27.

NOTE For the corresponding system requirements, refer to clause 6.23.4.6.4.

b. For each telecommand packet transporting a request to enable directory protection, the application data field shall have the structure specified in Figure 8.23-5.

directory path	
repository path	directory name
variable-length character-string	variable-length character-string

Figure 8.23-26 Enable directory protection

8.23.2.28 TC[23,28] disable directory protection

a. Each telecommand packet transporting a request to disable directory protection shall be of message subtype 28.

NOTE For the corresponding system requirements, refer to clause 6.23.4.6.5.

b. For each telecommand packet transporting a request to disable directory protection, the application data field shall have the structure specified in Figure 8.23-5.

directory path	
repository path	directory name
variable-length character-string	variable-length character-string

Figure 8.23-27 Disable directory protection

8.23.3 Enumeration

a. The values of the operation status shall be as specified in Table 8.23-1.

Table 8.23-1 Service 23 operation status

engineering value	raw value
"pending"	0
"in progress"	1

b. The values of the object type shall be as specified in Table 8.23-2.

Table 8.23-2 Service 23 object type

engineering value	raw value
"file"	0
"directory"	1

8.24 ST[24] file transfer

8.24.1 General

a. Each packet transporting a file transfer message shall be of service type 24

b. All implemented CFDP PDUs shall be conveyed by dedicated CCSDS packets as follows:

1. without CCSDS secondary packet header
2. with Packet Error Control field
3. Each CFDP PDU shall be contained in exactly one CCSDS packet
4. the packet conveying CFDP PDUs shall use a unique APID for each independent communication channel

8.24.2 Requests and reports

8.24.2.1 TC[24,1] start of a file transmission opportunity window

a. Each telecommand packet transporting a request to start of a file transmission opportunity window shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.24.4.4

b. For each telecommand packet transporting a request to start of a file transmission opportunity window, the application data field shall have the structure specified in Figure 8.24-1.

local CFDP entity ID	remote CFDP entity ID
enumerated	enumerated

NOTE 1 local CFDP entity refers to the on-board CFDP entity transmitting the file

NOTE 2 remote CFDP entity refers to the CFDP entity (on-ground or on-board) receiving the file

Figure 8.24-1 Start of a file transmission opportunity window

8.24.2.2 TC[24,2] stop of a file transmission opportunity window

a. Each telecommand packet transporting a request to stop of a file transmission opportunity window shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.24.4.5

b. For each telecommand packet transporting a request to stop of a file transmission opportunity window, the application data field shall have the structure specified in Figure 8.24-2.

local CFDP entity ID	remote CFDP entity ID
enumerated	enumerated

NOTE 1 local CFDP entity refers to the on-board CFDP entity transmitting the file

NOTE 2 remote CFDP entity refers to the CFDP entity (on-ground or on-board) receiving the file

Figure 8.24-2 Stop of a file transmission opportunity window

8.24.2.3 TC[24,3] start of a file reception opportunity window

a. Each telecommand packet transporting a request to start of a file reception opportunity window shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.24.4.6

b. For each telecommand packet transporting a request to start of a file reception opportunity window, the application data field shall have the structure specified in Figure 8.24-3.

local CFDP entity ID	remote CFDP entity ID
enumerated	enumerated

NOTE 1 local CFDP entity refers to the on-board CFDP entity receiving the file

NOTE 2 remote CFDP entity refers to the CFDP entity (on-ground or on-board) transmitting the file

Figure 8.24-3 Start of a file reception opportunity window

8.24.2.4 TC[24,4] stop of a file reception opportunity window

a. Each telecommand packet transporting a request to stop of a file reception opportunity window shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.24.4.7

b. For each telecommand packet transporting a request to stop of a file reception opportunity window, the application data field shall have the structure specified in Figure 8.24-4.

local CFDP entity ID	remote CFDP entity ID
enumerated	enumerated

NOTE 1 local CFDP entity refers to the on-board CFDP entity receiving the file

NOTE 2 remote CFDP entity refers to the CFDP entity (on-ground or on-board) transmitting the file

Figure 8.24-4 Stop of a file reception opportunity window

8.24.2.5 TC[24,5] suspend a file transaction

a. Each telecommand packet transporting a request to suspend a file transaction shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.24.4.8

b. For each telecommand packet transporting a request to suspend a file transaction, the application data field shall have the structure specified in Figure 8.24-5.

CFDP entity ID	transaction ID	
	source CFDP entity ID	transaction sequence number
enumerated	enumerated	unsigned integer

NOTE 1 the CFDP entity ID identifies the entity that suspends the file transaction

NOTE 2 the transaction ID is built with the source CFDP entity ID and the transaction sequence number, refer to clause 6.24.4.8

Figure 8.24-5 Suspend a file transaction

8.24.2.6 TC[24,6] resume a file transaction

a. Each telecommand packet transporting a request to resume a file transaction shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.24.4.9

b. For each telecommand packet transporting a request to resume a file transaction, the application data field shall have the structure specified in Figure 8.25-6.

CFDP entity ID	transaction ID	
	source CFDP entity ID	transaction sequence number
enumerated	enumerated	unsigned integer

NOTE 1 the CFDP entity ID identifies the entity that resumes the file transaction

NOTE 2 the transaction ID is built with the source CFDP entity ID and the transaction sequence number, refer to clause 6.24.4.9

Figure 8.24-6 Resume a file transaction

8.24.2.7 TC[24,7] cancel a file transaction

a. Each telecommand packet transporting a request to cancel a file transaction shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.24.4.10

b. For each telecommand packet transporting a request to cancel a file transaction, the application data field shall have the structure specified in Figure 8.24-7.

CFDP entity ID	transaction ID	
	source CFDP entity ID	transaction sequence number
enumerated	enumerated	unsigned integer

NOTE 1 the CFDP entity ID identifies the entity that cancels the file transaction

NOTE 2 the transaction ID is built with the source CFDP entity ID and the transaction sequence number, refer to clause 6.24.4.10

Figure 8.24-7 Cancel a file transaction

8.24.2.8 TC[24,8] report status of file transactions

a. Each telecommand packet transporting a request to report the status of file transactions shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.24.4.11

b. For each telecommand packet transporting a request to report the status of file transactions, the application data field shall have the structure specified in Figure 8.24-8.

CFDP entity ID	transaction direction
enumerated	enumerated

NOTE the transaction direction can be transmission, reception or both

Figure 8.24-8 Report the file transaction status

8.24.2.9 TM[24,9] file transactions status report

a. Each telemetry packet transporting a file transaction status report shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.24.4.11

b. For each telemetry packet transporting a file transaction status report, the source data field shall have the structure specified in Figure 8.24-9.

N
unsigned integer

c. *repeated N times*

transaction ID		destination CFDP entity ID	transaction state
source CFDP entity ID	transaction sequence number		
enumerated	unsigned integer		enumerated

repeated N times

repository path	file name	file size
variable character-string	variable character-string	unsigned integer

repeated N times

transaction progress	NAK counter
unsigned integer	unsigned integer

Figure 8.24-9 File transaction status report

8.24.2.10 TC[24,10] modify remote CFDP entity configuration

a. Each telecommand packet transporting a request to modify a remote CFDP entity configuration shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.24.4.12

b. For each telecommand packet transporting a request to modify a remote CFDP entity configuration, the application data field shall have the structure specified in Figure 8.24-10.

CFDP entity ID	default transmission mode	maximum positive acknowledgement directive interval	maximum negative acknowledgement directive interval	maximum number of positive acknowledgement directives expirations
enumerated	enumerated	unsigned integer	unsigned integer	unsigned integer

maximum number of negative acknowledgement directives expirations	transaction inactivity time limit	transaction check limit	transaction closure requested flag
unsigned integer	unsigned integer	unsigned integer	boolean

optional

Figure 8.24-10 Modify a CFDP entity configuration

8.24.2.11 TC[24,11] report remote CFDP entity configuration

a. Each telecommand packet transporting a request to report a remote CFDP entity configuration shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.24.4.13

b. For each telecommand packet transporting a request to report a remote CFDP entity configuration, the application data field shall have the structure specified in Figure 8.24-11.

CFDP entity ID
enumerated

Figure 8.24-11 Report CFDP entity configuration

8.24.2.12 TM[24,12] remote CFDP entity configuration report

a. Each telemetry packet transporting a remote CFDP entity configuration report shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.24.4.13

For each telemetry packet transporting a remote CFDP entity configuration report, the source data field shall have the structure specified in Figure 8.24-12.

CFDP entity ID	default transmission mode	maximum positive acknowledgement directive interval	maximum negative acknowledgement directive interval
enumerated	enumerated	unsigned integer	unsigned integer

maximum number of positive acknowledgement directives expirations	maximum number of negative acknowledgement directives expirations	transaction inactivity time limit	transaction check limit	transaction closure requested flag
unsigned integer	unsigned integer	unsigned integer	unsigned integer	boolean
			optional	

Figure 8.24-12 remote CFDP entity configuration report

8.24.2.13 TC[24,13] modify downlink manager configuration

a. Each telecommand packet transporting a request to modify the downlink manager configuration shall be of message subtype 13.

NOTE For the corresponding system requirements, refer to clause 6.24.5.4

b. For each telecommand packet transporting a request to modify a downlink manager configuration, the application data field shall have the structure specified in Figure 8.24-13.

Downlink Manager ID	file transaction entity ID	transmission mode (QoS)	maximum number of concurrent file transactions	file age ordering criteria	action after file succesful downlink	backup dir path
enumerated	enumerated	enumerated	unsigned integer	enumerated	enumerated	variable octet-string

Figure 8.24-13 Modify downlink manager configuration

8.24.2.14 TC[24,14] add directories to the downlink manager directory configuration

a. Each telecommand packet transporting a request to add directories to the downlink manager directory configuration shall be of message subtype 14.

NOTE For the corresponding system requirements, refer to clause 6.24.5.5

b. For each telecommand packet transporting a request to add directories to the downlink manager directory configuration, the application data field shall have the structure specified in Figure 8.24-14.

repeated N times

Downlink Manager ID	N	directory configuration		
		path	priority	destination CFDP entity ID
enumerated	unsigned integer	variable octet-string	unsigned integer	enumerated

Figure 8.24-14 Add directories to the downlink manager directory configuration

8.24.2.15 TC[24,15] remove directories from the downlink manager directory configuration

a. Each telecommand packet transporting a request to remove a directory to the downlink manager directory configuration shall be of message subtype 15.

NOTE For the corresponding system requirements, refer to clause 6.24.5.6

b. For each telecommand packet transporting a request to remove directories from the downlink manager directory configuration, the application data field shall have the structure specified in Figure 8.24-15.

repeated N times

Downlink Manager ID	N	directory path
enumerated	unsigned integer	variable octet-string

Figure 8.24-15 Remove directories to the downlink manager directory configuration

8.24.2.16 TC[24,16] report downlink manager configuration

a. Each telecommand packet transporting a request to report a downlink manager configuration shall be of message subtype 16.

NOTE For the corresponding system requirements, refer to clause 6.24.5.7

b. For each telecommand packet transporting a request to report a downlink manager configuration, the application data field shall have the structure specified in Figure 8.24-16.

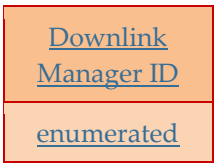

Figure 8.24-16 Report downlink manager directory configuration

8.24.2.17 TM[24,17] downlink manager configuration report

a. Each telemetry packet transporting a downlink manager configuration report shall be of message subtype 17.

NOTE For the corresponding system requirements, refer to clause 6.24.5.7

b. For each telemetry packet transporting a downlink manager configuration report, the source data field shall have the structure specified in Figure 8.24-17.

downlink Manager ID	unbounded files permission flag	file transaction entity ID	file transaction QoS	downlink manager status
enumerated	boolean	enumerated	enumerated	enumerated

maximum number of concurrent file transactions	current number of initiated file transactions	file age ordering criteria	action on succesful file downlinked	back up directory path
unsigned integer	unsigned integer	enumerated	enumerated	variable character-string

repeated N times

N	directory configuration		
	path	priority	destination CFDP entity ID
unsigned integer	variable octet-string	unsigned integer	enumerated

Figure 8.24-17 Downlink manager configuration report

8.24.2.18 TC[24,18] start downlink manager

a. Each telecommand packet transporting a request to start a downlink manager shall be of message subtype 18.

NOTE For the corresponding system requirements, refer to clause 6.24.5.8

b. For each telecommand packet transporting a request to start a downlink manager, the application data field shall have the structure specified in Figure 8.24-18.

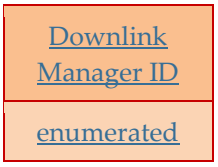

Figure 8.24-18 Start downlink manager

8.24.2.19 TC[24,19] suspend downlink manager

a. Each telecommand packet transporting a request to suspend a downlink manager shall be of message subtype 19.

NOTE For the corresponding system requirements, refer to clause 6.24.5.9

b. For each telecommand packet transporting a request to suspend a downlink manager, the application data field shall have the structure specified in Figure 8.24-19.

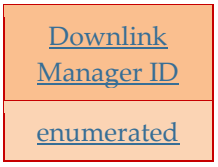

Figure 8.24-19 Suspend downlink manager

8.24.2.20 TC[24,20] stop downlink manager

a. Each telecommand packet transporting a request to start a downlink manager shall be of message subtype 20.

NOTE For the corresponding system requirements, refer to clause 6.24.5.10

b. For each telecommand packet transporting a request to suspend a downlink manager, the application data field shall have the structure specified in Figure 8.24-20.

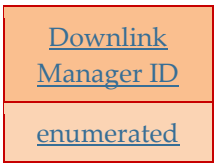

Figure 8.24-20 Stop downlink manager

8.25 ST[25] file data storage

8.25.1 General

a. Each packet transporting a file data storage message shall be of service type 25

8.25.2 Requests and reports

8.25.2.1 TC[25,1] set directory data storage attributes

a. Each telecommand packet transporting a request to set directory data storage attributes shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.25.3.5

b. For each telecommand packet transporting a request to set directory data storage attributes, the application data field shall have the structure specified in Figure 8.25-1.

directory path	max storage in directory	maximum file age
variable-length character-string	unsigned integer	unsigned integer

Figure 8.25-1 Set directory data storage attributes

NOTE maximum file age is the maximum time (in seconds) for the file in the directory after which the file will be automatically deleted. If maximum file age = 0 the deletion is disabled (it is user responsibility to delete it using TC[23,2])

8.25.2.2 TC[25,2] report directory data storage attributes

a. Each telecommand packet transporting a request to report directory data storage attributes shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.25.3.6

b. For each telecommand packet transporting a request to report directory data storage attributes, the application data field shall have the structure specified in Figure 8.25-2.

repeated N times

N	directory path
unsigned integer	variable-length character-string

Figure 8.25-2 Report directory data storage attributes

c. To report data storage attributes of all directories, N shall be set to 0.

