

OBOSS System Integration Manual for Reuse

Document No.: *Terma/SPD/OBOSS-III/013*
Date: *09.02.04*
Issue: *1*
Revision: *-*
Prepared by: *Gert Caspersen*
Reviewed by: *Keld Schultz*
Reviewed by:
Authorised by: *Carsten Jørgensen*
Distribution: *Terma*



Document Change Record

Issue	Date	Change
1	04.02.04	Initial Issue

© 2004, Terma A/S

The copyright of this document is vested in Terma A/S. This document may only be reproduced in whole or in part, stored in a retrieval system, transmitted in any form, or by any means electronic, mechanical, photocopying, or otherwise, with the prior permission of Terma A/S.

Table of Contents

1	Introduction	1
1.1	Purpose	1
1.2	Scope	1
2	References	2
2.1	Applicable Documents	2
2.2	Reference Documents	2
3	Abbreviations, Terms, and Definitions	3
3.1	Abbreviations	3
3.2	Terms and Definitions	3
4	Reuse Context	4
4.1	Telemetry & Telecommand Packet Utilization Standard	4
4.2	Command & Data Handler Platform	4
5	Prerequisites for Reuse	6
5.1	System Engineers	6
5.2	Data Handling Software Implementors	6
6	Reuse Process	7
7	Resource Adaptations	9
7.1	Internal PUS Packets	9
7.2	Field Type Constraints	10
8	Platform Adaptations	12
8.1	Satellite Clock	12
8.2	Ground I/F	13
8.3	Device Command Driver	14



9	Mission Adaptations	15
9.1	Application Process IDs	15
9.2	PUS Mission Constants	16
9.3	Parameter IDs & Types	17
9.4	Error Control Field	18
10	PUS Service Adaptations	19
10.1	Telecommand Verification Service	19
10.2	Onboard Traffic Management Service	20
10.3	Onboard Storage & Retrieval Service	21
10.4	Event Reporting Service	22
10.5	Memory Management Service	23
10.6	Device Level Commanding Service	23
10.7	Function Management Service	24
10.8	Onboard Scheduling Service	24
10.9	Onboard Monitoring Service	25
10.10	Housekeeping & Diagnostics Service	26
10.11	Large Data Service	26
10.12	Event/Action Service	28
11	Application Process Generation	29
11.1	Application Process Parameters	29
11.2	Telemetry Router	30
11.2.1	Packet Depositor Routing	30
11.2.2	Large Data Transfer Sending Sub-Service Routing	31
11.3	Application Process Bus I/F	31
11.4	Instantiation of PUS Services	32
11.4.1	Device Level Commanding Instance	33
11.4.2	Housekeeping & Diagnostics Instance	34
11.4.3	Memory Management Instance	35
11.4.4	Function Management Instance	37
11.4.5	Onboard Scheduling Instance	38
11.4.6	Onboard Monitoring Instance	40

11.4.7	Onboard Storage & Retrieval Instance	42
11.4.8	Large Data Transfer Instance	44
11.4.9	Event Action Instance	50
11.5	Telecommand Dispatcher	51
12	Main Program	53



1 Introduction

1.1 Purpose

This technical note is intended as a road map through the process of reusing the collection of software components developed as part of the 'OBOSS System Integration Frame Contract'. These components are intended to serve as a baseline for production of a new piece of data handling system software resulting in substantial savings with respect to time and money.

The target group of this note is systems engineers and programmers working on a satellite data handling system. A vast majority of the information feeding into the reuse process is bound to come from system engineers associated to the data handling system while specific adaptations have to be carried out by the software team responsible for the data handling software.

1.2 Scope

The current document was produced as part of Call-Off Order 1 on the 'OBOSS System Integration Frame Contract'. The product to which this manual for reuse applies is an updated release — referred to as OBOSS-III — of the 'Onboard Operations Support Software' (OBOSS).

2 References

2.1 Applicable Documents

Reference	Document
[PTC]	<i>Packet Telecommand Standard</i> ESA PSS-04-107, Issue 2
[PTM]	<i>Packet Telemetry Standard</i> ESA PSS-04-106, Issue 1
[PUS]	<i>Ground System & Operations — Telemetry & Telecommand Packet Utilization</i> ECSS-E-70-41A, January 2003

2.2 Reference Documents

Reference	Document
[OM]	<i>OBOSS-III Operations Manual</i> TERMA/SPD/OBOSS-III/012, Issue 1
[HOOD]	<i>HRT-HOOD: A Structured Design Method for Hard Real-Time Ada Systems</i> Alan Burns and Andy Wellings, Elsevier, 1995
[ECSS-E-40]	<i>Space Engineering — Software</i> ECSS-E-40B, Draft July 2000
[ADA]	<i>Annotated Ada Reference Manual</i> ISO/IEC 8652, Version 6

3 Abbreviations, Terms, and Definitions

3.1 Abbreviations

PUS Telemetry & Telecommand Packet Utilization standard

3.2 Terms and Definitions

None.

4 Reuse Context

The goal of the project was to develop a reusable onboard software architecture implementing a subset of the services defined in 'Telemetry & Telecommand Packet Utilization' [PUS] standard on a command and data handling platform. This architecture shall be able to support and ease (through intensive reuse) development of onboard command and data handling software for a series of future missions.

Consequently, the context of the project is dominated by three elements: The telemetry & telecommand packet utilization standard, the satellite platforms on which the data handling system architecture is to build, and the reusable software components being part of the reusable architecture. Each of these is discussed in the following.

4.1 Telemetry & Telecommand Packet Utilization Standard

This standard defines several general services whose onboard implementation supports satellite operation. It defines the application level of Packet Telecommand Standard [PTC] and Packet Telemetry Standard [PTM].

It is not within the scope of the project to develop a complete implementation of the telemetry & telecommand packet utilization standard. Consequently, a subset of the services and sub-services have been selected for implementation.

4.2 Command & Data Handler Platform

The architecture will be affected by assumptions made regarding the command and data handler platform on which it is to reside. This will define the environment of the command and data handler. The environment depicted in Figure 1 is considered as the baseline. Each element is briefly described below.

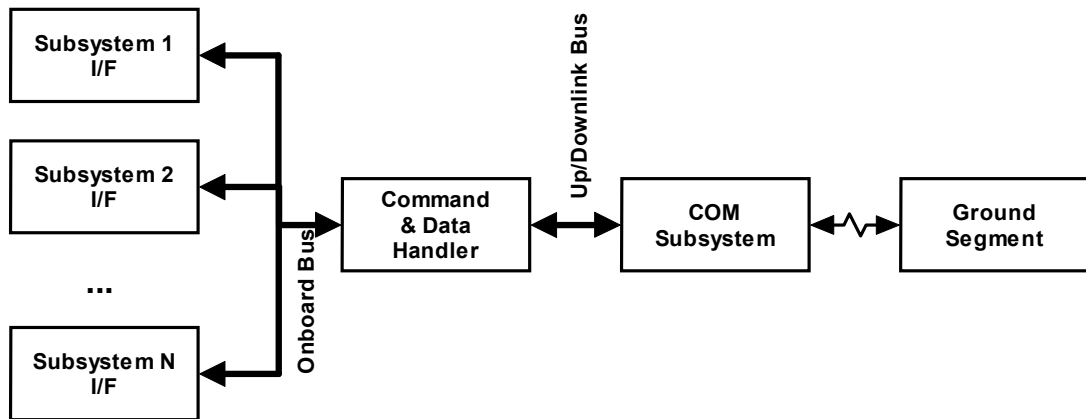


Figure 1: Command & Data Handler Platform

Ground Segment

Control centre, ground stations etc. responsible for operation of the satellite on which the command and data handler platform resides.

COM Subsystem

Communication subsystem providing the onboard support for the radio link between space segment and ground segment. It implements several layers in the ESA Packet Telecommand Standard [PTC] and Packet Telemetry Standard [PTM].

Up/Downlink Bus

Onboard bus dedicated to uplink and downlink data flows. May be implemented by use of an ESA 4-255 data bus.

Command & Data Handler

Onboard subsystem responsible for commanding of onboard subsystems and data collection from these.

Onboard Bus

The physical layer of the platform specific onboard bus.

Subsystem 1 I/F .. Subsystem N I/F

Collection of onboard subsystems controlled by the command & data handler. Includes payloads as well as attitude control subsystem, electrical power subsystem etc.

5 Prerequisites for Reuse

To succeed in reusing the reusable components, several prerequisites must be fulfilled by the reusing organisation. These have been partitioned into prerequisites for system engineers and prerequisites for data handling system programmers.

5.1 System Engineers

All system engineers participating in the reuse process shall at least possess the following skills:

- Extensive knowledge of ESA's 'Telemetry & Telecommand Packet Utilization Standard' [PUS].
- Thorough knowledge of all interfaces and subsystems to be controlled by the resulting data handling system software.
- Superficial knowledge of 'OBOSS-III Operations Manual' [OM].

5.2 Data Handling Software Implementors

Members of the team responsible for implementation of the data handling software shall at least possess the following skills:

- Extensive knowledge of ESA's 'Telemetry & Telecommand Packet Utilization Standard' [PUS].
- Thorough knowledge Hard Real-Time HOOD [HOOD].
- Profound knowledge of the Ada 95 programming language [ADA].
- Acquaintance with the ECSS 'Space Engineering — Software' standard [ECSS-E-40].
- Thorough knowledge of 'OBOSS-III Operations Manual' [OM].

6 Reuse Process

Reusing the software consists of several adaptation steps. Each major step generally consists of two consecutive phases:

1. Acquisition of information serving as input for the adaptation. This is generally performed by the systems engineering group. In the presentation of the adaptation steps this will be represented as a collection of *Input* entries.
2. Adaptation or instantiation of the software component by the data handling software team. In the presentation of the adaptation steps this will be represented as a collection of *Adaptation Step* entries with an identification of the *Ada Package* serving as baseline for the adaptation.

The following chapters present several work flow charts for the adaptation process. Some adaptation steps may be performed in parallel. This has been indicated in the diagrams by placing these as siblings in the graph.

The major phases of the reuse process are depicted in Figure 2 below.