8.25.2.3 TM[25,3] directory data storage attributes report

a. Each telemetry packet transporting a directory data storage attributes report shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.25.3.6

b. For each telemetry packet transporting a directory data storage attributes report, the source data field shall have the structure specified in Figure 8.25-3.

repeated N1 times

repeated N2 times

N1	directory path	max storage in directory	free storage in directory	max file age	N2	file name
unsigned integer	variable-length character-string	unsigned integer	unsigned integer	unsigned integer	Unsigned integer	variable-length character-string

Figure 8.25-3 Directory data storage attributes report

8.25.2.4 TC[25,4] add data sources to directory mapping

a. Each telecommand packet transporting a request to add data sources to directory mapping shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.25.3.7

b. For each telecommand packet transporting a request to add data sources to directory mapping, the application data field shall have the structure specified in Figure 8.25-4.

repeated N times

N	data source ID	directory path	file name prefix	max file size for data source	max duration of data write for data source	max storage size for data source
unsigned integer	unsigned integer	variable-length character-string	fixed-length character-string	unsigned integer	unsigned integer	unsigned integer

Figure 8.25-4 Add data sources to directory mapping

8.25.2.5 TC[25,5] remove data sources from directory mapping

a. Each telecommand packet transporting a request to remove data sources from directory mapping shall be of message subtype 5.

NOTE For the corresponding system requirements, refer to clause 6.25.3.8

b. For each telecommand packet transporting a request to remove data sources to directory mapping, the application data field shall have the structure specified in Figure 8.25-5.

repeated N times

N	data source ID
unsigned integer	unsigned integer

Figure 8.25-5 Remove data sources from directory mapping

8.25.2.6 TC[25,6] report data sources mapped to directory

a. Each telecommand packet transporting a request to report data sources to directory mapping configuration shall be of message subtype 6.

NOTE For the corresponding system requirements, refer to clause 6.25.3.9

b. For each telecommand packet transporting a request to report data sources to directory mapping configuration, the application data field shall have the structure specified in Figure 8.25-6.

repeated N times

N	data source ID
unsigned integer	unsigned integer

Figure 8.25-6 Report data sources to directory mapping configuration

c. To report the data source mapping of all directories, N shall be set to 0.

8.25.2.7 TM[25,7] data sources to directory mapping configuration report

a. Each telemetry packet transporting a data sources configuration report shall be of message subtype 7.

NOTE For the corresponding system requirements, refer to clause 6.25.3.9

b. For each telemetry packet transporting a data sources to directory mapping configuration report, the source data field shall have the structure specified in Figure 8.25-7.

repeated N times

N	data source ID	directory path	file name prefix	max file size for data source	max duration of data write for data source	max storage size for data source
unsigned integer	unsigned integer	variable-length character-string	fixed-length character-string	unsigned integer	unsigned integer	unsigned integer

Figure 8.25-7 Data Sources to directory mapping configuration report

8.25.2.8 TC[25,8] report data sources recording status

a. Each telecommand packet transporting a request to report data sources recording status shall be of message subtype 8.

NOTE For the corresponding system requirements, refer to clause 6.25.3.10

b. For each telecommand packet transporting a request to report data sources recording status, the application data field shall have the structure specified in Figure 8.25-8.

repeated N times

N	data source ID
unsigned integer	unsigned integer

Figure 8.25-8 Report data sources recording status

c. To report the recording status of all data sources, N shall be set to 0.

8.25.2.9 TM[25,9] data sources recording status report

a. Each telemetry packet transporting a data sources recording status report shall be of message subtype 9.

NOTE For the corresponding system requirements, refer to clause 6.25.3.10

b. For each telemetry packet transporting a data sources recording status report, the source data field shall have the structure specified in Figure 8.25-9.

repeated N times

N	data source ID	recording status
unsigned integer	unsigned integer	enumerated

NOTE Recording status can be “enabled” or “disabled”

Figure 8.25-9 Data Sources recording status report

8.25.2.10 TC[25,10] enable data sources recording status

a. Each telecommand packet transporting a request to enable data sources recording status shall be of message subtype 10.

NOTE For the corresponding system requirements, refer to clause 6.25.3.11

b. For each telecommand packet transporting a request to enable data sources status, the application data field shall have the structure specified in Figure 8.25-10.

repeated N times

N	data source ID
unsigned integer	unsigned integer

Figure 8.25-10 Enable data sources recording status

c. To enable the recording status of all data sources, N shall be set to 0.

8.25.2.11 TC[25,11] disable data sources recording status

a. Each telecommand packet transporting a request to disable data sources recording status shall be of message subtype 11.

NOTE For the corresponding system requirements, refer to clause 6.25.3.12

b. For each telecommand packet transporting a request to disable data sources status, the application data field shall have the structure specified in Figure 8.25-11.

repeated N times

N	data source ID
unsigned integer	unsigned integer

Figure 8.25-11 Disable data sources recording status

c. To disable the recording status of all data sources, N shall be set to 0.

8.25.2.12 TC[25,12] close file currently used for data storage

a. Each telecommand packet transporting a request to close the file currently used for data storage shall be of message subtype 12.

NOTE For the corresponding system requirements, refer to clause 6.25.3.13

b. For each telecommand packet transporting a request to close the file currently used for data storage, the application data field shall have the structure specified in Figure 8.25-12.

repeated N times

N	data source ID
unsigned integer	unsigned integer

Figure 8.25-12 Close current file used for data storage

c. To close of all files used for data storage, N shall be set to 0.

8.25.2.13 TC[25,13] add report-types to the application process PUS report-type data source control configuration

a. Each telecommand packet transporting a request to add report-types to the application process PUS report-type data source control configuration shall be of message subtype 13.

b. For each telecommand packet transporting a request to add a report-types to the application process PUS report-type data source control configuration , the application data field shall have the structure specified in Figure 8.25-13.

NOTE For the corresponding system requirements, refer to clause 6.25.4.4.1

repeated N1 times

repeated N2 times

repeated N3 times

data source ID	N1	application process ID	N2	service type	N3	message subtype
unsigned integer	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.25-13 Add report-types to the application process PUS report-type data source control configuration

c. To add all report types of an application process to the application process PUS report-type data source control configuration, N2 shall be set to 0.

d. To add all report types of a service type to the application process PUS report-type data source control configuration, N3 shall be set to 0.

8.25.2.14 TC[25,14] delete report-types from the application process PUS report-type data source control configuration

a. Each telecommand packet transporting a request to delete report-types from the application process PUS report-type data source control configuration shall be of message subtype 14.

b. For each telecommand packet transporting a request to delete report-types from the application process PUS report-type data source control configuration , the application data field shall have the structure specified in Figure 8.25-14.

NOTE For the corresponding system requirements, refer to clause 6.25.4.4.2

repeated N1 times

repeated N2 times

repeated N3 times

data source ID	N1	application process ID	N2	service type	N3	message subtype
unsigned integer	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.25-14 Delete report-types from the application process PUS report-type data source control configuration

c. To empty the entire application process PUS report-type data source control configuration, N1 shall be set to 0.

d. To delete an application process from the application process PUS report-type data source control configuration, N2 shall be set to 0.

e. To delete a service type from the application process PUS report-type data source control configuration, N3 shall be set to 0.

8.25.2.15 TC[25,15] report the content of the application process PUS report-type data source control configuration

a. Each telecommand packet transporting a request to report the content of the application process PUS report-type data sources control configuration shall be of message subtype 15.

b. For each telecommand packet transporting a request to report the content of the application process PUS report-type data source control configuration, the application data field shall have the structure specified in Figure 8.25-15.

NOTE For the corresponding system requirements, refer to clause 6.25.4.4.3

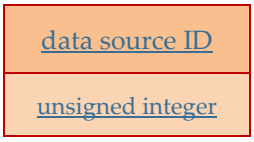

Figure 8.25-15 Report the content of the application process PUS report-type data source control configuration

8.25.2.16 TM[25,16] application process PUS report-type data source control configuration content report

a. Each telemetry packet transporting a report-types data sources report shall be of message subtype 16.

b. For each telemetry packet transporting a report-types data sources definition report, the source data field shall have the structure specified in Figure 8.25-16.

NOTE For the corresponding system requirements, refer to clause 6.25.4.4.3

repeated N1 times

repeated N2 times

repeated N3 times

data source ID	N1	application process ID	N2	service type	N3	message subtype
unsigned integer	unsigned integer	enumerated	unsigned integer	enumerated	unsigned integer	enumerated

Figure 8.25-16 Application process PUS report-type data source control configuration content report

8.26 ST[26] parameter extraction

8.26.1 General

a. Each packet transporting a parameter extraction message shall be of service type 26.

8.26.2 Requests and reports

8.26.2.1 TC[26,1] add parameter extraction definitions

a. Each telecommand packet transporting a request to add parameter extraction definitions shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.26.4.1

b. For each telecommand packet transporting a request to add parameter extraction definitions, the application data field shall have the structure specified in Figure 8.26-1.

repeated N times

N	parameter ID	application process ID	service type	message subtype	housekeeping parameter report structure ID / event definition ID	offset
unsigned integer (2 bytes)	enumerated (4 bytes)	enumerated (2 bytes)	enumerated (1 byte)	enumerated (1 byte)	enumerated (4 bytes)	unsigned integer (2 bytes)

Figure 8.26-1. Add parameter extraction definitions

NOTE 1 Offset is defined as number of bytes from the beginning of the telemetry packet data field.

NOTE 2 The interface reserves 4 bytes indistinctly for SID or EID, even SID length is always 2 bytes. The implementation uses only the 2 Less Significant Bytes when dealing with SIDs.

8.26.2.2 TC[26,2] delete parameter extraction definitions

a. Each telecommand packet transporting a request to delete parameter extraction definitions shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.26.4.2.

b. For each telecommand packet transporting a request to delete parameter extraction definitions, the application data field shall have the structure specified in Figure 8.26-2.

repeated N times

N	parameter ID
unsigned integer (2 bytes)	enumerated (4 bytes)

Figure 8.26-2. Delete parameter extraction definitions

c. To delete all parameter extraction definitions N shall be set to 0.

8.26.2.3 TC[26,3] report parameter extraction definitions

a. Each telecommand packet transporting a request to report the parameter extraction definitions shall be of message subtype 3.

NOTE For the corresponding system requirements, refer to clause 6.26.4.3.

b. For each telecommand packet transporting a request to report the parameter extraction definitions, the application data field shall be omitted.

8.26.2.4 TM[26,4] parameter extraction definitions report

a. Each telemetry packet transporting a parameter extraction definitions report shall be of message subtype 4.

NOTE For the corresponding system requirements, refer to clause 6.26.4.3.

b. For each telemetry packet transporting a parameter extraction definitions report, the source data field shall have the structure specified in Figure 8.26-3.

repeated N times

N	parameter ID	application process ID	service type	message subtype	housekeeping parameter report structure ID / event definition ID	Offset
unsigned integer (2 bytes)	enumerated (4 bytes)	enumerated (2 bytes)	enumerated (1 byte)	enumerated (1 byte)	enumerated (4 bytes)	unsigned integer (2 bytes)

Figure 8.26-3. Report the parameter extraction definitions

NOTE Offset is defined as number of bytes from the beginning of the telemetry packet data field.

8.27 ST[27] critical packet log management

8.27.1 General

a. Each packet transporting a critical packet log management message shall be of service type 27.

8.27.2 Requests and reports

8.27.2.1 TC[27,1] downlink critical packet log

a. Each telecommand packet transporting a request to downlink the critical packet log shall be of message subtype 1.

NOTE For the corresponding system requirements, refer to clause 6.27.4.3.2.

b. For each telecommand packet transporting a request to downlink the critical packet log, the application data field shall be omitted.

8.27.2.2 TC[27,2] clear downlinked packets from critical packet log

a. Each telecommand packet transporting a request to clear downlinked packets from the critical packet log shall be of message subtype 2.

NOTE For the corresponding system requirements, refer to clause 6.27.4.4.2.

b. For each telecommand packet transporting a request to clear downlinked packets from the critical packet log, the application data field shall be omitted.

9
Command Pulse Distribution Unit

9.1 Scope

A CPDU is a simple on-board unit designed to provide ground with direct access to equipment. For example, such direct access is used during contingency to reset an S-band transponder or a sensor.

Each CPDU is logically handled as an on-board application process, i.e. there is an application process identifier that represents that CPDU exclusively.

Each CPDU can be:

- directly accessed from the ground by addressing:
  - a virtual channel that logically links the ground to one or more multiplexer access points (MAPs), and
  - a multiplexer access point that is physically linked to that CPDU;
- indirectly accessed by use of an on-board application process that hosts a device access subservice, refer to the request to distribute CPDU commands specified in clause 6.2.6.2

Each CPDU has a number of addressable outputs. A subset of these addressable outputs are equipped with output lines that can be physically connected to an equipment.

Commanding a CPDU consists of issuing requests that contain CPDU command pulse instructions, each one identifying the CPDU addressable output and specifying the duration of the pulse to generate.

9.2 System requirements

9.2.1 CPDU

a. For each CPDU, the pulse duration unit used by that CPDU shall be declared when specifying that CPDU.

b. Each pulse duration unit shall be greater than or equal to 10 ms, and less than or equal to 15 ms.

c. The number of addressable outputs exposed by each CPDU shall be declared when specifying that CPDU.

NOTE This Standard supports CPDUs that expose up to 2^{12} addressable outputs. The CPDU suppliers can

equip a subset of the addressable outputs with output lines. These equipped addressable outputs are available for being physically connected.

d. Each CPDU addressable output shall be uniquely identified by an enumerated value represented by an unsigned integer that is greater than or equal to 0, and less than 2^{12}.

e. The list of CPDU addressable outputs that are equipped with output lines shall be declared when specifying that CPDU.

NOTE These outputs are named "CPDU equipped addressable outputs".

f. For each CPDU, the maximum number of command pulse instructions contained within a CPDU request shall be declared when specifying that CPDU.

NOTE The maximum number of command pulse instructions is constrained by the size of the TC segment, refer to ECSS-E-ST-50-04.

g. For each CPDU, the maximum number of command pulse instructions contained within a CPDU request that is at least 12 and at most 504 shall be declared when specifying that CPDU.

NOTE This maximum number of command pulse instructions determines the maximum size of the telecommand packet used to transport the related CPDU request. That maximum telecommand packet size is constrained by the maximum telecommand segment size, refer to ECSS-E-ST-50-04.

9.2.2 Accessibility

a. The list of CPDUs available on-board shall be declared when specifying the spacecraft architecture.

b. For each CPDU, the application process identifier used to refer to that CPDU shall be declared when specifying the spacecraft architecture.

c. For each CPDU, the list of multiplexer access points physically linked to that CPDU shall be declared when specifying the spacecraft architecture.

NOTE 1 The multiplexer access point identifier that equals to 0 is usually associated to a CPDU connected to a TC decoder without cross-coupling.

NOTE 2 See also clause 7.1.2.3.

d. For each CPDU and associated multiplexer access point, the virtual channel that is used to carry the associated TC segments shall be declared when specifying the spacecraft architecture.

NOTE 1 For TC segments, see ECSS-E-ST-50-04.

NOTE 2 The telecommand link to a CPDU is uniquely identified by the combination of the virtual channel identifier and the multiplexed access point identifier.

e. Each CPDU equipped addressable output that is physically connected shall be declared when specifying the spacecraft architecture.

NOTE These outputs are named "CPDU physically connected outputs".

f. For each CPDU physically connected output, the minimum pulse duration and the maximum pulse duration supported by that output shall be declared when specifying the spacecraft architecture.

NOTE These minimum and maximum pulse durations are constrained by the characteristic of the equipment that is physically connected.

9.2.3 CPDU request

a. Each CPDU request shall contain one or more command pulse instructions.

b. Each command pulse instruction shall contain:

14. the identifier of a CPDU physically connected output;

15. the duration exponential value used to calculate the duration of the command pulse to emit on that output.

NOTE 1 For item 1, refer to requirements in clause 9.2.1.

NOTE 2 For item 2, the pulse duration unit is specified in requirement 9.2.1a.

c. The duration exponential value in a command pulse instruction shall be an unsigned integer greater than or equal to 0, and less than or equal to 7.

NOTE When the CPDU executes a command pulse instruction, it generates a pulse on the specified output line of a duration equal to:

$$\text{Pulse duration unit of that CPDU} \times 2^{\text{duration exponential value}}$$

9.3 Interface requirements

9.3.1 CPDU request

a. Each telecommand packet transporting a CPDU request shall be a CCSDS space packet that contains:

1. a packet primary header with:
   (a) a packet version number set to 0,
   (b) a packet type set to 1,
   (c) a secondary header flag set to 0,
   (d) the application process identifier of the CPDU addressed by that request,
   (e) the 2 bits of the sequence flags set to "11",
   (f) the packet data length of the telecommand packet;

NOTE The possible values for the packet sequence counter are mission-specific

2. a packet data field with:
   (a) no packet secondary header,
   (b) an application data field,
   (c) no spare field,
   (d) a packet error control field that is a 16-bit CRC identical to the one used in the frame error control field of the telecommand protocol of the space data link.

   NOTE 1 The structure of the CCSDS space packet is described in clause 7.3.14.

   NOTE 2 For item 2(d), for the frame error control field of the telecommand protocol of the space data link, refer to ECSS-E-ST-50-04.

b. For each telecommand packet transporting a CPDU request, the application data field shall have the structure specified in .

repeated n times
with $1 \leq N \leq CPDU\ maximum\ number\ of\ instructions$

output line ID	reserved	duration exponential value
enumerated (12 bits)	bit-string (1 bit)	enumerated (3 bits)

NOTE The CPDU maximum number of instructions is defined in requirement 9.2.1g.

Figure 9-1 CPDU request

Annex A(informative) IEEE and MIL-STD real formats

A.1 IEEE standard format

A.1.1 General

The important features of the IEEE standard simple precision and double precision formats (refer to "IEEE 754 Standard for binary floating-point arithmetic" (Reference [7]) are provided below.

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

$(-1)^S \times 2^E \times (b_0 \cdot b_1 b_2 \ldots b_{p-1})$

where:

- $b_0 \cdot b_1 b_2 \ldots . b_{p-1}$ means $\frac{b_0}{2^0} + \frac{b_1}{2^1} + \frac{b_2}{2^2} + \ldots + \frac{b_{p-1}}{2^{p-1}}$
- 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. For both formats, the encoding of the real number values use 3 fields as follows:

- the sign field, on 1 bit, that states whether:
  - the value is positive, i.e. sign = 0, or
  - the value is negative, i.e. sign = 1;
- the exponent field:
  - on 8 bits for single-precision real values, or
  - on 11 bits for double-precision real values
- the fraction field, i.e. a bit-string containing the value $\cdot b_1 b_2 \ldots b_{p-1}$ with:
  - $p = 24$ for single-precision real values, or
  - $p = 53$ for double-precision real values.

A.1.2 Single-precision

The encoded value of a single-precision real parameter has the structure defined in Figure A-1 .

sign	exponent	fraction
1 bit	8 bits	23 bits

Figure A-1 **Single-precision real encoded value structure**

The encoded value structure of a single-precision real parameter provides the capability to represent the values reported in Table A-1 .

Table A-1 Single-precision real parameter encoded values

	value
if exponent = 255 and fraction <> 0	not a number
if exponent = 255 and fraction = 0	$(-1)^{sign} \times \infty$
if 0 < exponent < 255	$(-1)^{sign} \times 2^{exponent-127} \times (1, fraction)$
if exponent = 0 and fraction <> 0	$(-1)^{sign} \times 2^{-126} \times (0, fraction)$
if exponent = 0 and fraction = 0	0

In the cases where $Exponent = 0$ and $Fraction <> 0$, the values are said to be denormalized.

The range of possible values and precision for a simple-precision real parameter are as follows:

$$1{,}12 \times 10^{-38} \leq |value| \leq 3{,}40 \times 10^{38} (precision\ 1{,}15 \times 10^{-7})$$

A.1.3 Double-precision

The encoded value of a double-precision real parameter has the structure defined in Figure A-2 .

sign	exponent	fraction
1 bit	11 bits	52 bits

Figure A-2 **Double-precision real parameter encoded value structure**

The encoded value structure of a double-precision real parameter provides the capability to represent the values reported in Table A-2 .

Table A-2 Double-precision real parameter encoded values

	value
if exponent = 2 047 and fraction <> 0	not a number
if exponent = 2 047 and fraction = 0	$(-1)^{sign} \times \infty$
if 0 < exponent < 2 047	$(-1)^{sign} \times 2^{exponent-1023} \times (1, fraction)$
if exponent = 0 and fraction <> 0	$(-1)^{sign} \times 2^{-1022} \times (0, fraction)$
if exponent = 0 and fraction = 0	0

In the cases where $Exponent = 0$ and $Fraction <> 0$, the values are said to be denormalized.

The range of possible values and precision for a double-precision real parameter are as follows:

$2{,}22 \times 10^{-308} \leq |value| \leq 1{,}79 \times 10^{308} (precision\ 2{,}22 \times 10^{-16})$

A.2 United States Air Force military standard format

A.2.1 General

The important features of the United States Air Force military standard single-precision floating-point data and extended-precision floating-point data formats (refer to "Military Standard Sixteen-Bit Computer Instruction Set Architecture" MIL-STD-1750a, 2nd July 1980 (Reference [8]) are provided below.

Floating-point numbers are represented as a fractional mantissa times 2 raised to the power of the exponent. All floating-point numbers are assumed normalized or floating-point zero at the beginning of a floating-point operation and the results of all floating-point operations are normalized (a normalized floating-point number has the sign of the mantissa and the next bit of opposite value) or floating-point zero. A floating-point zero is defined as $0000\ 0000_{16}$, that is, a zero mantissa and a zero exponent (00_{16}). An extended floating-point zero is defined as $0000\ 0000\ 0000_{16}$, that is, a zero mantissa and a zero exponent.

For both floating-point and extended floating-point numbers, an overflow is defined as an exponent overflow and an underflow is defined as an exponent underflow.

A.2.2 Simple-precision

As shown in Figure A-3 , simple-precision floating-point data are represented as a 32-bit quantity consisting of a 24-bit 2's complement mantissa and an 8-bit 2/s complement exponent.

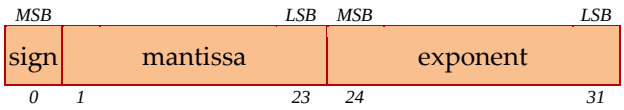

Figure A-3 **Single-precision floating-point data structure**

Some examples of the machine representation for 32-bit floating-point numbers are provided in Table A-3 .

Table A-3 Some examples of 32-bit floating-point numbers

decimal number	hexadecimal notation
$0{,}999\,9998 \times 2^{127}$	7FFF FFFF
$0{,}5 \times 2^{127}$	4000 007F
$0{,}625 \times 2^{4}$	5000 0004
$0{,}5 \times 2^{1}$	4000 0001
$0{,}5 \times 2^{0}$	4000 0000
$0{,}5 \times 2^{-1}$	4000 00FF
$0{,}5 \times 2^{-128}$	4000 0080
$0{,}0 \times 2^{0}$	0000 0000
$-1{,}0 \times 2^{0}$	8000 0000

decimal number	hexadecimal notation
$-0{,}500\,000\,1 \times 2^{-128}$	BFFF FF80
$-0{,}750\,000\,1 \times 2^{4}$	9FFF FF04

A.2.3 Extended

As shown in Figure A-4 , extended floating-point data are represented as a 48-bit quantity consisting of a 40-bit 2's complement mantissa and an 8-bit 2's complement exponent. The exponent bits 24 to 31 lie between the split mantissa bits 0 to 23 and bits 32 to 47. The most significant bit of the mantissa is the sign bit 0, and the least significant bit of the mantissa is bit 47.

Figure A-4 **extended floating-point data structure**

Some examples of the machine representation of 48-bit extended floating-point numbers are provided in Table A-4 .

Table A-4 Some examples of 48-bit extended floating-point numbers

Decimal Number	Mantissa (MSB)	Exp	Mantissa (LSB)
$0{,}5 \times 2^{127}$	400000	7F	0000
$0{,}5 \times 2^{0}$	400000	00	0000
$0{,}5 \times 2^{-1}$	400000	FF	0000
$0{,}5 \times 2^{-128}$	400000	80	0000
$-1{,}5 \times 2^{127}$	800000	7F	0000
$-1{,}0 \times 2^{0}$	800000	00	0000
$-1{,}0 \times 2^{-1}$	800000	FF	0000
$-1{,}0 \times 2^{-128}$	800000	80	0000
$0{,}0 \times 2^{0}$	000000	00	0000
$-0{,}75 \times 2^{-1}$	A00000	FF	0000

Annex B
CRC and ISO checksum

B.1 The cyclic redundancy code (CRC)

B.1.1 General

The packet error control field provides the capability for detecting data corruption introduced into a telemetry packet or a telecommand packet by the lower layers during the transmission, intermediate processing or storage of the packet. The Cyclic Redundancy Code (CRC), also known as the cyclic redundancy check, is an error detecting algorithm that uses the polynomial division to determine the value of the packet error control field.

The encoding/decoding procedure, which is described in detail in the following clauses, produces a 16-bit Packet Check Sequence (PCS) that is placed in the packet error control field. The algorithm used is also known under the name CRC-16-CCITT (See ITU-T V.41). The basic idea behind the CRC-16-CCITT is to treat the entire data packet proper as a binary number, which both the sender and receiver divide using the same divisor. The quotient is discarded. The remainder forms the 16-bit PCS that is placed in the packet error control field. The CRC-16-CCITT uses the following generator polynomial (G):

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

where the + represents the module 2 addition operator. That is, the polynomial expression is manipulated using modulo 2.

In the algorithm used, both encoder and decoder are initialized to the "all-ones" state for each packet.

The PCS generation is performed over the data that covers the entire packet including the packet header but excluding the packet error control field.

The error detection properties of the CRC can be expressed as follows:

The proportion of all errors in the data that are not detected is approximately 1,53 × 10-5.

An error in the data affecting an odd number of bits is always detected.

An error in the data affecting exactly two bits, no more than 65 535 bits apart, is always detected.

If an error in the data affects an even number of bits (greater than or equal to 4), the probability that the error is not detected is approximately 3 × 10-5 for a data length of 4 096 octets. The probability increases slightly for larger data lengths and decreases slightly for smaller data lengths.

A single error burst spanning 16 bits or less of the data is always detected. Not all intermediate bits in the error burst span need be affected.

This code is intended only for error detection purposes and no attempt should be made to utilize it for correction.

B.1.2 Symbols and conventions

The symbols and conventions defined in Table B-1 are used.

Table B-1 CRC symbols and conventions

symbol	meaning
n	The number of bits in the data packet proper.
M(x)	The (n-16)-bit message to be encoded, expressed as a polynomial with binary coefficients.
L(x)	The pre-setting polynomial. This pre-setting polynomial is given by: $L(x) = \sum_{i=0}^{15} x^i$
G(x)	The generating polynomial given by: $G(x) = x^{16} + x^{12} + x^5 + 1$
+	The modulo 2 addition operator (exclusive-or)
$C^*(x)$	The received block in polynomial form.
S(x)	The syndrome polynomial, which is zero if no error has been detected.

B.1.3 Encoding procedure

The encoding procedure accepts the (n-16)-bits message and generates a 16-bit-Packet Check Sequence (PCS) as follows:

$$PCS = (x^{16} \times M(x) + x^{n-16} \times L(x)) \; modulo \; G(x)$$

The encoding procedure differs from that of a conventional cyclic block encoding operation in that the $x^{n-16} \times L(x)$term has the effect of pre-setting the shift register to an "all ones" state (rather than a conventional all zeros state) prior to encoding.

B.1.4 Decoding procedure

The error detection syndrome, S(x) is given by:

$$S(x) = \left[x^{16} \times C^*(x) + x^n \times L(x)\right) \; modulo \; G(x)$$

If S(x) = 0 then no error is detected.

B.1.5 Verification of compliance

The binary sequences defined in Table B-2 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.