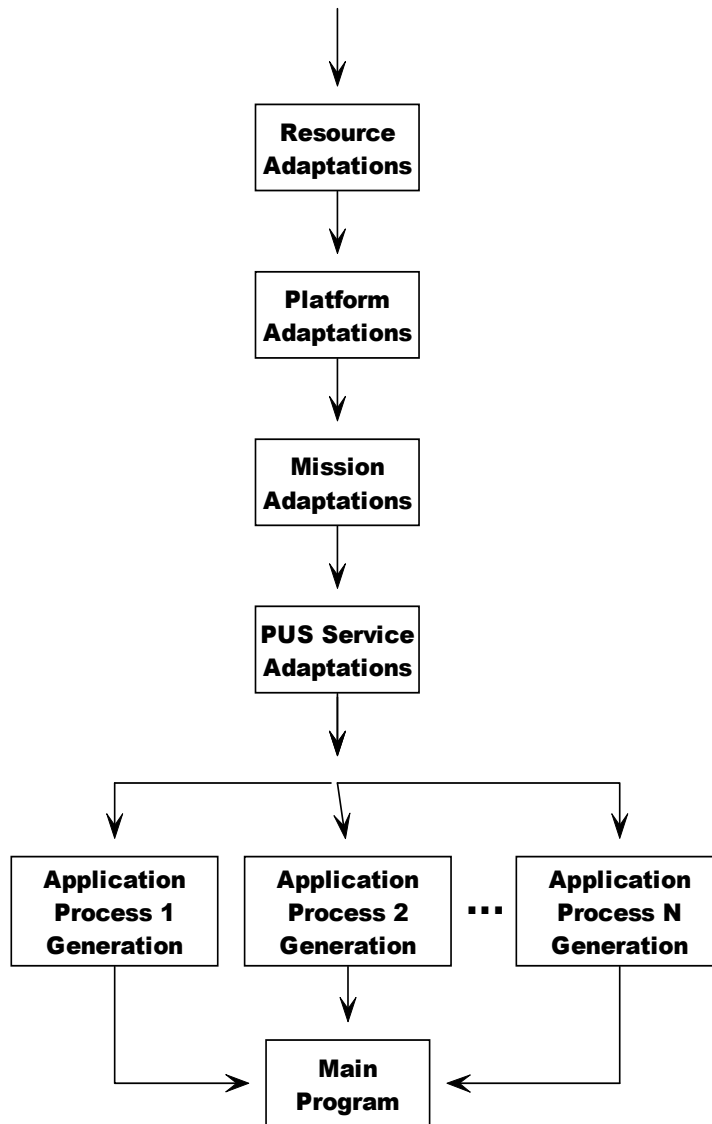


Figure 2: Overall Reuse Process

Each phase will be elaborated in the following chapters.

7 Resource Adaptations

A number of parameters control the amount of resources (e.g. buffers or Telemetry & Telecommand Packet Utilization standard packets) allocated in the system. Based on knowledge of the operational requirements for the mission these have to be tailored through the following steps:

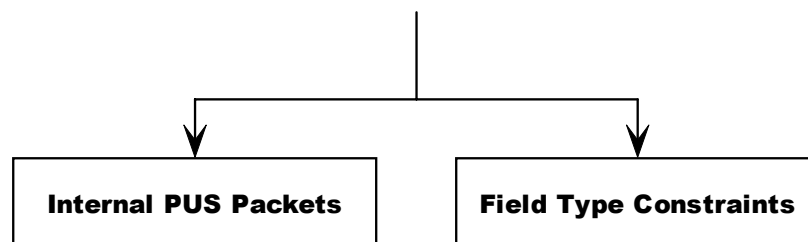


Figure 3: Resource Adaptations

7.1 Internal PUS Packets

Interpretation	
Internally in the system telecommands <i>and</i> telemetry packets are transformed into an internal format using a pool of PUS packet ‘containers’.	
Data in PUS telecommands and telemetry that is not part of Packet Header, Data Field Header or Packet Error Control are stored in buffers referred to as ‘Source Data Streams’.	
Two buffer pools exist: A pool of large buffers and a pool of small buffers.	
Input	
1	Size of large source data streams <i>in bits</i> .
2	Size of small source data streams <i>in bits</i> .
3	Number of large source data streams.
4	Number of small source data streams.
5	Number of PUS packet containers.
Considerations	

Input 1 should specify a size that is able to accommodate the largest possible *service data units* (SDUs) received or generated by instances of the large data transfer service.

Input 1 should specify a size that is able to accommodate the largest possible telecommand application data or telemetry source data applicable to the mission. These may be derived from the mission constants `<TCPKT_MAX_LENGTH>` and `<TMPKT_MAX_LENGTH>` (see section 9.2).

Input 3 and 4 should be sufficiently large to cater for the maximum number of telecommands *and* telemetry that may reside in the system simultaneously.

Beware: Time tagged telecommands awaiting execution in an onboard scheduling service occupy a PUS packet container *and* a source data stream until released and executed.

Beware: Telecommands associated to an event in an event/action service detection occupy a PUS packet container *and* a source data stream until it is explicitly deleted from the detection list.

The total number of source data streams (input 3 + input 4) shall exceed the number of PUS packets (input 5) by approximately 10%.

<i>Adaptation Step</i>	<i>Ada Package</i>
Update small source data size and small source data number according to 2 & 4.	Storage Configuration
Update large source data size and large source data number according to 1 & 3.	Storage Configuration
Update PUS packet number according to 5.	Storage Configuration

7.2 *Field Type Constraints*

<i>Interpretation</i>	
A number of field types and structures are defined in the Telemetry & Telecommand Packet Utilization standard (chapter 23).	
These may be of unlimited size. However, placing an upper limit on their size to avoid excessive memory usage is necessary. A collection of parameters exists limiting the maximum length of various data fields on a <i>system wide level</i> (i.e. applying to all telecommands and telemetry for a mission).	
<i>Input</i>	
1	Maximum length of bit string, octet string, and character string parameters (parameter type codes 6, 7 and 8).
2	Size – in bits – of string length field for variable length bit strings, octet strings and character strings (parameter codes (6,0), (7,0) and (8,0)).
3	Maximum number of packet service subtypes used by the total set of application processes using an onboard storage and retrieval service.
<i>Considerations</i>	
Input 1 has a major impact on the heap and stack usage in the system. Any parameter value to be read or stored takes the type of the union of all possible parameter types, including the potentially large string types.	
<i>Adaptation Step</i>	<i>Ada Package</i>

Modify Max Bit String Length, Max Octet String Length, and Max Character String Length according to input 1.	Parameter Representation Constraints
Modify Size of String Length Field according to input 2.	Parameter Representation Constraints
Modify Max Packet Store Subtype Length according to input 3.	Parameter Representation Constraints

8 Platform Adaptations

Parts of the system is bound to be platform specific, i.e. relying on characteristics of the underlying data handling system hardware. The following adaptations have to be carried out to cater for this:

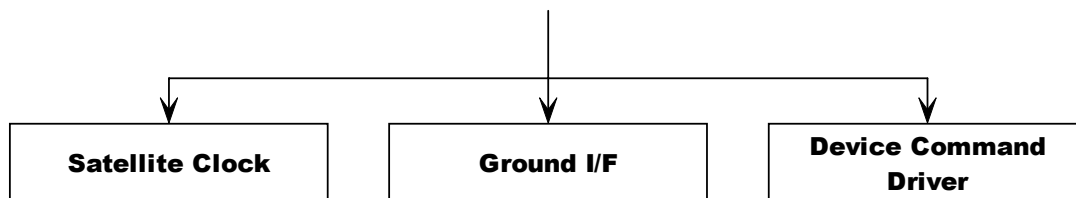
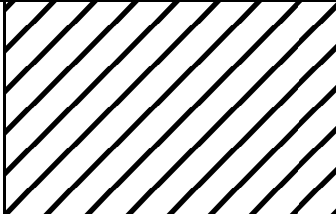


Figure 4: Platform Adaptations

8.1 Satellite Clock

<i>Interpretation</i>	
A global onboard time reference has to be made available — e.g. for insertion of time stamps into telemetry packets. This has to provide the current onboard time according to one common satellite clock device.	
<i>Input</i>	
1	Number of bytes used to represent the onboard time.
2	Frequency of external clock device.
3	Information on how to access external clock device.
<i>Considerations</i>	
None.	
<i>Adaptation Step</i>	<i>Ada Package</i>
Update Onboard Time (OBT) Byte Size and External Clock Frequency according to input 1 & 2.	External Onboard Clock
Implement device driver accessing external onboard clock device through provision of a body for compilation unit External Onboard Clock.	External Onboard Clock

8.2 Ground I/F

Interpretation	
<p>Ground I/F is a special application process responsible for the telecommand packet flow from the ground into the system and the telemetry packet flow from the system down to the ground.</p> <p>This component normally interfaces to the receiver and transmitter and implements any protocol layers situated below the packetisation layer of ESA's 'Packet Telecommand Standard' [PTC] and 'Packet Telemetry Standard' [PTM].</p>	
Input	
1	Knowledge of protocol layers situated below the packetisation layer (e.g. segmentation layer, transfer layer).
2	Knowledge of the interface to the receiver and transmitter including bus type and information on the controller used to access the bus.
3	Knowledge of any protocol applied on the transmitter/receiver interface.
4	Decide on error handling in case of reception of telecommand packets not complying to the mission specific interpretation of the standard.
Considerations	
<p>It is preferred if the protocol and bus controller used on the receiver/transmitter bus are chosen to be event driver – e.g. based on interrupts – on the uplink. Ground I/F is to have a sporadic behaviour regarding reception of telecommands. When a complete telecommand packet is received, it is to be injected into the system immediately.</p>	
Adaptation Step	Ada Package
<p>Provide a body for Forward Packet that transmits a given telemetry packet to the transmitter using any protocols applying to the interface (re. Input 1, 2 & 3).</p> <p>Transformations from internal PUS packet format to a raw string of bytes are to be based on the transformations provided by External PUS State package (see. Up Down Link Parameters for inspiration).</p>	Ground IF
<p>Provide a body for Receive that transforms a given string of bytes into the internal representation of PUS telecommand packets (re. Input 1, 2, 3 & 4).</p> <p>Transformations from a raw string of bytes to internal PUS packet format are to be based on the transformations provided by External PUS State package (see. Up Down Link Parameters for inspiration).</p>	Ground IF
<p>Write a 'device driver' based on input 2 & 3 receiving and transmitting byte strings on the transmitter/receiver bus.</p> <p>Reception of telecommand packets shall be event driven – e.g. based on interrupts – on the arrival of a block of bytes corresponding to a telecommand packet. When this event occurs, Ground IF.Receive shall be called with the byte string.</p>	

8.3 Device Command Driver

Interpretation	
<p>A 'device driver' has to provide access to platform registers and/or onboard busses to support device level commanding at the lowest level.</p> <p>This is intended as a 'backdoor' for simple commanding of onboard devices – e.g. in case of contingencies.</p> <p>The external representation of fields in device command service requests and service reports have to be defined.</p>	
Input	
1	List of devices to be commanded by on/off commands.
2	Definition of how on/off commands shall be distributed to devices from input 1.
3	List of devices to be commanded by register load commands.
4	Definition of how register load commands shall be distributed to devices from input 3.
5	Assignment of unique Device Addresses to devices from input 1 and 3.
6	Application ID of application process providing the device command distribution service.
Considerations	
<p><i>Note: Command pulse distribution unit (CPDU) commands are not supported in the current implementation.</i></p>	
Adaptation Step	Ada Package
Implement a Send operation for On Off Driver submitting a given on/off command to a device identified by a device address (re. Input 1, 2 & 5).	On Off Driver
Implement a Send operation for Register Driver submitting a given register load command to a device identified by a device address (re. Input 3, 4 & 5).	Register Driver

9 Mission Adaptations

These adaptations affect the entire data handling system. They are primarily concerned with allocation of identifiers at system level and the overall operational characteristics of the resulting data handling system.