Table B-2 Verification of CRC compliance

data	CRC
00 00	1D 0F
00 00 00	CC 9C
AB CD EF 01	04 A2
14 56 F8 9A 00 01	7F D5

B.1.6 Software implementation

CRC codes are particularly efficient when it comes to hardware implementation. Software implementation, on the other hand, is very complex. Two CRC calculation examples are implemented in the algorithm below, i.e.:

- a non-optimized calculation, the CRC function that calculates the CRC for one byte in serial fashion and returns the value of the calculated CRC checksum.
- an optimized function (approximately ten times faster than the non-optimised CRC function), the Crc_opt function that generates the CRC for one byte and returns the value of the new syndrome.

```
#include <stdio.h>
#include <stdint.h>
#define ERROR_DETECTED 0
#define NO_ERROR_DETECTED 1
/* Look-up table, only required for optimized CRC version */
uint16_t LTbl[256];
/* Unoptimized CRC version */
/* One step unoptimized CRC */
uint16_t Crc(Data, Syndrome)
  uint8_t Data;       /* Byte to be encoded */
  uint16_t Syndrome;  /* Original CRC syndrome */
{
  uint8_t icrc;            /* Loop index */
  for (icrc = 0; icrc < 8; icrc++) {
    if ((Data & 0x80) ^ ((Syndrome & 0x8000) >> 8)) {
      Syndrome = ((Syndrome << 1) ^ 0x1021) & 0xFFFF;
    } else {
      Syndrome = (Syndrome << 1) & 0xFFFF;
    }
    Data = Data << 1;
  }
  return (Syndrome);
}
/* Encoding procedure */
/* NOTE: Assumption is that enough memory has been allocated for byte */
/* stream B to allow for generation of the checksum value. */
```

```
/* The two checksum octets are placed in the destination field */
/* (as Nth and Nth + 1 octet of byte stream B). */
/* The destination field is also known as the packet error  */
/* control field. */
void crc_encode(B, octets)
  uint8_t* B;              /* Buffer */
  uint32_t octets;         /* Size of the buffer */
{
  uint32_t index;          /* Loop index */
  uint32_t Chk;            /* CRC syndrome */
  Chk = 0xFFFF;            /* Reset syndrome to all ones */
  for (index = 0; index < octets; index++)
    Chk = Crc (B[index], Chk);    /* Unoptimized CRC */
  B[octets + 1] = Chk & 0xff;
  B[octets] = (Chk >> 8) & 0xff;
}
/* Optimized CRC version */
/* Look-up table initialization */
void InitLtbl(table)
  uint16_t table[];         /* Table to initialise */
{
  uint16_t itable;          /* Loop index */
  uint16_t tmp;             /* Temporary value */
  for (itable = 0; itable < 256; itable++) {
    tmp = 0;
    if ((itable &   1) != 0) tmp = tmp ^ 0x1021;
    if ((itable &   2) != 0) tmp = tmp ^ 0x2042;
    if ((itable &   4) != 0) tmp = tmp ^ 0x4084;
    if ((itable &   8) != 0) tmp = tmp ^ 0x8108;
    if ((itable &  16) != 0) tmp = tmp ^ 0x1231;
    if ((itable &  32) != 0) tmp = tmp ^ 0x2462;
    if ((itable &  64) != 0) tmp = tmp ^ 0x48C4;
    if ((itable & 128) != 0) tmp = tmp ^ 0x9188;
    table[itable] = tmp;
  }
}
/* One step optimized CRC */
uint16_t Crc_opt(D, Chk, table)
  uint8_t      D;           /* Byte to be encoded */
  uint16_t     Chk;         /* Syndrome */
  uint16_t     table[];     /* Look-up table */
{
  return (((Chk << 8) & 0xFF00) ^ table[(((Chk >> 8) ^ D) & 0x00FF)]);
}
/* Encoding optimized procedure */
/* NOTE: Assumption is that enough memory has been allocated for byte */
/* stream B to allow for generation of the checksum value. */
/* The two checksum octets are placed in the destination field */
/* (as Nth and Nth + 1 octet of byte stream B). */
/* The destination field is also known as the packet error  */
/* control field. */
void crc_encode_opt(B, octets)
  uint8_t*     B;           /* Buffer */
  uint32_t     octets;      /* Size of the buffer */
{
  uint32_t index;           /* Loop index */
  uint32_t Chk;             /* CRC syndrome */
  Chk = 0xFFFF;             /* Reset syndrome to all ones */
  for (index = 0; index < octets; index++)
  {
    Chk = Crc_opt (B[index], Chk, LTbl);    /* Optimized CRC */
  }
```

```
  B[octets + 1] = Chk & 0xff;
  B[octets] = (Chk >> 8) & 0xff;
}
/* Decoding function using unoptimized CRC version */
uint8_t crc_decode(B, octets)
  uint8_t*     B;          /* Buffer to be checked */
  uint32_t     octets;     /* Length of the buffer incldуing the crc */
{
  /* Decoding procedure */
  /* The error detection syndrome, S(x) is given by: */
  /* S(x)=(x^16 * C¤(x) + x^n * L(x)) modulo G(x) */
  /* If S(x) = 0 then no error is detected. */

  uint32_t index;          /* Loop index */
  uint8_t  result;         /* Result of the decoding */
  uint16_t Chk;            /* CRC syndrome */
  Chk = 0xFFFF;            /* Reset syndrome to all ones */
  for (index = 0; index < octets; index++)   {
    Chk = Crc (B[index], Chk);    /* Unoptimized CRC */
  }
  if (Chk == 0)
      result = NO_ERROR_DETECTED;
  else
      result = ERROR_DETECTED;
  return result;
}
/* Print a buffer in hexadecimal format */
static void print_buffer(B, octets, method)
  uint8_t*     B;          /* Buffer to display */
  uint32_t     octets;     /* Length of the buffer in bytes */
  char*        method;     /* Method's string */
{
  uint32_t index;          /* Loop index */
  printf ("%sCRC - Data field with calculated CRC checksum is: ", method);
  for (index = 0; index < octets; index++)
    printf ("%02X ", B[index]);
}
/* Display the message related to the result of a decoding of the buffer */
static void print_status(result)
  uint8_t result;          /* Result, should be ERROR_DETECTED or NO_ERROR_DETECTED
*/
{
  if (result == ERROR_DETECTED)
    printf(" - Error-Detected decoding checksum\n");
  else
    printf(" - No-Error-Detected decoding checksum\n");
}
/* Simple program to test both CRC generating functions */
int main(void)
{
  uint32_t N;              /* Size of the buffer - only the data part */
  uint8_t status;          /* Status of the decoding */
  /* Declaration of test data (note that two extra octets are declared */
  /* for each data sequence to reserve room for the two checksum octets) */
  uint8_t VData1[] = {0x00, 0x00, 0x00, 0x00};
  uint8_t VData2[] = {0x00, 0x00, 0x00, 0x00, 0x00};
  uint8_t VData3[] = {0xab, 0xcd, 0xef, 0x01, 0x00, 0x00};
  uint8_t VData4[] = {0x14, 0x56, 0xf8, 0x9a, 0x00, 0x01, 0x00, 0x00};
  /* Initiate look-up table */
  InitLtbl (LTbl);
  /* Encode VData1 unoptimized version */
  N = 2;
```

```
crc_encode(VData1, N);
/* The last 2 octets of VData1 now contain the crc */
print_buffer(VData1, N + 2, "Unoptimized ");
/* Decode VData1 */
status = crc_decode(VData1, N + 2);
print_status(status);
/* Encode VData1 optimized version */
N = 2;
crc_encode_opt(VData1, N);
/* The last 2 octets of VData1 now contain the crc */
print_buffer(VData1, N + 2, "  Optimized ");
/* Decode VData1 */
status = crc_decode(VData1, N + 2);
print_status(status);
/* Encode VData2 unoptimized version */
N = 3;
crc_encode(VData2, N);
/* The last 2 octets of VData2 now contain the crc */
print_buffer(VData2, N + 2, "Unoptimized ");
/* Decode VData2 */
status = crc_decode(VData2, N + 2);
print_status(status);
/* Encode VData2 optimized version */
N = 3;
crc_encode_opt(VData2, N);
/* The last 2 octets of VData2 now contain the crc */
print_buffer(VData2, N + 2, "  Optimized ");
/* Decode VData2 */
status = crc_decode(VData2, N + 2);
print_status(status);
/* Encode VData3 unoptimized version */
N = 4;
crc_encode(VData3, N);
/* The last 2 octets of VData3 now contain the crc */
print_buffer(VData3, N + 2, "Unoptimized ");
/* Decode VData3 */
status = crc_decode(VData3, N + 2);
print_status(status);
/* Encode VData3 optimized version */
N = 4;
crc_encode_opt(VData3, N);
/* The last 2 octets of VData3 now contain the crc */
print_buffer(VData3, N + 2, "  Optimized ");
/* Decode VData3 */
status = crc_decode(VData3, N + 2);
print_status(status);
/* Encode VData4 unoptimized version */
N = 6;
crc_encode(VData4, N);
/* The last 2 octets of VData4 now contain the crc */
print_buffer(VData4, N + 2, "Unoptimized ");
/* Decode VData4 */
status = crc_decode(VData4, N + 2);
print_status(status);
/* Encode VData4 optimized version */
N = 6;
crc_encode_opt(VData4, N);
/* The last 2 octets of VData4 now contain the crc */
print_buffer(VData4, N + 2, "  Optimized ");
/* Decode VData4 */
status = crc_decode(VData4, N + 2);
print_status(status);
```

```
  return 0;
}
/* This program results in the following output:
Unoptimized CRC - Data field with calculated CRC checksum is: 00 00 1D 0F  - No-
Error-Detected decoding checksum
  Optimized CRC - Data field with calculated CRC checksum is: 00 00 1D 0F  - No-
Error-Detected decoding checksum
Unoptimized CRC - Data field with calculated CRC checksum is: 00 00 00 CC 9C  - No-
Error-Detected decoding checksum
  Optimized CRC - Data field with calculated CRC checksum is: 00 00 00 CC 9C  - No-
Error-Detected decoding checksum
Unoptimized CRC - Data field with calculated CRC checksum is: AB CD EF 01 04 A2  -
No-Error-Detected decoding checksum
  Optimized CRC - Data field with calculated CRC checksum is: AB CD EF 01 04 A2  -
No-Error-Detected decoding checksum
Unoptimized CRC - Data field with calculated CRC checksum is: 14 56 F8 9A 00 01 7F
D5  - No-Error-Detected decoding checksum
  Optimized CRC - Data field with calculated CRC checksum is: 14 56 F8 9A 00 01 7F
D5  - No-Error-Detected decoding checksum
*/
```

B.2 The ISO checksum

B.2.1 General

The ISO checksum is an error-detecting algorithm that uses integer arithmetic to determine the value of the packet error control field.

The encoding/decoding procedure, which is described in detail in the following clauses, produces a 16-bit packet checksum (2 octets) that is placed in the packet error field. The ISO checksum algorithm (See ISO 8473-1:1998) uses two main computations, one based on the value of the data octets in the data packet and the other is a weighted value of the data octets, whereby the weight is determined by the position of the octet in the data packet proper. The combination of both octets provides the 16-bit packet checksum.

The ISO checksum procedure can be easily implemented in software on processors using a compact and efficient algorithm. In contrast to the CRC algorithm (see clause B.1), it does not require a look-up table and it does not perform bitwise operations on the data to be checked.

This Standard specifies that the ISO checksum procedure can be used to check the contents of an on-board memory area using the services of the memory management service (see clause 6.6). All octets of the on-board 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 have been introduced into a telemetry packet or a telecommand packet) during the transmission, intermediate processing or storage of the packet. 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 equal to those of the CRC. The error detection properties can be expressed as follows:

The proportion of all errors in the data that are not detected is approximately $1{,}54 \times 10^{-5}$, i.e. the checksum detects virtually the same proportion of all errors as does the CRC.

A single bit in error is always detected.

In contrast to the CRC, an error in the data that 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 that affects an even number of bits.

An error in the data affecting exactly two bits, no more than 2 040 bits apart, is always detected.

The probability that a single error burst spanning 16 bits or less of the data is not detected is approximately $1{,}9 \times 10^{-7}$. Not all intermediate bits in the error burst span need be affected.

This probability is non-zero because the algorithm does not detect an error burst which causes 8 consecutive bits to change from all zeros to all ones or vice-versa.

This code is intended only for error detection purposes and no attempt should be made to utilize it for correction.

B.2.2 Symbols and conventions

The symbols and conventions defined in Table B-3 are used.

Table B-3 ISO symbols and conventions

symbol	meaning
C_0, C_1	Variables used in the encoding and decoding procedures. C0 represents the calculation based on the value of the octets, C1 represents the weighted calculations.
Bi	The integer value of the i^{th} octet to be checked.
N	The number of octets of data to be checked.
CK_1	The value of the left most octet of the calculated checksum.
CK_2	The value of the right most octet of the calculated checksum.

B.2.3 Encoding procedure

The encoding procedure takes as input N octets of data to be checked and generates a 16-bit checksum value. This checksum value is placed in the packet error control field.

The algorithm used is:

Initialize C0 and C1 to zero.

Process each octet of the data to be checked, sequentially from i = 1 to N as follows:

C0 = (C0 + Bi) modulo 255

C1 = (C1 + C0) modulo 255

Calculate an intermediate checksum value as:

CK1 = ~(C0 + C1) //The bits are flipped.

CK2 = C1

If CK1 = 0, then CK1 = 255.

If CK2 = 0, then CK1 = 255.

Place the resulting values of CK1 and CK2 in their destination fields.

B.2.4 Decoding procedure

The decoding procedure takes as input N+ 2 octets of data to be checked and reports whether an error is detected or not. The N+2 octets consist of:

- the N octets of data to be checked (the data packet proper), and
- the 2 checksum octets that are appended to the N octets of data.

The algorithm used is:

If either, but not both, checksum octets contain the value zero, then report Error-Detected.

Initialize C0 and C1 to zero.

Process each octet of the data to be checked, sequentially from i = 1 to N+2 as follows:

C0 = (C0 + Bi) modulo 255

C1 = (C1 + C0) modulo 255

When all the octets have been processed, if the values of C0 and C1 are both zero, then report No-Error-Detected; otherwise report Error-Detected.

B.2.5 Verification of compliance

The binary sequences defined in Table B-4 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.

Table B-4 Verification of ISO compliance

data	CRC
00 00	FF FF
00 00 00	FF FF
AB CD EF 01	9C F8
14 56 F8 9A 00 01	24 DC

B.2.6 Software implementation

```
#include <stdio.h>
#include <stdint.h>
#define ERROR_DETECTED 0
#define NO_ERROR_DETECTED 1
/* Encoding procedure */
/* NOTE: Assumption is that enough memory has been allocated for byte */
/* stream B to allow for generation of the checksum value. */
/* The two checksum octets are placed in the destination field */
/* (as Nth and Nth + 1 octet of byte stream B). */
/* The destination field is also known as the packet error  */
/* control field. */
void iso16_encode(B, octets)
  uint8_t* B;        /* Buffer to be checked */
  uint32_t octets;   /* Length of the buffer */
{
  uint8_t  C0;
  uint8_t  C1;
  uint8_t  CK1;
  uint8_t  CK2;
  uint32_t index;
  /* Initialize C0 and C1 to zero */
  C0 = 0;
  C1 = 0;
  /* Process each octet of the data to be checked, sequentially from index = 1 to
octets as follows: */
  for (index = 0; index < octets; index++ ) {
    /*   C0 = (C0 + Bi) modulo 255 */
    C0 = ((C0 + B[index]) % 255);
    /*   C1 = (C1 + C0) modulo 255 */
    C1 = (C1 + C0) % 255;
  }
  /* Calculate an intermediate checksum value as: */
  /* CK1 = ~((C0 + C1) % 255); // flip the bits (~) for negative 1's complement */
  /* CK2 = C1; */
  /* if (0 == CK1) CK1 = 255; */
  /* if (0 == CK2) CK2 = 255; */
  CK1 = ~((C0 + C1) % 255); /* flip the bits (~) for negative 1's complement */
  CK2 = C1;
  if (0 == CK1) CK1 = 255;
  if (0 == CK2) CK2 = 255;
  /* Place the resulting values of CK1 and CK2 in their destination fields. */
  B[octets] = CK1;
  B[octets + 1] = CK2;
}
/* Decoding procedure of the buffer including the calculated ISO checksum in the
last two octets */
uint16_t iso16_decode(B, octets)
```

```
  uint8_t* B;        /* Buffer to be decoded */
  uint32_t octets;   /* Length of the buffer */
{
  uint8_t  C0;
  uint8_t  C1;
  uint32_t index;
  /* The last two octets (at position octets-2 and octets-1) contain the calculated
checksum. */
  /* If either, but not both, checksum octets contains the value zero, then report
Error-Detected. */
  if ((B[octets-2] == 0 && B[octets-1] !=0) || (B[octets-1] == 0 && B[octets-2] !=
0))
    return ERROR_DETECTED;
  /* Initialize C0 and C1 to zero */
  C0 = 0;
  C1 = 0;
  /* Process each octet of the data to be checked, sequentially from index = 1 to
octets+2 as follows: */
  for ( index = 0; index < octets; index++ ) {
    /* C0 = (C0 + Bi) modulo 255 */
    C0 = (C0 + B[index]) % 255;
    /* C1 = (C1 + C0) modulo 255 */
    C1 = (C1 + C0) % 255;
  }
  /* When all the octets have been processed, if the values of C0 and C1 are both
zero, then  */
  /* report No-Error-Detected; otherwise report Error-Detected. */
  if (C0 == 0 && C1 == 0)
    return NO_ERROR_DETECTED;
  else
    return ERROR_DETECTED;
}
/* Print a buffer in hexadecimal format */
void print_buffer(B, octets)
  uint8_t* B;        /* Buffer to be displayed */
  uint32_t octets;   /* Length of the buffer */
{
  uint32_t index;
  printf("Data field with calculated ISO Checksum is: ");
  for (index = 0; index < octets; index++)
    printf("%02X ", B[index]);
}
/* Display the message related to the result of a decoding of the buffer */
void print_status(result)
  uint32_t result;          /* Result to be displayed */
{
  if (result == ERROR_DETECTED) {
    printf(" - Error-Detected decoding checksum\n");
    printf(" This can mean that either:\n");
    printf(" 1. One of the two checksum octets initially contains the value 0,
or\n");
    printf(" 2. The calculated checksum does not result in two octets with value
0\n");
  } else {
    printf(" - No-Error-Detected decoding checksum\n");
  }
}
/* Verification of compliance */
int main()
{
  uint32_t N;
  uint16_t result;
```

```
    /* Declaration of test data (note that two extra octets are declared */
    /* for each data sequence to reserve room for the two checksum octets) */
    uint8_t VData1[] = {0x00, 0x00, 0x00, 0x00};
    uint8_t VData2[] = {0x00, 0x00, 0x00, 0x00, 0x00};
    uint8_t VData3[] = {0xab, 0xcd, 0xef, 0x01, 0x00, 0x00};
    uint8_t VData4[] = {0x14, 0x56, 0xf8, 0x9a, 0x00, 0x01, 0x00, 0x00};
    /* Encode VData1 */
    N = 2;
    iso16_encode(VData1, N);
    /* The last 2 octets of VData1 now contain the checksum */
    print_buffer(VData1, N + 2);
    /* Decode VData1 */
    result = iso16_decode(VData1, N + 2);
    print_status(result);
    /* Encode VData2 */
    N = 3;
    iso16_encode(VData2, N);
    /* The last 2 octets of VData2 now contain the checksum */
    print_buffer(VData2, N + 2);
    /* Decode VData2 */
    result = iso16_decode(VData2, N + 2);
    print_status(result);
    /* Encode VData3 */
    N = 4;
    iso16_encode(VData3, N);
    /* The last 2 octets of VData3 now contain the checksum */
    print_buffer(VData3, N + 2);
    /* Decode VData3 */
    result = iso16_decode(VData3, N + 2);
    print_status(result);
    /* Encode VData4 */
    N = 6;
    iso16_encode(VData4, N);
    /* The last 2 octets of VData4 now contain the checksum */
    print_buffer(VData4, N + 2);
    /* Decode VData4 */
    result = iso16_decode(VData4, N + 2);
    print_status(result);
    return 0;
}
/* This program results in the following output:
Data field with calculated ISO Checksum is: 00 00 FF FF - No-Error-Detected
decoding checksum
Data field with calculated ISO Checksum is: 00 00 00 FF FF - No-Error-Detected
decoding checksum
Data field with calculated ISO Checksum is: AB CD EF 01 9C F8 - No-Error-Detected
decoding checksum
Data field with calculated ISO Checksum is: 14 56 F8 9A 00 01 24 DC - No-Error-
Detected decoding checksum
*/
```

Annex C(informative) Summary of requests and reports for PUS standard services

C.1 Convention

This annex provides a summary of the message types defined in this Standard.

The summary is organised per service and subservice types.

The tailoring rules used during the deployment of the service type model for a given mission, i.e. to identify what message type applies to what service are also reported in that annex.

Each message type is associated to its applicability constraint (refer to the applicability constraint of the related capability type, requirement 5.3.5b).

C.2 Requests and reports

C.2.1 ST[01] request verification

C.2.1.1. Acceptance and reporting

Table C-1 shows the message types of the acceptance and reporting subservice type.

Table C-1 Acceptance and reporting message types

system	interface	message type		
6.1.4.2	8.1.2.1	TM[1,1]	successful acceptance verification report	minimum
6.1.4.3	8.1.2.2	TM[1,2]	failed acceptance verification report	minimum

C.2.1.2. Execution reporting

Table C-2 shows the message types of the execution reporting subservice type.

Table C-2 Execution reporting message types

system	interface	message type		
6.1.5.1.1	8.1.2.3	TM[1,3]	successful start of request execution verification report	minimum
6.1.5.1.2	8.1.2.4	TM[1,4]	failed start of request execution verification report	minimum
6.1.5.2.1	8.1.2.5	TM[1,5]	successful progress of request execution verification report	minimum
6.1.5.2.2	8.1.2.6	TM[1,6]	failed progress of request execution verification report	minimum
6.1.5.3.1	8.1.2.7	TM[1,7]	successful completion of request execution verification report	minimum
6.1.5.3.2	8.1.2.8	TM[1,8]	failed completion of request execution verification report	minimum

C.2.1.3. Routing and reporting

Table C-3 shows the message types of the routing and reporting subservice type.

Table C-3 Routing and reporting message types

system	interface	message type		
6.1.3.3	8.1.2.9	TM[1,10]	failed routing verification report	minimum

C.2.2 ST[02] device access

C.2.2.1. Device access

Table C-4 shows the message types of the device access subservice type.

Table C-4 Device access message types

system	interface	message type		
6.2.3a		at least one of: • TC[2,1] • TC[2,2] • TC[2,4] • TC[2,7]		minimum
6.2.4.2	8.2.2.1	TC[2,1]	distribute on/off device commands	by declaration

system	interface	message type		
6.2.5.2	8.2.2.2	TC[2,2]	distribute register load commands	by declaration
6.2.5.3	8.2.2.4	TC[2,5]	distribute register dump commands	requires TC[2,2]
6.2.5.3	8.2.2.5	TM[2,6]	register dump report	TC[2,5] response
6.2.6.2	8.2.2.3	TC[2,4]	distribute CPDU commands	by declaration
6.2.7.1.2	8.2.2.6	TC[2,7]	distribute physical device commands	by declaration
6.2.7.1.3	8.2.2.7	TC[2,8]	acquire data from physical devices	implied by TC[2,7]
6.2.7.1.3	8.2.2.8	TM[2,9]	physical device data report	TC[2,8] response
6.2.7.2.2	8.2.2.9	TC[2,10]	distribute logical device commands	requires TC[2,7]
6.2.7.2.3	8.2.2.10	TC[2,11]	acquire data from logical devices	implied by TC[2,10]
6.2.7.2.3	8.2.2.11	TM[2,12]	logical device data report	TC[2,11] response

C.2.3 ST[03] housekeeping

C.2.3.1. Housekeeping reporting

Table C-5 shows the message types of the housekeeping reporting subservice type.

Table C-5 Housekeeping reporting message types

system	interface	message type		
6.3.3.4	1.1.1.1	TM[3,25]	category 1 housekeeping parameter report	by declaration
6.3.3.4	1.1.1.1	TM[3,26]	category 2 housekeeping parameter report	by declaration
6.3.3.4	1.1.1.1	TM[3,50]	category 3 housekeeping parameter report	by declaration
6.3.3.4	1.1.1.1	TM[3,51]	category 4 housekeeping parameter report	by declaration

system	interface	message type		
6.3.3.4	1.1.1.1	TM[3,52]	category 5 housekeeping parameter report	by declaration
6.3.3.4	1.1.1.1	TM[3,53]	category 6 housekeeping parameter report	by declaration
6.3.3.5.1	8.3.2.3	TC[3,5]	enable the periodic generation of housekeeping parameter reports	by declaration
6.3.3.5.2	8.3.2.4	TC[3,6]	disable the periodic generation of housekeeping parameter reports	implied by TC[3,5]
6.3.3.6.1	8.3.2.1	TC[3,1]	create a housekeeping parameter report structure	by declaration
6.3.3.6.2	8.3.2.2	TC[3,3]	delete housekeeping parameter report structures	implied by TC[3,1]
6.3.3.7	8.3.2.5	TC[3,9]	report housekeeping parameter report structures	requires TC[3,1]
6.3.3.7	8.3.2.6	TM[3,10]	housekeeping parameter report structure report	TC[3,9] response
6.3.3.9	8.3.2.8	TC[3,29]	append parameters to a housekeeping parameter report structure	requires TC[3,1]
6.3.3.10	8.3.2.9	TC[3,31]	modify the collection interval of housekeeping parameter report structures	by declaration
6.3.3.11	8.3.2.1010	TC[3,33]	report the periodic generation properties of housekeeping parameter report structures	by declaration
6.3.3.11	8.3.2.11	TM[3,34]	housekeeping parameter report periodic generation properties report	TC[3,33] response
6.3.3.8	8.3.2.77	TC[3,27]	generate a one shot report for housekeeping parameter report structures	by declaration

C.2.3.2. Parameter functional reporting configuration

Table C-6 shows the message types of the parameter functional reporting configuration subservice type.

Table C-6 Parameter functional reporting configuration message types

system	interface	message type		
6.3.5.3	8.3.2.12	TC[3,37]	apply parameter functional reporting configurations	minimum
6.3.5.4.1	8.3.2.13	TC[3,38]	create a parameter functional reporting definition	by declaration
6.3.5.4.2	8.3.2.1414	TC[3,39]	delete parameter functional reporting definitions	implied by TC[3,38]
6.3.5.5	8.3.2.1515	TC[3,40]	report parameter functional reporting definitions	requires TC[3,38]
6.3.5.5	8.3.2.1616	TM[3,41]	parameter functional reporting definition report	TC[3,40] response
6.3.5.6.1	8.3.2.17	TC[3,42]	add parameter report definitions to a parameter functional reporting definition	requires TC[3,38]
6.3.5.6.2	8.3.2.18	TC[3,43]	remove parameter report definitions from a parameter functional reporting definition	implied by TC[3,42]
6.3.5.6.3	8.3.2.19	TC[3,44]	modify the periodic generation properties of parameter report definitions of a parameter functional reporting definition	by declaration

C.2.4 ST[04] parameter statistics reporting

C.2.4.1. Parameter statistics reporting

Table C-7 shows the message types of the parameter statistics reporting subservice type.

Table C-7 Parameter statistics reporting message types

system	interface	message type		
6.4.4	8.4.2.3	TC[4,3]	reset the parameter statistics	minimum
6.4.5.2	8.4.2.1	TC[4,1]	report the parameter statistics	minimum
6.4.5.3	8.4.2.2	TM[4,2]	parameter statistics report	TC[4,1] response implied by 6.4.6.1a
6.4.6.1a		support for the periodic reporting of the results of the parameter statistics evaluation		by declaration

system	interface	message type		
6.4.6.2	8.4.2.4	TC[4,4]	enable the periodic parameter statistics reporting	implied by 6.4.6.1a
6.4.6.3	8.4.2.5	TC[4,5]	disable the periodic parameter statistics reporting	implied by 6.4.6.1a
6.4.7.1	8.4.2.6	TC[4,6]	add or update parameter statistics definitions	by declaration
6.4.7.2	8.4.2.7	TC[4,7]	delete parameter statistics definitions	implied by TC[4,6]
6.4.7.3	8.4.2.8	TC[4,8]	report the parameter statistics definitions	requires TC[4,6]
6.4.7.3	8.4.2.9	TM[4,9]	parameter statistics definition report	TC[4,8] response

C.2.5 ST[05] event reporting

C.2.5.1. Event reporting

Table C-8 shows the message types of the event reporting subservice type.