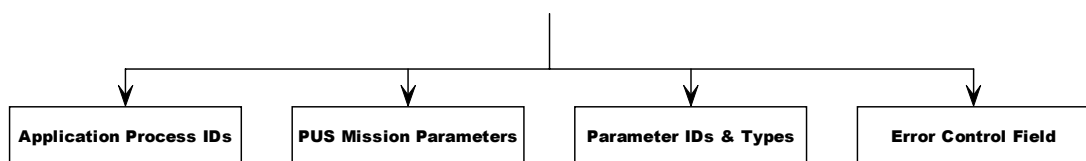


Figure 5: Mission Adaptations

9.1 Application Process IDs

Interpretation	
Each application process to be implemented by the data handling system <i>or</i> to be interfaced by the data handling system (if it is an external onboard subsystem) shall be assigned a unique application process ID.	
Input	
1	A complete list of onboard application processes.
2	Assignment of unique application process IDs (integer constants) to application processes in input 1.
3	Selection of external representation of application process IDs when these occur in source data.
4	Selection of a subset of application process IDs used in telecommand Source ID field.
5	Selection of parameter format code for telecommand Source ID field.
6	Selection of a subset of application process IDs used in telemetry Destination ID field.
7	Selection of parameter format code for telemetry Destination ID field.
Considerations	
<i>Note:</i> Two application process IDs are reserved: Time Packets having value 0 and Idle Packets having value 2047 (7FF _{hex}).	
Adaptation Step	Ada Package

Update APID enumeration type and the associated representation clause according to input 1 and 2.	Mission Parameters
Update type definition for External APID according to input 3.	Mission Parameters
Update Get and Put operations for application process IDs transforming these between external and internal representations. Involves adding one alternative in case statements for each application process ID from input 2. See existing implementation for inspiration. <i>Note:</i> Get operation shall raise exception Invalid Data APID if the external representation of a read application process ID does not match any of the internally defined application process IDs.	Mission Parameters
Update type definition for Source ID according to input 4.	Mission Parameters
Update type definition for External Source ID according to input 5.	Mission Parameters
Update type definition for Destination ID according to input 6.	Mission Parameters
Update type definition for External Destination ID according to input 7.	Mission Parameters

9.2 PUS Mission Constants

Interpretation	
<p>Appendix B of the Telemetry & Telecommand Packet Utilization standard defines a range of telemetry and telecommand related parameters that may vary from mission to mission. A subset of these is applicable to the data handling software being produced here.</p> <p>The set has been extended with a collection of mission specific parameters related to the implementation of Telemetry & Telecommand Packet Utilization standard services provided (especially controlling the dynamic behaviour of these).</p>	
Input	
1	<p>System wide values to the below parameters from Appendix B: 'Mission Constants' of the Telemetry & Telecommand Packet Utilization standard.</p> <ul style="list-style-type: none"> ● DIAG_MIN_INTERV ● NUM_SOURCE_BITS ● NUM_SUB_SCHEDULES ● TCPKT_MAX_LENGTH ● TMPKT_MAX_LENGTH ● TC_CHECKSUM_TYPE ● TM_CHECKSUM_TYPE ● SMALLEST_ADDRESSABLE_UNIT
2	A system-wide time representation used for all time-stamps onboard.
3	Period by which <i>all</i> instances of an onboard monitoring service generate out-of-limit reports.

Considerations	
<p>For input 2, only the CUC time format (re. [PTM]) is currently supported.</p> <p>Re. Input 3: Note that this value applies to all onboard monitoring services. The value should thus be the minimum of the required report generation intervals.</p>	
Adaptation Step	Ada Package
Adapt Diag Min Interval, Num Source Bits, Num Sub Schedules, Tc-pkt Max Length, Tmpkt Max Length, TC Checksum Type, TM Checksum Type, and Smallest Addressable Unit according to input 1.	Mission Parameters
Adapt Canonical Time Rep Spec according to input 2.	Mission Parameters
Adapt Max Monitoring Reporting Delay according to input 3.	Mission Parameters

9.3 *Parameter IDs & Types*

Interpretation	
<p>Every onboard parameter for which parameter values are to be collected by an application process is to be assigned a unique ID and some type information specifying its type and representation.</p> <p><i>This information applies at the system level, meaning that it is common to all application processes.</i></p>	
Input	
1	Selection of external format for parameter IDs – also referred to as parameter numbers in the standard – when these occur in telecommand or telemetry packets.
2	Complete list of onboard parameters for which values are to be collected by application processes.
3	List of parameter IDs assigning unique parameter IDs to each parameter in input 2.
4	List of Parameter Codes ¹ defining type and representation to each parameter ID from input 3.
Considerations	
<p>During early definition phases it may be advantageous to partition the set of available parameter IDs into disjoint partitions, assigning one partition to each onboard application process.</p> <p>List of parameter IDs may be non-contiguous (i.e. containing holes with unused parameter IDs).</p>	
Adaptation Step	Ada Package
Update External Parameter ID type definition according to input 1.	Mission Parameters

¹ Chapter 23 ‘Parameter Types and Structure Rules’ defines Parameter Code based on available types and representations in the Telemetry & Telecommand Packet Utilization standard [PUS].

Update Put and Get operations to write and read parameter IDs on the external format defined by input 1. See existing implementation for inspiration.	Parameter Structure Descriptions
Update Parameter Is Defined predicate to return true for every parameter ID in input 3.	Parameter Structure Descriptions
Update Get Parameter Code to map any defined parameter ID into its Parameter Code as given by input 4. See existing implementation for inspiration.	Parameter Structure Descriptions

9.4 **Error Control Field**

Interpretation	
Interpretation of the optional Packet Error Control field may vary from mission to mission and may even differ between telecommand packets and telemetry packets. Size, checksum algorithm and other characteristics shall thus be defined.	
Input	
1	Specification of presence or absence of packet error control field in telecommand packets. If it is absent, step 2 below may be skipped.
2	Checksum algorithm used to calculate packet error control field for telecommand packets.
3	Specification of presence or absence of packet error control field in telemetry packets. If it is absent, step 4 below may be skipped.
4	Checksum algorithm used to calculate packet error control field for telemetry packets.
Considerations	
The Telemetry & Telecommand Packet Utilization standard prescribes for packet error control fields to be present in telecommand packets. Only ISO checksum is supported by the current implementation.	
Adaptation Step	Ada Package
Update TC Checksum Type according to input 2 if input 1 prescribes the presence of such a field.	Mission Parameters
Update TM according to input 4 if input 3 prescribes the presence of such a field.	Mission Parameters

10 PUS Service Adaptations

Although a PUS service may be provided by a number of application processes, some characteristics associated to each service type has to be shared at the system level. An adaptation step may be skipped if the data handling software does not contain *any* instances of this service – normally implying that none of the onboard application processes provides the service in question.

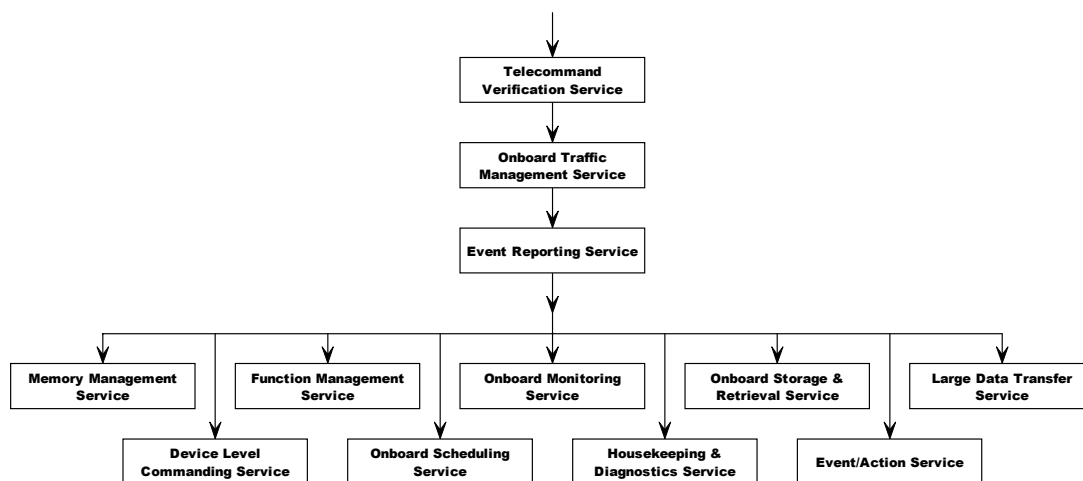


Figure 6: PUS Service Adaptations

10.1 Telecommand Verification Service

The following adaptations apply to service type 1 in the Telemetry & Telecommand Packet Utilization standard.

This step *is not* optional as all other services from the Telemetry & Telecommand Packet Utilization standard use the telecommand verification service.