Table C-8 Event reporting message types

system	interface	message type		
6.5.4	8.5.2.1	TM[5,1]	informative event report	minimum
6.5.4	8.5.2.2	TM[5,2]	low severity anomaly report	minimum
6.5.4	8.5.2.3	TM[5,3]	medium severity anomaly report	minimum
6.5.4	8.5.2.4	TM[5,4]	high severity anomaly report	minimum
b	8.5.2.5	TC[5,5]	enable the report generation of event definitions	by declaration
6.5.5.3	8.5.2.6	TC[5,6]	disable the report generation of event definitions	implied by TC[5,5]
6.5.5.4	8.5.2.7	TC[5,7]	report the list of disabled event definitions	requires TC[5,5]
6.5.5.4	8.5.2.8	TM[5,8]	disabled event definitions list report	TC[5,7] response
b	8.5.2.5	TC[5,9]	generate event report	by declaration

C.2.6 ST[06] memory management

C.2.6.1. Raw data memory management

Table C-9 shows the message types of the raw data memory management subservice type.

Table C-9 Raw data memory management message types

system	interface	message type		
6.6.3.3.1	8.6.2.2	TC[6,2]	load raw memory data areas	minimum
6.6.3.4	8.6.2.5	TC[6,5]	dump raw memory data	minimum
6.6.3.4	8.6.2.6	TM[6,6]	dumped raw memory data report	TC[6,5] response
6.6.3.5	8.6.2.9	TC[6,9]	check raw memory data	by declaration
6.6.3.5	8.6.2.10	TM[6,10]	checked raw memory data report	TC[6,9] response
6.6.3.6	8.6.2.19	TC[6,19]	load raw memory data areas by reference	by declaration
6.6.3.7	8.6.2.20	TC[6,20]	dump raw memory data areas to file	by declaration
6.6.3.3.2	8.6.2.11	TC[6,11]	load a raw memory atomic data area in a non-interruptible transaction	by declaration

C.2.6.2. Structured data memory management

Table C-10 shows the message types of the structured data memory management subservice type.

Table C-10 Structured data memory management message types

system	interface	message type		
6.6.4.4	8.6.2.1	TC[6,1]	load object memory data	minimum
6.6.4.5	8.6.2.3	TC[6,3]	dump object memory data	minimum
6.6.4.5	8.6.2.4	TM[6,4]	dumped object memory data report	TC[6,3] response
6.6.4.6	8.6.2.7	TC[6,7]	check object memory data	by declaration
6.6.4.6	8.6.2.8	TM[6,8]	checked object memory data report	TC[6,7] response

system	interface	message type		
6.6.4.7	8.6.2.17	TC[6,17]	check an object memory object	by declaration
6.6.4.7	8.6.2.18	TM[6,18]	checked object memory object report	TC[6,17] response
6.6.4.8	8.6.2.21	TC[6,21]	load object memory data areas by reference	by declaration
6.6.4.9	8.6.2.22	TC[6,22]	dump object memory data areas to file	by declaration

C.2.6.3. Common memory management

Table C-11 shows the message types of the common memory management subservice type.

Table C-11 Common memory management message types

system	interface	message type		
6.6.5.1	8.6.2.12	TC[6,12]	abort all memory dumps	minimum

C.2.6.4. Memory configuration

Table C-12 shows the message types of the memory configuration subservice type.

Table C-12 Memory configuration message types

system	interface	message type		
6.6.6.1.1a		scrubbing memories support		by declaration
6.6.6.1.4	8.6.2.13	TC[6,13]	enable the scrubbing of a memory	implied by 6.6.6.1.1a
6.6.6.1.5	8.6.2.14	TC[6,14]	disable the scrubbing of a memory	implied by 6.6.6.1.1a
6.6.6.2.1a		write protecting memories support		by declaration
6.6.6.2.4	8.6.2.15	TC[6,15]	enable the write protection of a memory	implied by 6.6.6.2.1a
6.6.6.2.5	8.6.2.16	TC[6,16]	disable the write protection of a memory	implied by 6.6.6.2.1a

C.2.7 ST[07] (reserved)

C.2.8 ST[08] (reserved)

C.2.9 ST[09] time management

C.2.9.1. Time reporting

Table C-13 shows the message types of the time reporting subservice type.

Table C-13 Time reporting message types

system	interface	message type		
6.9.4.1a		exactly one of: • TM[9,2] • TM[9,3]		minimum
6.9.4.2	8.9.2.2	TM[9,2]	CUC time report	by declaration
6.9.4.3	8.9.2.3	TM[9,3]	CDS time report	by declaration

C.2.9.2. Time reporting control

Table C-14 shows the message types of the time reporting control subservice type.

Table C-14 Time reporting control message types

system	interface	message type		
6.9.5.1.1	8.9.2.1	TC[9,1]	set the time report generation rate	minimum

C.2.9.3. Time control

Table C-14 shows the message types of the time control subservice type.

Table C-15 Time control message types

system	interface	message type		
6.9.5.1.1	8.9.2.1	TC[9,4]	set the reference time with absolute time	minimum
6.9.5.1.1	8.9.2.1	TC[9,5]	set the reference time with relative time	minimum

C.2.9.4. Time distribution

Table C-14 shows the message types of the time distribution subservice type.

Table C-16 Time distribution message types

system	interface	message type		
6.9.5.1.1	8.9.2.1	TC[9,6]	start time distribution to on-board users	minimum
6.9.5.1.1	8.9.2.1	TC[9,7]	stop time distribution to on-board users	minimum

C.2.10 ST[10] (reserved)

C.2.11 ST[11] time-based scheduling

C.2.11.1. Time-based scheduling

Table C-17 shows the message types of the time-based scheduling subservice type.

Table C-17 Time-based scheduling message types

system	interface	message type		
6.11.4.3.2	8.11.2.1	TC[11,1]	enable the time-based schedule execution function	minimum
6.11.4.3.3	8.11.2.2	TC[11,2]	disable the time-based schedule execution function	minimum
6.11.4.4	8.11.2.3	TC[11,3]	reset the time-based schedule	minimum
6.11.4.5	8.11.2.4	TC[11,4]	insert activities into the time-based schedule	minimum
6.11.5.2.1	8.11.2.20	TC[11,20]	enable time-based sub-schedules	by declaration
6.11.5.2.2	8.11.2.21	TC[11,21]	disable time-based sub-schedules	implied by TC[11,20]
6.11.5.2.3	8.11.2.18	TC[11,18]	report the status of each time-based sub-schedule	requires TC[11,20]
6.11.5.2.3	8.11.2.19	TM[11,19]	time-based sub-schedule status report	TC[11,18] response
6.11.6.2.1	8.11.2.22	TC[11,22]	create time-based scheduling groups	by declaration
6.11.6.2.2	8.11.2.23	TC[11,23]	delete time-based scheduling groups	implied by TC[11,22]

system	interface	message type		
6.11.6.3.1	8.11.2.24	TC[11,24]	enable time-based scheduling groups	implied by TC[11,22]
6.11.6.3.2	8.11.2.25	TC[11,25]	disable time-based scheduling groups	implied by TC[11,24]
6.11.6.3.3	8.11.2.26	TC[11,26]	report the status of each time-based scheduling group	requires TC[11,22]
6.11.6.3.3	8.11.2.27	TM[11,27]	time-based scheduling group status report	TC[11,26] response
6.11.8.1	8.11.2.15	TC[11,15]	time-shift all scheduled activities	by declaration
6.11.8.2	8.11.2.17	TC[11,17]	summary-report all time-based scheduled activities	by declaration
6.11.7.1	8.11.2.13	TM[11,13]	time-based schedule summary report	TC[11,17] response
6.11.8.3	8.11.2.16	TC[11,16]	detail-report all time-based scheduled activities	by declaration
6.11.7.2	8.11.2.10	TM[11,10]	time-based schedule detail report	TC[11,10] response
6.11.9.2	8.11.2.5	TC[11,5]	delete time-based scheduled activities identified by request identifier	by declaration
6.11.9.3	8.11.2.7	TC[11,7]	time-shift scheduled activities identified by request identifier	by declaration
6.11.9.4	8.11.2.12	TC[11,12]	Summary-report time-based scheduled activities identified by request identifier	by declaration
6.11.7.1	8.11.2.13	TM[11,13]	time-based schedule summary report	TC[11,12] response
6.11.9.5	8.11.2.9	TC[11,9]	detail-report time-based scheduled activities identified by request identifier	by declaration
6.11.7.2	8.11.2.10	TM[11,10]	time-based schedule detail report	TC[11,9] response
6.11.10.3	8.11.2.6	TC[11,6]	delete the time-based scheduled activities identified by a filter	by declaration
6.11.10.4	8.11.2.8	TC[11,8]	time-shift the scheduled activities identified by a filter	by declaration
6.11.10.5	8.11.2.14	TC[11,14]	summary-report the time-based scheduled activities identified by a filter	by declaration

system	interface	message type		
6.11.7.1	8.11.2.13	TM[11,13]	time-based schedule summary report	TC[11,14] response
6.11.10.6	8.11.2.11	TC[11,11]	detail-report the time-based scheduled activities identified by a filter	by declaration
6.11.7.2	8.11.2.10	TM[11,10]	time-based schedule detail report	TC[11,11] response

C.2.12 ST[12] on-board monitoring

C.2.12.1. Parameter monitoring

Table C-18 shows the message types of the parameter monitoring subservice type.

Table C-18 Parameter monitoring message types

system	interface	message type		
6.12.3.5.1	8.12.2.15	TC[12,15]	enable the parameter monitoring function	minimum
6.12.3.5.2	8.12.2.16	TC[12,16]	disable the parameter monitoring function	minimum
6.12.3.6.1	8.12.2.1	TC[12,1]	enable parameter monitoring definitions	minimum
6.12.3.6.2	8.12.2.2	TC[12,2]	disable parameter monitoring definitions	minimum
6.12.3.7.1	8.12.2.12	TM[12,12]	check transition report	minimum
6.12.3.7.2	8.12.2.3	TC[12,3]	change the maximum transition reporting delay	by declaration
6.12.3.7.3	8.12.2.29	TC[12,29]	enable check transition list logging	by declaration
6.12.3.7.4	8.12.2.30	TC[12,30]	disable check transition list logging	by declaration
6.12.3.8.1	8.12.2.5	TC[12,5]	add parameter monitoring definitions	by declaration
6.12.3.8.1b		if TC[12,5], at least one of: • TC[12,4] • TC[12,6]		implied by TC[12,5]
6.12.3.8.2	8.12.2.4	TC[12,4]	delete all parameter monitoring definitions	by declaration
6.12.3.8.3	8.12.2.6	TC[12,6]	delete parameter monitoring definitions	by declaration

system	interface	message type		
6.12.3.8.4	8.12.2.7	TC[12,7]	modify parameter monitoring definitions	by declaration
6.12.3.9	8.12.2.8	TC[12,8]	report parameter monitoring definitions	requires TC[12,5] or TC[12,7]
6.12.3.9	8.12.2.9	TM[12,9]	parameter monitoring definition report	TC[12,8] response
6.12.3.10	8.12.2.13	TC[12,13]	report the status of each parameter monitoring definition	requires TC[12,1]
6.12.3.10	8.12.2.14	TM[12,14]	parameter monitoring definition status report	TC[12,13] response
6.12.3.11	8.12.2.10	TC[12,10]	report the current out-of-limits	by declaration
6.12.3.11	8.12.2.11	TM[12,11]	current out-of-limits report	TC[12,10] response

C.2.12.2. Functional monitoring

Table C-19 shows the message types of the functional monitoring subservice type.

Table C-19 Functional monitoring message types

system	interface	message type		
6.12.4.4.1	8.12.2.17	TC[12,17]	enable the functional monitoring function	minimum
6.12.4.4.2	8.12.2.18	TC[12,18]	disable the functional monitoring function	minimum
6.12.4.5.2	8.12.2.19	TC[12,19]	enable functional monitoring definitions	minimum
6.12.4.5.3	8.12.2.20	TC[12,20]	disable functional monitoring definitions	minimum
6.12.4.6.1	8.12.2.21	TC[12,21]	protect functional monitoring definitions	by declaration
6.12.4.6.2	8.12.2.22	TC[12,22]	unprotect functional monitoring definitions	implied by TC[12,21]
6.12.4.7.1	8.12.2.23	TC[12,23]	add functional monitoring definitions	by declaration
6.12.4.7.2	8.12.2.24	TC[12,24]	delete functional monitoring definitions	implied by TC[12,23]
6.12.4.8	8.12.2.25	TC[12,25]	report functional monitoring definitions	requires TC[12,23]

system	interface	message type		
6.12.4.8	8.12.2.26	TM[12,26]	functional monitoring definition report	TC[12,25] response
6.12.4.9	8.12.2.27	TC[12,27]	report the status of each functional monitoring definition	by declaration
6.12.4.9	8.12.2.28	TM[12,28]	functional monitoring definition status report	TC[12,27] response

C.2.13 ST[13] large packet transfer

C.2.13.1. Large packet downlink

Table C-20 shows the message types of the large packet downlink subservice type.

Table C-20 Large packet downlink message types

system	interface	message type		
6.13.3.3.1	8.13.2.1	TM[13,1]	first downlink part report" for the first part	minimum
6.13.3.3.1	8.13.2.2	TM[13,2]	intermediate downlink part report" for the intermediate parts	minimum
6.13.3.3.1	8.13.2.3	TM[13,3]	last downlink part report" for the last part	minimum

C.2.13.2. Large packet uplink

Table C-21 shows the message types of the large packet uplink subservice type.

Table C-21 Large packet uplink message types

system	interface	message type		
6.13.4.3.1	8.13.2.4	TC[13,9]	uplink the first part" for the first part	minimum
6.13.4.3.1	8.13.2.5	TC[13,10]	uplink an intermediate part" for the intermediate parts	minimum
6.13.4.3.1	8.13.2.6	TC[13,11]	uplink the last part" for the last part	minimum
6.13.4.3.3	8.13.2.7	TM[13,16]	large packet uplink abortion report	minimum

C.2.14 ST[14] real-time forwarding control

C.2.14.1. Real-time forwarding control

Table C-22 shows the message types of the real-time forwarding control subservice type.