Interpretation
Representation and interpretation of packet fields associated to the service have to be adapted on a mission by mission basis.
<i>Note that adaptations apply to all application processes providing a telecommand verification service - which in reality covers them all.</i>
Input

1	Parameter format codes for Step Number and Code fields included in telecommand verification packets.
Considerations	
A collection of system level verification codes is predefined and used by the other supported services from the Telemetry & Telecommand Packet Utilization standard to indicate cause of failing telecommand verifications. See Mission Verification Values. <i>These are reserved and shall be supported.</i>	
Adaptation Step	Ada Package
Update Verification Code and Verification Step Number type derivations according to input 1.	External Telecommand Verification Types

10.2 Onboard Traffic Management Service

The overall telecommand and telemetry packet flows are implemented by this service.

This step *is not* optional as all other services from the Telemetry & Telecommand Packet Utilization standard use the service.

Interpretation	
A sporadic object is responsible for onboard routing of telecommands and telemetry. Telecommands and telemetry submitted for routing are forwarded to the destination application process.	
A special case applies if one or more global instances of the event/action service are included. These should receive copies of event reports being routed.	
Input	
1	Length of the queue for PUS packets awaiting routing.
2	Priority of the protected object implementing PUS Packet Queue.
3	Application process ID for Packet Router.
4	Priority and stack size of sporadic task forwarding packets.
5	Application process ID and Ada package name for all packages implementing an application process (<i>remember</i> Ground Interface process).
6	Identification of application processes providing (global) instances of an event/action service.
Considerations	
Input 1 shall be large enough to cater for potential bursts in the generation of telecommands and telemetry for the entire collection of onboard application processes.	
Input 2 shall exceed priority of any task submitting packets for routing.	
Adaptation Step	Ada Package
Update Application ID (re. 3).	Packet Router Parameters

Update Packet buffer size & priority (re. 1 & 2).	Packet Router Parameters
Update Packet Distributer Stack Size & Priority (re 4).	Packet Router Parameters
For each (<i>APID</i> , <i>Package</i>) identified in 5: Add a with- clause and insert a branch in Distribute PUS Packet mapping <i>APID</i> into <i>Package.Forward TC</i> , <i>Package.Forward TM</i> or <i>Package.Forward Packet</i> as appropriate.	Packet Router
For each application process identified in 6, add a branch in Distribute PUS Packet forwarding a <i>shared copy</i> of any telemetry event reports.	Packet Router

10.3 Onboard Storage & Retrieval Service

The following adaptations apply to service type 15 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation	
Representation and interpretation of fields related to the onboard storage and retrieval service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for list count fields ² , Downlink Set Type fields, Deletion Set fields, and Source Sequence Count fields included in telecommand and telemetry packets for onboard storage & retrieval service.
2	Set of Packet Store IDs used by any application process providing the storage and retrieval service.
3	Length of string constituting the Store ID field in telecommand and telemetry packets.
4	Partitioning of packet stores (and their Packet Store IDs) among the application processes providing the Storage and Retrieval sub-service; each packet store must belong to one and only one such application process.
Considerations	
The order of the Packet Store IDs in the type definition is important if several application processes provide the Storage and Retrieval sub-service: The identifiers must then be given in an order that allows subtypes to be defined for the Packet Store IDs supported by the individual application processes providing the service.	
Adaptation Step	Ada Package
Update derived type definitions corresponding to input 1.	External Packet Store Types
Define the type Packet Store ID corresponding to input 2.	Mission Parameters

² : These indicate length of enclosed lists and are referred to as N, N1, N2 etc.

Update Packet Store ID Length according to input 3.	Mission Parameters
Define subtypes for Packet Store IDs supported by each application process providing the service: subtype <application process> Store ID is Packet Store ID range <lower bound> .. <upper bound> corresponding to input 4.	Mission Parameters
Adapt Packet Store Association State mapping store identifiers into the application processes administering it, corresponding to input 4.	Storage Selection Manager

10.4 Event Reporting Service

The following adaptations apply to service type 5 in the Telemetry & Telecommand Packet Utilization standard.

This step *is not* optional as all other services from the Telemetry & Telecommand Packet Utilization standard uses the event reporting service.

Interpretation	
Representation and interpretation of fields related to the event reporting service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for Report ID field included in event reporting packets.
2	List of Report IDs included in any mission specific event reports.
3	Map from Report IDs into parameter codes for fields contained in such event reports.
4	
Considerations	
Note that adaptations <i>apply to all application processes</i> using the event reporting service – which in the current version is the entire set of services.	
Note that Report IDs 0 to 11 are reserved (see ‘OBOSS-III Operations Manual’ [OM]).	
Adaptation Step	Ada Package
Update derived type declaration of Report ID according to input 1.	External Event Reporting Types
Update internal representation of Report according to map from report IDs to fields given by input 2 and input 3.	Event Reporting Types
Update Put operation to transform event reports given by input 2 and 3 to their external representation (see current implementation for inspiration).	External Event Reporting Types

10.5 Memory Management Service

The following adaptations apply to service type 6 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation	
Representation and interpretation of fields related to the memory management service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for Start Address, N and Length field included in telecommand and telemetry packets related to absolute addresses (service subtypes 2, 5, 6, and 9).
2	Length of Memory ID field – being a fixed size character string – in telecommand and telemetry packets related to absolute addresses (service subtypes 2, 5, 6, and 9).
Considerations	
Adaptations apply to all application processes providing a memory management service. Current implementation only support absolute addresses i.e. service subtypes 2, 5, 6, and 9.	
Adaptation Step	Ada Package
Update derived type declaration of Start Address, Length and List Length (being N) according to input 1.	External Memory Management Types
Update constant declaration of Memory ID Length according to input 2.	External Memory Management Types

10.6 Device Level Commanding Service

The following adaptations apply to service type 2 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation	
Representation and interpretation of fields related to the device level commanding service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for N, Address ³ , and Register Data fields in telecommand packets.
2	An upper limit for the length field N indicating number of device commands in one telecommand packet.

³ : This format will apply to addresses in on/off commands *as well as* to register addresses.

Considerations	
Note that in the current implementation only one instance of this service may exist onboard.	
Adaptation Step	Ada Package
Update derived types for List Length, Device Addr, and Register Data according to input 1.	External Device Command Distribution Types
Update declaration of Max No Of Commands according to input 2.	External Device Command Distribution Types

10.7 Function Management Service

The following adaptations apply to service type 8 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation	
Representation and interpretation of fields related to the function management service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for Function ID and List Length (N) fields in telecommand packets.
Considerations	
Note that adaptations apply to <i>all application processes</i> providing a function management service.	
Note that the external format for Parameter# field is globally defined in Mission Parameters, see section 9.3.	
Adaptation Step	Ada Package
Update derived type for Activity ID according to input 1.	External Function Management Types
Update Function ID Length definition according to input 1.	External Function Management Types

10.8 Onboard Scheduling Service

The following adaptations apply to service type 11 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation

Representation and interpretation of fields related to the onboard scheduling service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for Sub-schedule ID, List Length ⁴ , Scheduling Event, Sequence Count, and Range fields in telecommand packets and telemetry packets.
Considerations	
Adaptations apply to <i>all application processes</i> providing an onboard scheduling service. Current implementation <i>does not</i> support interlocking.	
Adaptation Step	Ada Package
Update derived types for Sub Schedule ID, Scheduling Event Spec, Sequence Count, Scheduling Range, and List Length according to input 1.	External On Board Scheduling Types

10.9 Onboard Monitoring Service

The following adaptations apply to service type 12 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation	
Representation and interpretation of fields related to the onboard monitoring service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for Check Position ,Check Selection Parameter#, Checking Status ⁵ , Parameter Monitoring Interval, List Lengths ⁶ , Monitoring Status, Validity Parameter#, and Value #REP fields in telecommand packets and telemetry packets.
2	A single application process ID that will be the destination for telemetry reports generated by all onboard monitoring services.
Considerations	
Adaptations apply to <i>all application processes</i> providing an onboard monitoring service. Input 2 will normally be the application process ID associated to the Ground application process.	
Adaptation Step	Ada Package

⁴ : This covers fields named N, N1, N2, and Number of Telecommands.

⁵ : This covers Previous Checking Status and Current Checking Status fields.

⁶ : This covers all fields named N, NOL, NOE etc.



Update derived type definitions for List Length, Parameter Number, Interval, Checking Status, Value Filter, Validity Parameter Number, Check Selection Parameter Number, Monitoring Status, and Check Position according to input 1.	External Onboard Monitoring Types
Update definition of TM Destination according to input 2.	External Onboard Monitoring Types

10.10 Housekeeping & Diagnostics Service

The following adaptations apply to service type 3 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation	
Representation and interpretation of fields related to the housekeeping & diagnostics service may vary on a mission by mission basis.	
Input	
1	Parameter format codes for List Lengths ⁷ , Collection Interval, Mode, SID, Threshold Type, Threshold, and Timeout fields in telecommand packets and telemetry packets.
Considerations	
Adaptations apply to <i>all application processes</i> providing a housekeeping & diagnostics service.	
Adaptation Step	Ada Package
Update derived type definitions for Count Type, Timeout Type, Threshold, Threshold Type, Interval Type, Mode, and Structure ID according to input 1.	External HK Collector Types

10.11 Large Data Service

The following adaptations apply to service type 13 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation
Representation and interpretation of fields related to the large data transfer service may vary on a mission by mission basis.
Input

⁷ : This covers N, NPAR, NPAR1, NPAR2, NREP, NSID fields.