Table C-22 Real-time forwarding control message types

system	interface	message type		
6.14.3.5.1	8.14.2.1	TC[14,1]	add report types to the application process forward-control configuration	minimum
6.14.3.5.2	8.14.2.2	TC[14,2]	delete report types from the application process forward-control configuration	minimum
6.14.3.5.3	8.14.2.3	TC[14,3]	report the content of the application process forward-control configuration	requires TC[14,1]
6.14.3.5.3	8.14.2.4	TM[14,4]	application process forward-control configuration content report	TC[14,3] response
6.14.3.3.1a		capability to control, per housekeeping parameter report structure, the forwarding of housekeeping parameter reports		by declaration
6.14.3.6.1	8.14.2.5	TC[14,5]	add structure identifiers to the housekeeping parameter report forward-control configuration	implied by 6.14.3.3.1a
6.14.3.6.2	8.14.2.6	TC[14,6]	delete structure identifiers from the housekeeping parameter report forward-control configuration	implied by 6.14.3.3.1a
6.14.3.6.3	8.14.2.7	TC[14,7]	report the content of the housekeeping parameter report forward-control configuration	requires TC[14,5]
6.14.3.6.3	8.14.2.8	TM[14,8]	housekeeping parameter report forward-control configuration content report	TC[14,7] response
6.14.3.3.1b		capability to control, per event definition, the forwarding of event reports		by declaration
6.14.3.7.2	8.14.2.10	TC[14,14]	add event definition identifiers to the event report blocking forward-control configuration	implied by 6.14.3.3.1b
6.14.3.7.1	8.14.2.9	TC[14,13]	delete event definition identifiers from the event report blocking forward-control configuration	implied by 6.14.3.3.1b
6.14.3.7.3	8.14.2.11	TC[14,15]	report the content of the event report blocking forward-control configuration	requires TC[14,14]

system	interface	message type		
6.14.3.7.3	8.14.2.12	TM[14,16]	event report blocking forward-control configuration content report	TC[14,15] response

C.2.15 ST[15] on-board storage and retrieval

C.2.15.1. Storage and retrieval

Table C-23 shows the message types of the storage and retrieval subservice type.

Table C-23 Storage and retrieval message types

system	interface	message type		
6.15.3.4.2	8.15.2.1	TC[15,1]	enable the storage function of packet stores	minimum
6.15.3.4.3	8.15.2.2	TC[15,2]	disable the storage function of packet stores	minimum
6.15.3.5.2	8.15.2.12	TC[15,14]	change the open retrieval start time tag of packet stores	minimum
6.15.3.5.3	8.15.2.13	TC[15,15]	resume the open retrieval of packet stores	implied by TC[15,16]
6.15.3.5.4	8.15.2.14	TC[15,16]	suspend the open retrieval of packet stores	minimum
6.15.3.6.1a		by-time-range retrieval function support		by declaration
6.15.3.6.2	8.15.2.7	TC[15,9]	start the by-time-range retrieval of packet stores	implied by 6.15.3.6.1a
6.15.3.6.2	8.15.2.7	TC[15,10]	abort all requests to copy the packets contained in a packet store selected by time window	implied by TC[15,24]
6.15.3.6.3	8.15.2.15	TC[15,17]	abort the by-time-range retrieval of packet stores	implied by 6.15.3.6.1a
6.15.3.7	8.15.2.16	TC[15,18]	report the status of each packet store	by declaration
6.15.3.7	8.15.2.17	TM[15,19]	packet store status report	TC[15,18] response
6.15.3.8.1	8.15.2.8	TC[15,11]	delete the content of packet stores up to the specified time	by declaration
6.15.3.9.1	8.15.2.18	TC[15,20]	create packet stores	by declaration
6.15.3.9.2	8.15.2.19	TC[15,21]	delete packet stores	implied by TC[15,20]

system	interface	message type		
6.15.3.9.3	8.15.2.20	TC[15,22]	report the configuration of each packet store	requires TC[15,20]
6.15.3.9.3	8.15.2.21	TM[15,23]	packet store configuration report	TC[15,22] response
6.15.3.9.4	8.15.2.22	TC[15,24]	copy the packets contained in a packet store selected by time window	requires TC[15,20]
6.15.3.10.1	8.15.2.23	TC[15,25]	resize packet stores	by declaration
6.15.3.10.2	8.15.2.24	TC[15,26]	change a packet store type to circular	by declaration
6.15.3.10.3	8.15.2.25	TC[15,27]	change a packet store type to bounded	implied by TC[15.26]
6.15.3.10.4	8.15.2.26	TC[15,28]	change the virtual channel used by a packet store	by declaration
6.15.3.10.5	8.15.2.35	TC[15,41]	change the default retrieval priority of a packet store	
6.15.3.11.1	8.15.2.1010	TC[15,12]	summary-report the content of packet stores	by declaration
6.15.3.11.1	8.15.2.11	TM[15,13]	packet store content summary report	TC[15,12] response

C.2.15.2. Packet selection

Table C-24 shows the message types of the packet selection subservice type.

Table C-24 Packet selection message types

system	Interface	message type		
6.15.4.4.1	8.15.2.3	TC[15,3]	add report types to the application process storage-control configuration	minimum
6.15.4.4.2	8.15.2.4	TC[15,4]	delete report types from the application process storage-control configuration	minimum
6.15.4.4.3	8.15.2.5	TC[15,5]	report the content of the application process storage-control configuration	requires TC[15,3]
6.15.4.4.3	8.15.2.6	TM[15,6]	application process storage-control configuration content report	TC[15,5] response
6.15.4.2.1a		control, per housekeeping parameter report structure, the storage of housekeeping parameter reports		by declaration

system	Interface	message type		
6.15.4.5.1	8.15.2.27	TC[15,29]	add structure identifiers to the housekeeping parameter report storage-control configuration	implied by 6.15.4.2.1a
6.15.4.5.2	8.15.2.28	TC[15,30]	delete structure identifiers from the housekeeping parameter report storage-control configuration	implied by 6.15.4.2.1a
6.15.4.5.3	8.15.2.31	TC[15,35]	report the content of the housekeeping parameter report storage-control configuration	requires TC[15,29]
6.15.4.5.3	8.15.2.32	TM[15,36]	housekeeping parameter report storage-control configuration content report	TC[15,36] response
6.15.4.2.1b		control, per event definition, the storage of event reports		by declaration
6.15.4.6.1	8.15.2.3030	TC[15,34]	add event definition identifiers to the event report blocking storage-control configuration	implied by 6.15.4.2.1b
6.15.4.6.2	8.15.2.29	TC[15,33]	delete event definition identifiers from the event report blocking storage-control configuration	implied by 6.15.4.2.1b
6.15.4.6.3	8.15.2.33	TC[15,39]	report the content of the event report blocking storage-control configuration	requires TC[15,33]
6.15.4.6.3	8.15.2.34	TM[15,40]	event report blocking storage-control configuration content report	TC[15,39] response

C.2.16 ST[16] (reserved)

C.2.17 ST[17] test

C.2.17.1. Test

Table C-25 shows the message types of the test subservice type.

Table C-25 Test message types

system	interface	message type		
6.17.3	8.17.2.1	TC[17,1]	perform an are-you-alive connection test	minimum
6.17.3	8.17.2.2	TM[17,2]	are-you-alive connection test report	TC[17,1] response
6.17.4.2	8.17.2.3	TC[17,3]	perform an on-board connection test	by declaration
6.17.4.2	8.17.2.4	TM[17,4]	on-board connection test report	TC[17,3] response
6.17.5	8.17.2.5	TC[17,5]	perform a variable size are-you-alive connection test	by declaration
6.17.5	8.17.2.6	TM[17,6]	variable size are-you-alive connection test report	TC[17,6] response

C.2.18 ST[18] on-board operations procedure

C.2.18.1. OBCP management

Table C-26 shows the message types of the OBCP management subservice type.

Table C-26 OBCP management message types

system	interface	message type		
6.18.4.4.1a		at least one of: • TC[18,1] • TC[18,13] • TC[18,19]		minimum
6.18.4.4.2	8.18.2.1	TC[18,1]	direct-load an OBCP	by declaration
6.18.4.4.3	8.18.2.11	TC[18,13]	load an OBCP by reference	by declaration
6.18.4.4.4	8.18.2.2	TC[18,2]	unload an OBCP	implied by TC[18,1] or TC[18,13]
6.18.4.4.5	8.18.2.3	TC[18,3]	activate an OBCP	minimum

system	interface	message type		
6.18.4.4.6	8.18.2.17	TC[18,19]	load by reference and activate an OBCP	by declaration
6.18.4.4.7	8.18.2.4	TC[18,4]	stop an OBCP	minimum
6.18.4.4.8	8.18.2.18	TC[18,20]	stop and unload an OBCP	by declaration
6.18.4.4.9	8.18.2.10	TC[18,12]	abort an OBCP	minimum
6.18.4.4.10	8.18.2.15	TC[18,17]	abort all OBCPs and report	by declaration
6.18.4.4.10	8.18.2.16	TM[18,18]	aborted OBCP report	TC[18,17] response
6.18.4.5.1	8.18.2.8	TC[18,8]	report the execution status of each OBCP	minimum
6.18.4.5.1	8.18.2.9	TM[18,9]	OBCP execution status report	TC[18,8] response
6.18.4.6.1	8.18.2.5	TC[18,5]	suspend an OBCP	by declaration
6.18.4.6.2	8.18.2.6	TC[18,6]	resume an OBCP	implied by TC[18,5]
6.18.4.6.3	8.18.2.12	TC[18,14]	activate and execute one OBCP step	by declaration
6.18.4.6.4	8.18.2.13	TC[18,15]	resume and execute one OBCP step	implied by TC[18,14]
6.18.4.7.1	8.18.2.7	TC[18,7]	communicate parameters to an OBCP	by declaration
6.18.4.8.1	8.18.2.14	TC[18,16]	set the observability level of OBCPs	by declaration

C.2.18.2. OBCP engine management

Table C-27 shows the message types of the OBCP engine management subservice type.

Table C-27 OBCP engine management message types

system	interface	message type		
6.18.5.2.1	8.18.2.19	TC[18,21]	start the OBCP engine	minimum
6.18.5.2.2	8.18.2.20	TC[18,22]	stop the OBCP engine	minimum

C.2.19 ST[19] event-action

C.2.19.1. Event-action

Table C-28 shows the message types of the event-action subservice type.

Table C-28 Event-action message types

system	interface	message type		
6.19.6.1	8.19.2.8	TC[19,8]	enable the event-action function	minimum
6.19.6.2	8.19.2.9	TC[19,9]	disable the event-action function	minimum
6.19.7.1	8.19.2.4	TC[19,4]	enable event-action definitions	minimum
6.19.7.2	8.19.2.5	TC[19,5]	disable event-action definitions	minimum
6.19.8.1	8.19.2.1	TC[19,1]	add event-action definitions	minimum
6.19.8.2a		at least one of: TC[19,2] TC[19,3]		implied by TC[19,1]
6.19.8.3	8.19.2.2	TC[19,2]	delete event-action definitions	by declaration
6.19.8.4	8.19.2.3	TC[19,3]	delete all event-action definitions	by declaration
6.19.8.5	8.19.2.6	TC[19,6]	report the status of each event-action definition	by declaration
6.19.8.5	8.19.2.7	TM[19,7]	event-action status report	TC[19,6] response
6.19.8.6	8.19.2.10	TC[19,10]	report event-action definitions	requires TC[19,1]
6.19.8.6	8.19.2.11	TM[19,11]	event-action definition report	TC[19,10] response

C.2.20 ST[20] Parameter management

C.2.20.1. Parameter management

Table C-29 shows the message types of the parameter management subservice type.

Table C-29 Parameter management message types

system	interface	message type		
6.20.4.1	8.20.2.1	TC[20,1]	report parameter values	minimum
6.20.4.1	8.20.2.2	TM[20,2]	parameter value report	TC[20,1] response
6.20.4.2	8.20.2.3	TC[20,3]	set parameter values	by declaration
6.20.5.2	8.20.2.4	TC[20,4]	change raw memory parameter definitions	by declaration
6.20.5.3	8.20.2.5	TC[20,5]	change object memory parameter definitions	by declaration
6.20.5.4	8.20.2.6	TC[20,6]	report parameter definitions	requires TC[20,4] or TC[20,5]
6.20.5.4	8.20.2.7	TM[20,7]	parameter definition report	TC[20,6] response

C.2.21 ST[21] request sequencing

C.2.21.1. Request sequencing

Table C-30 shows the message types of the request sequencing subservice type.

Table C-30 Request sequencing message types

system	interface	message type		
6.21.6.1a		at least one of: • TC[21,1] • TC[21,2] • TC[21,8]		minimum
6.21.6.2	8.21.2.1	TC[21,1]	direct-load a request sequence	by declaration
6.21.6.3	8.21.2.2	TC[21,2]	load a request sequence by reference	by declaration

system	interface	message type		
6.21.6.4	8.21.2.3	TC[21,3]	unload a request sequence	implied by TC[21,1] or TC[21,2]
6.21.6.6	8.21.2.8	TC[21,8]	load by reference and activate a request sequence	by declaration
6.21.6.5	8.21.2.4	TC[21,4]	activate a request sequence	minimum
6.21.6.7	8.21.2.5	TC[21,5]	abort a request sequence	minimum
6.21.6.8	8.21.2.13	TC[21,13]	abort all request sequences and report	by declaration
6.21.6.8	8.21.2.14	TM[21,14]	aborted request sequence report	TC[21,13] response
6.21.7	8.21.2.6	TC[21,6]	report the execution status of each request sequence	by declaration
6.21.7	8.21.2.7	TM[21,7]	request sequence execution status report	TC[21,6] response
6.21.8	8.21.2.9	TC[21,9]	checksum a request sequence	by declaration
6.21.8	8.21.2.10	TM[21,10]	request sequence checksum report	TC[21,9] response
6.21.9	8.21.2.11	TC[21,11]	report the content of a request sequence	by declaration
6.21.9	8.21.2.12	TM[21,12]	request sequence content report	TC[21,11] response

C.2.22 ST[22] position-based scheduling

C.2.22.1. Position-based scheduling

Table C-31 shows the message types of the position-based scheduling subservice type.