1	<p>Parameter format codes for Sequence Number, List Length, and Reason Code.</p> <p>Sequence Number is used for representation of field 'Sequence Number' in:</p> <ul style="list-style-type: none"> ● The sending sub-service, service type (13,1), (13,2), (13,3), (13,7), (13,5), (13,6). ● The receiving sub-service, service type (13,9), (13,10), (13,11), (13,12), (13,14), (13,15) <p>List Length is used for representation of field 'N' in:</p> <ul style="list-style-type: none"> ● The sending sub-service, service type (13,6) ● The receiving sub-service, service type (13,15) <p>Reason Code is used for representation of field 'Reason Code' in:</p> <ul style="list-style-type: none"> ● The sending sub-service, service type (13,16) ● The receiving sub-service, service type (13,4)
2	<p>Reason code values for Reason Code.</p> <p>For the sending sub-service:</p> <ol style="list-style-type: none"> 1: Wrong Sequence Number In Acknowledge 2: Reception Acknowledge Timeout 3: Part Stream Allocation Failed 4: Part Packet Allocation Failed 5: Part Packet Deposit Failed 6: Illegal Part From Sender State 7: Sender Logic Error <p>For the receiving sub-service:</p> <ol style="list-style-type: none"> 8: Timeout Waiting For Part 9: Illegal Command Waiting For Part 10: Illegal Part Sequence Number 11: Repeated Part Errorneous 12: Receiver Logic Error
3	<p>Octet string length of a service data unit (SDU) part. This size applies for all SDU part downlink telemetry and SDU part uplink telecommands, for all large data transfer services.</p>
4	<p>Event Report ID's for event reports generated by the large data transfer service:</p> <ol style="list-style-type: none"> 1: Uninstantiated Sending Subservice 2: SDU Conversion Failure Report 3: Current Uplink Aborted
Considerations	
Adaptations apply to <i>all application processes</i> providing a large data transfer instance.	
Adaptation Step	Ada Package
Update derived typed definitions for List Length, Sequence Number, and Reason Code according to input 1.	External Large Data Transfer Types
Update the Reason Code constants values according to input 2.	External Large Data Transfer Types



Update the constant SDU Part Byte Size according to input 3.	External Large Data Transfer Types
Update the constant Event Report IDs according to input 4.	Event Reporting Types

10.12 Event/Action Service

The following adaptations apply to service type 19 in the Telemetry & Telecommand Packet Utilization standard.

This step may be skipped if no onboard application processes provide this service.

Interpretation	
Representation and interpretation of fields related to the event/action service may vary on a mission by mission basis.	
Input	
1	<p>Parameter format codes for List Length, APID, RID, and Action Status.</p> <p>List Length is used for representation of field 'N' in service subtype (19,1), (19,2), (19,4), (19,5), (19,7)</p> <p>APID is used for representation of field 'Application Process ID' in service subtype (19,1), (19,2), (19,4), (19,5), (19,7)</p> <p>RID is used for representation of field 'RID' in service subtype (19,1), (19,2), (19,4), (19,5), (19,7)</p> <p>Action Status is used for representation of field 'Action Status' in service (19,7)</p>
2	<p>Event Report IDs for event reports generated by the event action service:</p> <p>1: Unsupported Telemetry Packet</p>
Considerations	
<p>Adaptations apply to <i>all application processes</i> providing an event action instance.</p> <p>When an enabled action is activated by an event, the action TC will have its sequence number updated before it is executed.</p>	
Adaptation Step	Ada Package
Update derived typed definitions for List Length, APID, RID, and Action Status according to input 1.	External Event Action Types
Update the constant Event Report IDs according to input 2.	Event Reporting Types

11 Application Process Generation

The steps below have to be repeated once for each onboard application process that is to provide instances of Telemetry & Telecommand Packet Utilization standard services based on the reusable software components.

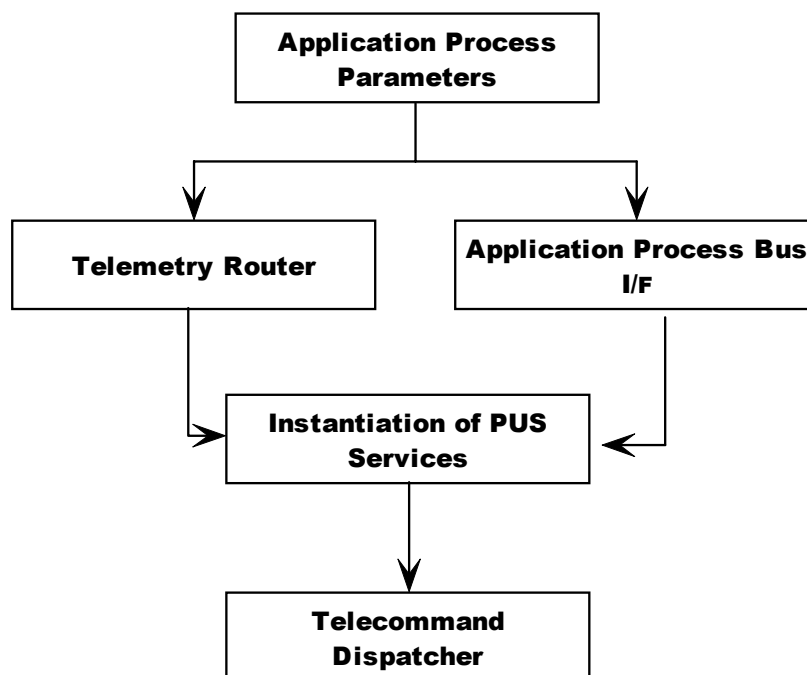


Figure 7: Application Process Generation

11.1 Application Process Parameters

This step shall be performed once for each new application process added to the system.

Interpretation
Some parameters from Annex B of the Telemetry & Telecommand Packet Utilization standard vary from application process to application process. A subset of these is used by the current implementation and has to be adapted for each new application process.
Input

1	<p>Selection of APPL TIME CODE mission parameter in appendix B of the Telemetry & Telecommand Packet Utilization standard:</p> <ul style="list-style-type: none"> ● Presence or absence of time fields. ● CUC or CDS format, if time fields are present. ● Parameter format code for time fields, if present. <p>This applies to time stamps enclosed in telemetry packets generated by the given application process.</p>
2	<p>Selection of a queue length for the queue of telecommand packets and telemetry packets at the interface to the application process. This queue buffers up any bursts in arrival of packets destined for the given application process.</p>
Considerations	
<p>Input 2 should be sufficiently large to cater for bursty arrival of telecommands from ground as well as arrival of telecommand packets or telemetry packets from other onboard application processes. If the queue is full arriving packets are discarded without notice.</p> <p>Current implementation <i>does not</i> support absence of time fields.</p> <p>Current implementation <i>only</i> supports CUC time format.</p>	
Adaptation Step	
<p>Add a new entry to The Parameters mapping the application process ID assigned to this application process into a collection of Application Process Mission Parameters corresponding to input 1 and input 2.</p>	
Ada Package	
<p>Application Process Parameters</p>	

11.2 Telemetry Router

This step shall be performed once for each new application process added to the system.

11.2.1 Packet Depositor Routing

Interpretation	
<p>A sequential sequence numbering shall be maintained on a stream of telemetry packets originating from a range of Telemetry & Telecommand Packet Utilization standard services provided by the application process. This router is to update sequence numbers and pass the packet on for routing.</p>	
Input	
1	<p>Indication of how telecommand packets and telemetry packets generated by the application process shall be routed.</p> <p>These may be routed using the provided onboard traffic management service, or handled specially e.g. inserted directly into an onboard mass store.</p>
Considerations	
<p>Normal routing is based on the Optional Deposit operation provided by the Packet Router.</p>	
Adaptation Step	
Ada Package	

<p>Make a new Ada package through instantiation of generic package Packet Depositor with a procedure reflecting the choice from input 1. Data_Handling_System.Router may serve as inspiration.</p>	<p>Packet Depositor</p>
---	-------------------------

11.2.2 Large Data Transfer Sending Sub-Service Routing

The following steps are mandatory if, and only if, the application process is to provide the large data transfer sending sub-service (see section 11.4.8).

The Optional Deposit operation provided by the Large Data Transfer Service instance shall be used for normal routing. (The Packet Depositor instance generated in section 11.2.1 is always used as generic actual parameter for the Large Data Transfer instance.)

11.3 Application Process Bus I/F

This step shall only be performed *if the application process provides an onboard monitoring service or a housekeeping & diagnostics service.*

Interpretation	
<p>A collection of onboard parameters may be associated to the application process such that they may be subjected to monitoring, housekeeping collection etc.</p> <p>A specific application process bus interface component has to be 'plugged' into the existing framework that given a parameter ID will collect the current value for this parameter.</p>	
Input	
1	List of parameter IDs for onboard parameters to be associated to application process. This shall be a subset of the system wide list established in 9.3 'Parameter IDs & Types'.
2	For each parameter identified by input 1 it shall be specified <i>how</i> the current value may be read (e.g. from a register on the data handling system platform or from an onboard bus).
3	<p>If the application process is to provide housekeeping & diagnostics service:</p> <ul style="list-style-type: none"> ● Specification of the filter being applied in filtered mode. This shall indicate <i>how</i> two parameter samples and a threshold value map into a boolean indicating whether the sample is to be included. See the definition of services (3,19) and (3,20).
4	<p>If the application process is to provide an onboard monitoring service:</p> <ul style="list-style-type: none"> ● A list of Check Selection Parameter Numbers ● Specification of <i>how</i> the current value of the onboard check selection parameter associated to a given Check Selection Parameter Number is read.
5	<p>If the application process is to provide an onboard monitoring service:</p> <ul style="list-style-type: none"> ● A list of Parameter Validity Numbers ● Specification of <i>how</i> the current value of the onboard validity parameter associated to a given Parameter Validity Number is read.
Considerations	

<p>Type and representation for the parameters identified by input 1 has been specified in 9.3 'Parameter IDs & Types'.</p> <p>An onboard parameter may be associated to more than one application process.</p>	
<i>Adaptation Step</i>	<i>Ada Package</i>
<p>Implement an Application Process Bus IF Ada package providing the following operations:</p> <ul style="list-style-type: none"> ● <i>Is Legal Parameter ID</i> being a predicate taking a parameter ID and returning true iff it is in the list given by input 1. ● <i>Receive</i> operation mapping a parameter ID contained in input 1 into the current parameter value⁸ for this onboard parameter. 	<p>Power_Conditioning_System.Driver_IF may serve as inspiration.</p>
<p>If the application process is to provide a housekeeping & diagnostics service (service type 3) then the Application Process Bus IF shall be extended with a <i>Threshold Is Exceeded</i> operation mapping two parameter values and a Threshold into a boolean being true iff the value shall be included in a housekeeping or diagnostics report. See input 3.</p>	<p>Power_Conditioning_System.Driver_IF may serve as inspiration.</p>
<p>If the application process is to provide a housekeeping & diagnostics service (service type 3) then the Application Process Bus IF shall be extended with a <i>Threshold Is Legal</i> operation mapping a parameter ID and a Threshold specification into a boolean being true iff the given threshold — i.e. type and value — is legal for the specified onboard parameter. See input 3.</p>	<p>Power_Conditioning_System.Driver_IF may serve as inspiration.</p>
<p>If the application process is to provide an onboard monitoring service then the Application Process Bus IF shall be extended with an <i>Is Selected</i> operation mapping a Check Selection Parameter Number into the current value of the associated check selection parameter. See input 4.</p>	<p>Power_Conditioning_System.Driver_IF may serve as inspiration.</p>
<p>If the application process is to provide an onboard monitoring service then the Application Process Bus IF shall be extended with an <i>Is Valid</i> operation mapping a Parameter Validity Number into the current value of the associated validity parameter. See input 5.</p>	<p>Power_Conditioning_System.Driver_IF may serve as inspiration.</p>

11.4 Instantiation of PUS Services

Each application process may provide one and only one instance of each service from the Telemetry & Telecommand Packet Utilization standard. The services are considered in turn, and adaptations take place if they are to be included in the application process resulting in the following steps:

⁸ : These parameter values shall belong to the tagged type Standard Value covering the supported parameter types and structures as defined in chapter 23 of Telemetry & Telecommand Packet Utilization standard.

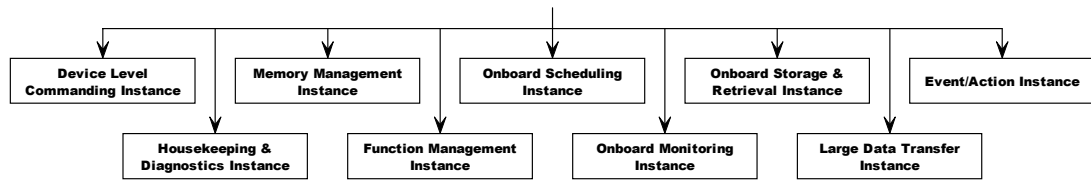


Figure 8: Service Instantiations

11.4.1 Device Level Commanding Instance

The following steps should only be performed if the application process is to provide a device level commanding service (service type 2).

Beware that only one application process in the system may provide an instance of this service.

Interpretation	
A specific instance of the device level commanding service has to be produced using the associated application process ID and knowledge of the expected operational scenarios including the application process.	
Input	
1	Application process ID assigned to current application process.
2	Task priority and length of protected queue queueing telecommands up for execution.
3	Task priority and stack size for sporadic task executing telecommands.
Considerations	
Note that task priority assigned to queue in input 2 <i>shall</i> exceed the priority of any tasks depositing telecommands for execution.	
Adaptation Step	Ada Package
Update Application ID according to input 1.	External Device Command Distribution Types
Update Telecommand Buffer Size and Telecommand buffer Priority according to input 2.	External Device Command Distribution Types
Update Telecommand Interpreter Priority and Telecommand Interpreter Stack Size according to input 3.	External Device Command Distribution Types

11.4.2 Housekeeping & Diagnostics Instance

The following steps should only be performed if the application process is to provide a housekeeping & diagnostic data reporting service (service type 3).

Interpretation	
A specific instance of the housekeeping & diagnostics service has to be produced based on specific characteristics of the application process and knowledge of how it will be operated during the mission.	
Input	
1	Application process ID assigned to current application process.
2	Maximum number of structure IDs to be defined at the same time during operation.
3	Maximum number of sampling events to be active at any one time.
4	Maximum number of simply-commutated parameters, super-commutated parameters, and repetitions of super-commutated parameters to be present in any housekeeping or diagnostics report.
5	Identification of the application process bus interface implemented in section 11.3. The operations Receive, Is Legal Parameter ID, and Threshold Is Legal, and Threshold Is Exceeded shall be available.
6	Identification of telemetry router implemented in section 11.2. The operation Optional Deposit shall be available.
7	Minimum period by which parameter values will be collected ⁹ , and the maximum number of collections that shall occur within such a period during operation.
8	Task priority and stack size of cyclic object performing parameter collection.
9	Priority of protected object implementing time line of collection events.
10	Task priority and length of protected queue queueing telecommands up for execution.
11	Task priority and stack size for sporadic task executing telecommands.
12	Task priority assigned to protected object managing the set of report definitions.
Considerations	
Task priority assigned to queue of telecommands by input 9 shall exceed ceiling of priority of any tasks submitting telecommands for execution.	
Task priority of input 10 shall exceed task priorities from input 11 and input 8.	
Adaptation Step	Ada Package
Make an instance of the generic Ada package implementing house-keeping & diagnostics using the below relations between above inputs and formal generic parameters. See Power_Conditioning_System.Parameters and Power_Conditioning_System.HK_Collector for inspiration.	HK Collector
Formal Generic Parameter	Input

⁹ : This will specify the period of the cyclic task performing the parameter collection.

Application ID	1		
Max SIDs	2		
Max Sampling Events	3		
Max Simple Params	4		
Max Super Params			
Max Super Reps			
Is Legal Parameter ID	5		
Receive			
Threshold Is Exceeded			
Threshold Is Legal			
Optional Deposit	6		
Max No Of Checks In HK Diag Period	7		
HK Diag Period			
HK Diag Event Task Priority	8		
HK Diag Event Task Size			
HK Diag Event List Priority	9		
Telecommand Buffer Priority	10		
Telecommand Buffer Size			
Telecommand Interpreter Priority	11		
Telecommand Interpreter Stack Size			
HK Diag Report Definition Priority	12		

11.4.3 Memory Management Instance

The following steps should only be performed if the application process is to provide a memory management service (service type 6).

Interpretation	
A specific instance of the memory management service has to be produced based on specific characteristics of the application process, knowledge of how it will be operated during the mission, and knowledge of how the memory being managed is accessed (e.g. paged, memory mapped).	
Input	
1	Application process ID assigned to current application process.
2	Maximum length of lists indicated by the N field in telecommand packets and telemetry packets.

3	Maximum length of Data fields in telecommand packets and telemetry packets.
4	List of memory IDs identifying onboard memory blocks to be managed by the service. This list indicates possible values in Memory ID fields in telecommand packets and telemetry packets.
5	For each memory ID in input 4, a specification shall be given as to <i>how</i> the identified memory block may be read and/or written and how an ISO checksum may be calculated for the block.
6	Identification of telemetry router implemented in section 11.2. The operation Optional Deposit shall be available.
7	Task priority and length of protected queue queueing telecommands up for execution.
8	Task priority and stack size for sporadic task executing telecommands.
Considerations	
Task priority assigned to queue in input 7 <i>shall</i> exceed the priority of any tasks depositing telecommands for execution.	
Adaptation Step	
Make an instance of the generic Ada package implementing memory management using the below relations between the above inputs and the formal generic parameters. See Data_Handling_System.Memory_Manager for inspiration.	
Ada Package	
Memory Management	
Formal Generic Parameter	
Input	
Application ID	1
Max List Length	2
Max Data Length	3
Memory ID	4 ¹⁰
Memory ID Rep	
Memory ID Rep Vals	
Write Data	5 ¹¹
Read Data	
Calculate ISO Checksum	
Optional Deposit	6
Telecommand Buffer Priority	7
Telecommand Buffer Size	

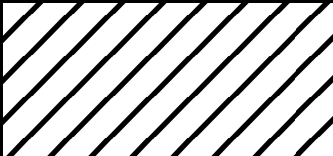
¹⁰ : Memory ID defines an internal enumeration type identifying the supported memory blocks. Memory ID Rep and Memory ID Rep Vals in combination define a mapping from internal memory IDs – being enumeration literals – to the external memory IDs being fixed character strings.

¹¹ : For memory blocks that are memory mapped – i.e. accessible from the normal memory space – generic package Memory Access may be used.

Telecommand Interpreter Priority	8	
Telecommand Interpreter Stack Size		

11.4.4 Function Management Instance

The following steps should only be performed if the application process is to provide a function management service (service type 8).

Interpretation	
A specific instance of the function management service has to be produced based on specific characteristics of the application process and knowledge of how it will be operated during the mission.	
Input	
1	Application process ID assigned to current application process.
2	Identification of telemetry router implemented in section 11.2. The operation Optional Deposit shall be available.
3	List of function IDs identifying the functions to be supported by the service. This list identifies possible values in the Function ID field of telecommand packets and telemetry packets.
4	For each function ID in input 3, a list of associated parameters shall be given. This list shall specify assigned parameter ID as well as the parameter code associated to the parameter, see section 23 of the Telemetry & Telecommand Packet Utilization Standard.
5	For every function identified in input 3 the following shall be specified: <ul style="list-style-type: none"> What actions are to be performed when the function is performed with a given set of parameter values as identified in input 4..
6	Task priority and length of protected queue queueing telecommands up for execution.
7	Task priority and stack size for sporadic task executing telecommands.
8	Identification of a Telecommand Verification Generator instance associated to the current application process.
Considerations	
Task priority assigned to queue in input 6 <i>shall</i> exceed the priority of any tasks depositing telecommands for execution.	
Adaptation Step	Ada Package
Implement a function interpreter that is able to perform the functions identified in input 3 as specified in input 5. See Power_Conditioning_System.Function_Interpreter for inspiration.	

Make an instance of the generic Ada package implementing memory management using the below relations between above inputs and formal generic parameters. See Power_Conditioning_System.Function_Management for inspiration.		Generic TC Translator
<i>Formal Generic Parameter</i>	<i>Input</i>	
Application ID	1	
Optional Deposit	2	
Activate Function	Function Interpreter	
Deactivate Function		
Perform Activity		
Telecommand Buffer Priority	6	
Telecommand Buffer Size		
Telecommand Interpreter Priority	7	
Telecommand Interpreter Stack Size		
Telecommand_Verifier	8	

11.4.5 Onboard Scheduling Instance

The following steps should only be performed if the application process is to provide an onboard scheduling service (service type 11).

<i>Interpretation</i>	
A specific instance of the onboard scheduling service has to be produced based on specific characteristics of the application process and knowledge of how it will be operated during the mission.	
<i>Input</i>	
1	Application process ID assigned to current application process.
2	Maximum number of telecommands on the command schedule.
3	Maximal length of parameter lists in telecommands and telemetry reports.
4	List of sub-schedule IDs identifying the sub-schedules to be handled by the service. This list identifies possible values in the Sub-schedule ID field of telecommand packets and telemetry packets. The list shall include a special value meaning 'All Sub-schedules'.
5	The period by which the schedule shall be checked for any due telecommands, and the maximum number of telecommands that can be due within such a period.
6	Identification of telemetry router implemented in section 11.2. The operation Optional Deposit shall be available.