Table C-31 Position-based scheduling message types

system	interface	message type		
6.22.6.3.2	8.22.2.1	TC[22,1]	enable the position-based schedule execution function	minimum
6.22.6.3.3	8.22.2.2	TC[22,2]	disable the position-based schedule execution function	minimum
6.22.6.4	8.22.2.28	TC[22,28]	set the orbit number	by declaration
6.22.6.5	8.22.2.3	TC[22,3]	reset the position-based schedule	minimum
6.22.6.6	8.22.2.4	TC[22,4]	insert activities into the position-based schedule	minimum
6.22.7.2.1	8.22.2.20	TC[22,20]	enable position-based sub-schedules	by declaration
6.22.7.2.2	8.22.2.21	TC[22,21]	disable position-based sub-schedules	implied by TC[22,20]
6.22.7.2.3	8.22.2.18	TC[22,18]	report the status of each position-based sub-schedule	by declaration
6.22.7.2.3	8.22.2.19	TM[22,19]	position-based sub-schedule status report	TC[22,18] response
6.22.8.2.1	8.22.2.22	TC[22,22]	create position-based scheduling groups	by declaration
6.22.8.2.2	8.22.2.23	TC[22,23]	delete position-based scheduling groups	implied by TC[22,22]
6.22.8.3.1	8.22.2.24	TC[22,24]	enable position-based scheduling groups	implied by TC[22,22]
6.22.8.3.2	8.22.2.25	TC[22,25]	disable position-based scheduling groups	implied by TC[22,24]
6.22.8.3.3	8.22.2.26	TC[22,26]	report the status of each position-based scheduling group	requires TC[22,22]
6.22.8.3.3	8.22.2.27	TM[22,27]	position-based scheduling group status report	TC[22,26] response
6.22.10.2	8.22.2.15	TC[22,15]	position-shift all scheduled activities	by declaration

system	interface	message type		
6.22.10.3	8.22.2.17	TC[22,17]	summary-report all position-based scheduled activities	by declaration
6.22.9.13	8.22.2.13	TM[22,13]	position-based schedule summary report	TC[22,17] response
6.22.10.4	8.22.2.16	TC[22,16]	detail-report all position-based scheduled activities	by declaration
6.22.9.2	8.22.2.10	TM[22,10]	position-based schedule detail report	TC[22,16] response
6.22.11.2	8.22.2.5	TC[22,5]	delete position-based scheduled activities identified by request identifier	by declaration
6.22.11.3	8.22.2.7	TC[22,7]	position-shift scheduled activities identified by request identifier	by declaration
6.22.11.4	8.22.2.12	TC[22,12]	summary-report position-based scheduled activities identified by request identifier	by declaration
6.22.9.1	8.22.2.13	TM[22,13]	position-based schedule summary report	TC[22,12] response
6.22.11.5	8.22.2.9	TC[22,9]	detail-report position-based scheduled activities identified by request identifier	by declaration
6.22.9.2	8.22.2.10	TM[22,10]	position-based schedule detail report	TC[22,9] response
6.22.12.3	8.22.2.6	TC[22,6]	delete the position-based scheduled activities identified by a filter	by declaration
6.22.12.4	8.22.2.8	TC[22,8]	position-shift the scheduled activities identified by a filter	by declaration
6.22.12.5	8.22.2.14	TC[22,14]	summary-report the position-based scheduled activities identified by a filter	by declaration
6.22.9.1	8.22.2.13	TM[22,13]	position-based schedule summary report	TC[22,14] response
6.22.12.6	8.22.2.11	TC[22,11]	detail-report the position-based scheduled activities identified by a filter	by declaration
6.22.9.2	8.22.2.10	TM[22,10]	position-based schedule detail report	TC[22,11] response

C.2.23 ST[23] file management

C.2.23.1. File handling

Table C-32 shows the message types of the file handling subservice type.

Table C-32 File handling message types

system	interface	message type		
6.23.4.1.1	8.23.2.1	TC[23,1]	create a file	minimum
6.23.4.1.2	8.23.2.2	TC[23,2]	delete a file	minimum
6.23.4.2	8.23.2.3	TC[23,3]	report the attributes of a file	minimum
6.23.4.2	8.23.2.4	TM[23,4]	file attribute report	TC[23,3] response
6.23.4.3.1	8.23.2.5	TC[23,5]	lock a file	by declaration
6.23.4.3.2	8.23.2.6	TC[23,6]	unlock a file	implied by TC[23,5]
6.23.4.4	8.23.2.7	TC[23,7]	find files	by declaration
6.23.4.4	8.23.2.8	TM[23,8]	found files report	TC[23,7] response
6.23.4.6.1	8.23.2.9	TC[23,9]	create a directory	by declaration
6.23.4.6.2	8.23.2.10	TC[23,10]	delete a directory	implied by TC[23,9]
6.23.4.6.3	8.23.2.11	TC[23,11]	rename a directory	implied by TC[23,9]
6.23.4.7	8.23.2.12	TC[23,12]	report the content of a repository	by declaration
6.23.4.7	8.23.2.13	TM[23,13]	repository content report	TC[23,12] response
6.23.4.7	8.23.2.1225	TC[23,25]	checksum a file	by declaration
6.23.4.7	8.23.2.13	TM[23,26]	checksum report of a file	TC[23,25] response
6.23.4.7.4	8.23.2.1227	TC[23,27]	enable directory protection	by declaration
6.23.4.7.5	8.23.2.1228	TC[23,28]	disable directory protection	implied by TC[23,27]

C.2.23.2. File copy

Table C-33 shows the message types of the file copy subservice type.

Table C-33 File copy message types

system	interface	message type		
6.23.5.2.2	8.23.2.14	TC[23,14]	copy a file	minimum
6.23.5.2.3	8.23.2.15	TC[23,15]	move a file	by declaration
6.23.5.3.1	8.23.2.16	TC[23,16]	suspend file copy operations	by declaration
6.23.5.3.2	8.23.2.17	TC[23,17]	resume file copy operations	implied by TC[23,16]
6.23.5.3.3	8.23.2.19	TC[23,19]	suspend all file copy operations involving a repository path	by declaration
6.23.5.3.4	8.23.2.20	TC[23,20]	resume all file copy operations involving a repository path	implied by TC[23,19]
6.23.5.4.1	8.23.2.18	TC[23,18]	abort file copy operations	by declaration
6.23.5.4.2	8.23.2.21	TC[23,21]	abort all file copy operations involving a repository path	by declaration
6.23.5.5.2	8.23.2.22	TC[23,22]	enable the periodic reporting of the file copy status	by declaration
6.23.5.5.3	8.23.2.24	TC[23,24]	disable the periodic reporting of the file copy status	implied by TC[23,22]
6.23.5.5.4	8.23.2.23	TM[23,23]	file copy status report	TC[23,22] response

C.2.24 ST[24] file transfer

C.2.24.1. File transaction

Table C-33 shows the message types of the file transaction subservice type.

Table C-34 File transaction message types

System	interface	message type		
6.24.4.5	8.24.2.1	TC[24,1]	start of a file transmission opportunity window	minimum
6.24.4.6	8.24.2.2	TC[24,2]	stop of a file transmission opportunity window	implied by TC[24,1]
6.24.4.7	8.24.2.3	TC[24,3]	start of a file reception opportunity window	minimum
6.24.4.8	8.24.2.4	TC[24,4]	stop of a file reception opportunity window	implied by TC[24,3]
6.24.4.9	8.24.2.5	TC[24,5]	suspend a file transaction	minimum
6.24.4.10	8.24.2.6	TC[24,6]	resume a file transaction	implied by TC[24,5]
6.24.4.11	8.24.2.7	TC[24,7]	cancel a file transaction	minimum
6.24.4.12	8.24.2.8	TC[24,8]	report status of active file transactions	minimum
6.24.4.12	8.24.2.9	TM[24,9]	file transactions status report	TC[24,8] Response
6.24.4.13	8.24.2.10	TC[24,10]	modify CFDP entity configuration	minimum
6.24.4.14	8.24.2.11	TC[24,11]	report CFDP entity configuration	minimum
6.24.4.14	8.24.2.12	TM[24,12]	CFDP entity configuration report	TC[24,11] response

C.2.24.2. File downlink manager

Table C-34 shows the message types of the file downlink manager subservice type.

Table C-35 File downlink manager message types

System	interface	message type		
6.24.5.4	8.24.2.13	TC[24,13]	modify downlink manager configuration	minimum
6.24.5.5	8.24.2.14	TC[24,14]	add directories to the downlink manager directory configuration	minimum
6.24.5.6	8.24.2.15	TC[24,15]	remove directories from the downlink manager directory configuration	implied by TC[24,14]
6.24.5.7	8.24.2.16	TC[24,16]	report downlink manager directory configuration	minimum
6.24.5.7	8.24.2.17	TM[24,17]	downlink manager directory configuration report	TC[24,16] Response
6.24.5.8	8.24.2.18	TC[24,18]	start downlink manager	minimum
6.24.5.9	8.24.2.19	TC[24,19]	suspend downlink manager	minimum
6.24.5.10	8.24.2.20	TC[24,20]	stop downlink manager	implied by TC[24,18]

C.2.25 ST[25] file data storage

C.2.25.1. File data storage

Table C-35 shows the message types of the file data storage subservice type.

Table C-36 File data storage message types

system	interface	message type		
6.25.3.5	8.25.2.1	TC[25,1]	set directory data storage attributes	minimum
6.25.3.6	8.25.2.2	TC[25,2]	report directory data storage attributes	minimum
6.25.3.6	8.25.2.3	TM[25,3]	directory data storage attributes report	TC[25,2] response
6.25.3.7	8.25.2.4	TC[25,4]	add data sources to directory mapping	by declaration
6.25.3.8	8.25.2.5	TC[25,5]	remove data sources from directory mapping	implied by [25,4]
6.25.3.9	8.25.2.6	TC[25,6]	report data sources mapped to directory	requires [25,4]
6.25.3.9	8.25.2.7	TM[25,7]	data sources mapped to directory report	TC[25,6] response
6.25.3.10	8.25.2.8	TC[25,8]	report data source recording status	minimum
6.25.3.10	8.25.2.9	TM[25,9]	data source recording status report	TC[25,8] Response
6.25.3.11	8.25.2.10	TC[25,10]	enable data sources recording status	minimum
6.25.3.12	8.25.2.11	TC[25,11]	disable data sources recording status	implied by TC[25,10]
6.25.3.13	8.25.2.12	TC[25,12]	close file currently used for data storage	minimum

C.2.25.2. PUS report-type data source control

Table C-36 shows the message types of the PUS report-type data source control subservice type.

Table C-37 PUS report-type data source control message types

System	interface	message type		
6.25.4.4.1	8.25.2.13	TC[25,13]	add report-types to the application process PUS report-type data source control configuration	by declaration
6.25.4.4.2	8.25.2.14	TC[25,14]	delete report-types from the application process PUS report-type data source control configuration	implied by TC[25,13]
6.25.4.4.3	8.25.2.15	TC[25,15]	report the content of the application process PUS report-type data source control configuration	requires TC[25,13]
6.25.4.4.3	8.25.2.16	TM[25,16]	application process PUS report-type data source control configuration content report	TC[25,15] response

C.2.26 ST[26] parameter extraction

C.2.26.1. Parameter extraction

Table C-37 shows the message types of the parameter extraction subservice type.

Table C-38 Parameter extraction message types

System	interface	message type		
6.26.5.1	8.26.2.1	TC[26,1]	add parameter extraction definitions	minimum
6.26.5.2	8.26.2.2	TC[26,2]	delete parameter extraction definitions	implied by TC[26,1]
6.26.5.3	8.26.2.3	TC[26,3]	Report parameter extraction definitions	minimum
6.26.5.3	8.26.2.4	TM[26,4]	Parameter extraction definitions report	TC[26,3] response

C.2.27 ST[27] critical packet event log

C.2.27.1. Critical packet event log

Table C-38 shows the message types of the critical packet event log subservice type.

Table C-39 Critical packet event log message types

System	interface	message type		
6.27.4.3.2	8.27.2.1	TC[27,1]	downlink critical packet log	minimum
6.27.4.4.2	8.27.2.2	TC[27,2]	clear downlinked packets from critical packet log	minimum

Annex D(informative) System and interface specification index

service type name	service type ID	system	interface
		see page	
Request verification	1	53	492
Device access	2	62	498
Housekeeping	3	75	503
Parameter statistics reporting	4	98	516
Event reporting	5	108	520
Memory management	6	114	524
(reserved)	7		
(reserved)	8		
Time management	9	143	536
(reserved)	10		
Time-based scheduling	11	156	541
On-board monitoring	12	187	553
Large packet transfer	13	218	571
Real-time forwarding control	14	226	574
On-board storage and retrieval	15	247	581
(reserved)	16		
Test	17	296	600
On-board operations procedure	18	300	602
Event-action	19	322	610
On-board parameter management	20	332	615
Request sequencing	21	339	619
Position-based scheduling	22	352	626
File management	23	386	639
File transfer	24	411	651
File data storage	25	436	663
Parameter extra32ction	26	459	672
Critical packet log management	27	464	674

Bibliography

CCSDS A30.0-G-3	CCSDS Glossary, July 1997
CCSDS 102.0-B-5	Packet Telemetry, Issue 5, November 2000
CCSDS 200.0-G-6	Telecommand Summary of Concept and Rationale, January 1987
CCSDS 202.0-B-3	Telecommand Part 2 – Data Routing Service, Issue 3, June 2001
CCSDS 133.0-B-2	CCSDS Space Packet Protocol, Issue 2, June 2020
CCSDS 301.0-B-4	CCSDS Time Code Formats, Issue 4, November 2010
CCSDS 727.0-B-5	CCSDS File Delivery Protocol (CFDP) Recommended Standard, Issue 5, July 2020
CCSDS 720.1-G-4	CCSDS File Delivery Protocol (CFDP) Introduction, Issue 4, May 2021
CCSDS 732.0-B-2	AOS Space Data Link Protocol, issue 2, 23 May 2007
CCSDS 910.2-G-1	Standard Terminology, Conventions and Methodology (TCM) for defining Data Services, November 1994
ECSS-S-ST-00	ECSS system – Description, implementation and general requirements
ECSS-E-ST-50-03	Space engineering – Space data links – Telemetry transfer frame protocol
ECSS-E-ST-50-04	Space engineering – Space data links – Telecommand protocols synchronization and channel coding
ESA PSS-04-151	Telecommand Decoder Specification, Issue 1, September 1993
ESA PSS-07-101	Packet utilization standard, Issue 1, May 1994
ITU-T V.41	Code-Independent Error Control System – Data Communication over the Telephone Network, 1989 (before renaming known as Information Technology - CCITT V.41)
ISO 8473-1:1998	Information Technology - Protocol for Providing the Connectionless-Mode Network Service: Protocol specification second edition
SANA	Space Packet Protocol Application Process Identifier (APID) registry of the Space Assigned Numbers Authority (SANA) , https://sanaregistry.org/r/space_packet_protocol_application_process_id/
ANSI	American National Standard for Information Systems – Coded Character Sets – 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII) INCITS 4-1986[R2017]