7	Specification of <i>how</i> due telecommands are to be submitted for execution. Normally they will be passed on to the Packet Router implementing the onboard traffic management.																												
8	Task priority and length of protected queue queueing telecommands up for execution.																												
9	Task priority and stack size for sporadic task executing telecommands.																												
10	Priority of protected object implementing the schedule time line.																												
11	Task priority and stack size of cyclic object performing command scheduling.																												
Considerations																													
Task priority assigned to queue of telecommands by input 8 shall exceed ceiling of priority of any tasks submitting telecommands for execution. Task priority of input 10 shall exceed task priorities from input 9 and input 11.																													
Adaptation Step																													
<p>Make an instance of the generic Ada package implementing onboard scheduling using the following relations between above inputs and formal generic parameters. See Data_Handling_System.TC_Scheduler for inspiration.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Formal Generic Parameter</th> <th style="text-align: center;">Input</th> </tr> </thead> <tbody> <tr> <td>Application ID</td> <td style="text-align: center;">1</td> </tr> <tr> <td>Schedule Size</td> <td style="text-align: center;">2</td> </tr> <tr> <td>Max List Length</td> <td style="text-align: center;">3</td> </tr> <tr> <td>Sub Schedule ID</td> <td rowspan="4" style="text-align: center; vertical-align: middle;">4¹²</td> </tr> <tr> <td>All Sub Schedules</td> </tr> <tr> <td>Sub Schedule ID Rep</td> </tr> <tr> <td>Sub Schedule ID Rep Vals</td> </tr> <tr> <td>Scheduling Period</td> <td rowspan="2" style="text-align: center; vertical-align: middle;">5</td> </tr> <tr> <td>Max No Of Telecommands In Scheduling Period</td> </tr> <tr> <td>Optional Deposit</td> <td style="text-align: center;">6</td> </tr> <tr> <td>Forward TC</td> <td style="text-align: center;">7</td> </tr> <tr> <td>Telecommand Buffer Priority</td> <td rowspan="2" style="text-align: center; vertical-align: middle;">8</td> </tr> <tr> <td>Telecommand Buffer Size</td> </tr> <tr> <td>Telecommand Interpreter Priority</td> <td rowspan="2" style="text-align: center; vertical-align: middle;">9</td> </tr> <tr> <td>Telecommand Interpreter Stack Size</td> </tr> <tr> <td>Schedule Priority</td> <td style="text-align: center;">10</td> </tr> </tbody> </table>		Formal Generic Parameter	Input	Application ID	1	Schedule Size	2	Max List Length	3	Sub Schedule ID	4 ¹²	All Sub Schedules	Sub Schedule ID Rep	Sub Schedule ID Rep Vals	Scheduling Period	5	Max No Of Telecommands In Scheduling Period	Optional Deposit	6	Forward TC	7	Telecommand Buffer Priority	8	Telecommand Buffer Size	Telecommand Interpreter Priority	9	Telecommand Interpreter Stack Size	Schedule Priority	10
Formal Generic Parameter	Input																												
Application ID	1																												
Schedule Size	2																												
Max List Length	3																												
Sub Schedule ID	4 ¹²																												
All Sub Schedules																													
Sub Schedule ID Rep																													
Sub Schedule ID Rep Vals																													
Scheduling Period	5																												
Max No Of Telecommands In Scheduling Period																													
Optional Deposit	6																												
Forward TC	7																												
Telecommand Buffer Priority	8																												
Telecommand Buffer Size																													
Telecommand Interpreter Priority	9																												
Telecommand Interpreter Stack Size																													
Schedule Priority	10																												
Ada Package																													
On Board Scheduler																													

¹² : Sub Schedule ID defines an internal enumeration type identifying the possible sub-schedules. Sub Schedule ID Rep and Sub Schedule ID Rep Vals in combination defines a mapping from internal sub-schedule IDs – being enumeration literals – to the external sub-schedule IDs.



Scheduler Task Priority	11	
Scheduler Task Stack Size		

11.4.6 Onboard Monitoring Instance

The following steps should only be performed if the application process is to provide an onboard monitoring service (service type 12).

<i>Interpretation</i>	
A specific instance of the onboard monitoring service has to be produced based on specific characteristics of the application process and knowledge of how it will be operated during the mission.	
<i>Input</i>	
1	Application process ID assigned to current application process.
2	Maximum number of parameters that may be monitored at one time.
3	Maximum number of checks that may be applied by the monitoring service at one time.
4	The period by which monitoring of due parameters checks shall be performed, and the maximum number of checks that can be due within such a period.
5	The maximal time span from an out-of-limit event is detected to an out-of-limit report containing the event is submitted for routing. The maximum number of out-of-limit events that may occur within this time span shall also be specified.
6	Identification of telemetry router implemented in section 11.2. The operation Optional Deposit shall be available.
7	Identification of the application process bus interface implemented in section 11.3. The operations Receive, Is Legal Parameter ID, Is Valid, and Is Selected shall be available.
8	Task priority and length of protected queue queueing telecommands up for execution.
9	Task priority and stack size for sporadic task executing telecommands.
10	Priority of protected object implementing monitoring list.
11	Priority of protected object implementing collection of check definitions.
12	Priority of protected object implementing time line of monitoring events.
13	Task priority and stack size of cyclic object performing parameter monitoring.
14	Priority of protected object implementing list of OOL transitions.
15	Priority and stack size of cyclic task generating out-of-limit reports.
<i>Considerations</i>	

<p>Task priority assigned to queue of telecommands by input 8 shall exceed ceiling of priority of any tasks submitting telecommands for execution.</p> <p>Task priority of input 10 shall exceed task priorities from input 9 and input 13.</p> <p>Task priority of input 11 shall exceed task priority from input 9.</p> <p>Task priority of input 12 shall exceed task priorities from input 9 and input 13.</p> <p>Task priority of input 14 shall exceed task priorities from input 13 and input 15.</p>		
Adaptation Step	Ada Package	
<p>Make an instance of the generic Ada package implementing onboard monitoring using the below relations between above inputs and formal generic parameters.</p> <p>See Power_Conditioning_System.Monitoring for inspiration.</p>	Monitor	
Formal Generic Parameter		Input
Application ID		1
Max Params		2
Max Checks		3
Monitoring Period		4
Max No Of Checks In Monitoring Period		
Maximum Reporting Delay		5
Max No Of OOL Transitions		
Optional Deposit		6
Is Legal Parameter ID		7
Receive		
Is Valid		
Is Selected		
Telecommand Buffer Priority		8
Telecommand Buffer Size		
Telecommand Interpreter Priority		9
Telecommand Interpreter Stack Size		
Monitoring List Priority		10
Check Definitions Collection Priority		11
Monitor Time line Priority	12	
Monitor Task Priority	13	
Monitor Task Stack Size		
OOL Transition List Priority	14	
OOL Reporter Priority	15	

OOL Reporter Task Stack Size		
------------------------------	--	--

11.4.7 Onboard Storage & Retrieval Instance

The on-board storage and retrieval service (service type 15) consists of two sub-services: the Storage and Retrieval sub-service and the Packet Selection sub-service, each of which may be instantiated for any application process.

The following steps should only be performed if the application process is to provide a Storage and Retrieval sub-service.

Interpretation		
The storage & retrieval sub-service manages a collection of onboard stores to which telemetry packets may be sent by application processes providing a Packet Selection sub-service – to be defined below.		
Input		
1	Application process ID assigned to current application process.	
2	List of packet store IDs identifying the packet stores administered by this application process.	
3	For each packet store Id in input 2, a specification shall be given for <i>how</i> one inserts, deletes and extracts packets from that store.	
4	Routine for reading packet store identifiers of the right base type.	
5	Identification of telemetry router implemented in section 11.2. The operation Optional Deposit shall be available.	
6	Task priority and length of protected queue queueing telecommands up for execution.	
7	Task priority and stack size for sporadic task executing telecommands.	
8	For each packet store (administered by this application process) its storage strategy, i.e. 'cyclic' or 'bounded' must be decided as well as its maximal size (in terms of the number of stored packets).	
Considerations		
The routines referred to in input 3 are produced by using operations made available by instances of Packet Store. Each such instance is based on the characteristics defined in input 8 above. Task priority assigned to queue in input 6 shall exceed the priority of any tasks depositing telecommands for execution.		
Adaptation Step	Ada Package	
Make an instance of the generic Ada package implementing the storage and retrieval sub-service using the below relations between the above inputs and the formal generic parameters. See Onboard_Storage.Storage_And_Retrieval_Manager for inspiration.	Storage And Retrieval	
Formal Generic Parameter		Input
Application ID		1

Local Packet Store ID	2	
Delete All Packets	3	
Delete To Packet		
Delete To Time		
Get Packets In Range		
Get Packets In Period		
Get Store ID	4	
Optional Deposit	5	
Telecommand Buffer Priority	6	
Telecommand Buffer Size		
Telecommand Interpreter Priority	7	
Telecommand Interpreter Stack Size		
Make (global) instances of Packet Stores for each of the stores identified in input 1 and characterised by input 8. See High_Priority_Store for inspiration.		
Formal Generic Parameter	Input	
My APID	1	
Strategy	8	
Max Packets		

The following steps should only be performed if the application process is to provide a Packet Selection sub-service.

Interpretation	
The Packet Selection sub-service manages a collection of storage selection definitions controlling a possible transmission of telemetry packets to application processes implementing a storage & retrieval sub-service – as defined above.	
Input	
1	Application process ID assigned to current application process.
2	Identification of telemetry router implemented in section 11.2. The operation Deposit shall be available.
3	Defining the maximal number of stores to which this process may store telemetry.
4	Task priority and length of protected queue queueing telecommands up for execution.
5	Task priority and stack size for sporadic task executing telecommands.
Considerations	
Task priority assigned to queue in input 4 shall exceed the priority of any tasks depositing telecommands for execution.	

Adaptation Step		Ada Package
<p>Make an instance of the generic Ada package implementing the packet selection sub-service using the below relations between the above inputs and the formal generic parameters.</p> <p>See Power_Conditioning_System.Storage_Selection_Defs for inspiration.</p>		Storage Selection
Formal Generic Parameter	Input	
Application ID	1	
Optional_Deposit	2	
Max Store IDs	3	
Telecommand_Buffer_Priority	4	
Telecommand_Buffer_Size		
Telecommand_Interpreter_Priority	5	
Telecommand_Interpreter_Stack_Size		

11.4.8 Large Data Transfer Instance

The large data transfer service (service type 13) consists of two symmetrical but distinct sub-services: The sending and the receiving sub-service. Any application process may provide an instance of on, or both, of the sub-services.

The following steps shall only be performed if the application process is to provide either, or both, of the large data transfer sub-services.

Interpretation	
<p>A specific instance of the large data transfer has to be produced based on the specific characteristics of the application process, and knowledge of how it will be operated during the mission.</p> <p>This is the parent instance for the large data transfer.</p> <p>This parent instance contains default procedures used by its child instances.</p>	
Input	
1	Application process ID assigned to current application process.
2	The Packet Depositor instance, instantiated in section 11.2.1.
3	The maximum length of any list of sequence numbers in telecommand packets and telemetry packets.
Considerations	
None.	
Adaptation Step	Ada Package

<p>Make an instance of the generic Ada package implementing the large data transfer using the below relations between the above inputs and the formal generic parameters.</p> <p>See Data_Handling_System.Large_Data_Manager for inspiration.</p>		<p>Large Data Transfer</p>
<i>Formal Generic Parameter</i>	<i>Input</i>	
Application Process ID	1	
The Depositor	2	
Sequence Number List Length	3	

The following steps shall only be performed if the application process is to provide the large data transfer sending sub-service.

<i>Interpretation</i>	
<p>A specific instance of the large data transfer.sender has to be produced based on the specific characteristics of the application process, knowledge of how it will be operated during the mission.</p> <p>This is the large data transfer sender child of the parent instance for the large data transfer.</p>	
<i>Input</i>	
1	Task priority and length of protected queue queueing telecommands up for execution.
2	Task priority and stack size for sporadic task executing telecommands.
3	Task priority and length of protected queue queueing telemetry up that are to be send as service data units.
4	Task priority and stack size for sporadic task sending telemetry as service data units.
5	Task priority of the protected object encapsulating the state change mechanism. The state object reflects the reaction to received telecommands, and telemetry being send as service data units, and timeouts.
6	Maximum size of queue holding part sequence numbers the receiver of a service data unit has requested to be repeated (re-downlinked).
7	The timeout interval after having sent all parts of a service data unit. The timeout interval is specified in milliseconds. After having sent the last part of a service data unit the sending sub-service shall receive either an acknowledgement telecommand that all parts have been received, or a request to repeat parts telecommand. Otherwise the downlink of the service data unit times out, and is aborted.
8	A timeout handler procedure to call when a downlink of a service data unit times out. A default sender timeout handler procedure is implemented in the parent instance for the large data transfer.
9	Task priority of the protected object encapsulating the control of the actual timeout timer, i.e. start and stop of the timer.
10	Task priority of the active timeout timer task.
<i>Considerations</i>	

Task priorities:

- Task priority assigned to queue in input 1 shall exceed the priority of any task depositing telecommands for execution.
- Task priority assigned to queue in input 3 shall exceed the priority of any task depositing telemetry (or telecommands) to the large data transfer service, i.e. using the Optional Deposit of the large data transfer service.
- Task priority assigned to object in input 5 shall at least be the ceiling of the task priorities assigned to tasks in input 2 and 4.
- Task priority assigned to the active task in input 10 shall at least be the ceiling of the task priorities assigned to tasks in input 2 and 4, and at most be equal to the priority assigned to the object in input 5.
- Task priority assigned to object in input 9 shall at least be the ceiling of the task priority assigned to object in input 5 and active task in input 10.

The last three task priority constraints can be summarized as follows:

$$\text{Priority}(\text{Input}9) \geq \text{Priority}(\text{Input}5) \geq \text{Priority}(\text{Input}10) \geq \text{Maximum of } \{\text{Priority}(\text{Input}2), \text{Priority}(\text{Input}4)\}.$$

If, and only if, the size of the queue holding part sequence numbers to be repeated is set to zero, the large data transfer sending sub-service is prohibited from repeating parts. Any telecommand request from the receiver of a service data unit to repeat parts will be rejected as an unsupported activity.

A default sender timeout handler procedure is implemented in the parent instance for the large data transfer. The default sender timeout handler does not carry out any actions.

Adaptation Step		Ada Package
<p>Make an instance of the generic Ada package implementing the large data transfer.sender using the below relations between the above inputs and the formal generic parameters. The Ada packet shall be instantiated as a child of the parent instance for the large data transfer.</p> <p>See Data Handling System.Large Data Manager.Large Data Transmitter for inspiration.</p>		Large Data Transfer.Sender
Formal Generic Parameter	Input	
TC Handler Buffer Priority	1	
TC Handler Buffer Size		
TC Handler Interpreter Priority	2	
TC Handler Interpreter Stack Size		
SDU Sender Buffer Priority	3	
SDU Sender Buffer Size		
SDU Sender Interpreter Priority	4	
SDU Sender Interpreter Stack Size		
Protected State Priority	5	
Max Resend Queue Length	6	
Timeout Interval In Milliseconds	7	

Sender Timeout Handler (Defaulted by Default Sender Timeout Handler implemented in the parent instance for the large data transfer)	8	
Protected Timer Control Priority	9	
Active Timer Priority	10	

The following steps shall only be performed if the application process is to provide the large data transfer receiving sub-service.

Interpretation	
<p>A specific instance of the Large Data Transfer.Receiver has to be produced based on the specific characteristics of the application process, knowledge of how it will be operated during the mission.</p> <p>This is the large data transfer receiver child of the parent instance for the large data transfer.</p>	
Input	
1	Task priority and length of protected queue queueing telecommands up for execution.
2	Task priority and stack size for sporadic task executing telecommands.
3	Task priority of the protected object encapsulating the state change mechanism. The state object reflects the reaction to received telecommands, and timeouts.
4	The timeout interval after having received a part (or repeated part) of a service data unit. The timeout interval is specified in milliseconds. After having received a part (or repeated part) of a service data unit, which is not the last part (or last repeated part), the receiving sub-service shall receive yet a part (or repeated part). If this part (or repeated part) is not received within the timeout interval, the uplink of the service data unit times out, and is aborted.
5	A timeout handler procedure to call when a uplink of a service data unit times out. A default receiver timeout handler procedure is implemented in the parent instance for the large data transfer.
6	Task priority of the protected object encapsulating the control of the actual timeout timer, i.e. start and stop of the timer.
7	Task priority of the active timeout timer task.
Considerations	

Task priorities:

- Task priority assigned to queue in input 1 shall exceed the priority of any task depositing telecommands for execution.
- Task priority assigned to object in input 3 shall at least be the ceiling of the task priority assigned to the task in input 2.
- Task priority assigned to the active task in input 7 shall at least be the ceiling of the task priority assigned to the task in input 2, and at most be equal to the priority assigned to the object in input 3.
- Task priority assigned to object in input 6 shall at least be the ceiling of the task priority assigned to object in input 4 and active task in input 7.

The last three task priority constraints can be summarized as follows:

$$\text{Priority}(\text{Input}6) \geq \text{Priority}(\text{Input} 3) \geq \text{Priority}(\text{Input} 7) \geq \text{Priority}(\text{Input} 2).$$

A default receiver timeout handler procedure is implemented in the parent instance for the large data transfer. The default receiver timeout handler does not carry out any actions.

Adaptation Step		Ada Package
<p>Make an instance of the generic Ada package implementing the Large Data Transfer.Receiver using the below relations between the above inputs and the formal generic parameters. The Ada packet shall be instantiated as a child of the parent instance for the large data transfer.</p> <p>See Data Handling System.Large Data Manager.Large Data Receiver for inspiration.</p>		Large Data Transfer.Receiver
Formal Generic Parameter	Input	
TC Handler Buffer Priority	1	
TC Handler Buffer Size		
TC Handler Interpreter Priority	2	
TC Handler Interpreter Stack Size		
Protected State Priority	3	
Timeout Interval In Milliseconds	4	
Receiver Timeout Handler ¹³	5	
Protected Timer Control Priority	6	
Active Timer Priority	7	

The following steps shall only be performed if the application process is to provide either, or both, of the large data transfer sub-services.

Interpretation

¹³ Defaulted by Default Receiver Timeout Handler implemented in the parent instance for the large data transfer.

<p>A specific instance of the Large Data Transfer.Service has to be produced based on the specific characteristics of the application process, knowledge of how it will be operated during the mission, and knowledge of which of the two sub-services (or both) are to be instantiated.</p> <p>This is the large data transfer service child of the parent instance for the large data transfer. This is the instance providing the service interface to the large data transfer service.</p>		
Input		
1	<p>The PUS packet handler for the large data transfer receiving sub-service.</p> <p>If the receiving sub-service is to be used, this input must be the corresponding PUS packet handler from the Large Data Transfer.Receiver instance.</p> <p>A default receiver telecommand PUS packet handler is implemented in the parent instance for the Large Data Transfer. This default subprogram is an error handler for the cases where no actual generic parameter is supplied for this parameter, but the large data transfer receiver is non the less forwarded a telecommand PUS packet to be handled.</p>	
2	<p>The PUS packet handler for the Large Data Transfer sending sub-service.</p> <p>If the sending sub-service is to be used, this input must be the corresponding PUS packet handler from the Large Data Transfer.Sender instance.</p> <p>A default sender telecommand PUS packet handler is implemented in the parent instance for the large data transfer. This default subprogram is an error handler for the cases where no actual generic parameter is supplied for this parameter, but the large data transfer sender is non the less forwarded a telecommand PUS packet to be handled.</p>	
3	<p>The telemetry deposit function for the Large Data Transfer sending sub-service.</p> <p>If the sending sub-service is to be used, this input must be the corresponding telemetry deposit function from the Large Data Transfer.Sender instance.</p> <p>A default telemetry deposit function is implemented in the parent instance for the Large Data Transfer. This default subprogram is an error handler for the cases where no actual generic parameter is supplied for this parameter, but the Large Data Transfer sender is non the less forwarded a telemetry packet to be sent as a service data unit.</p>	
Considerations		
<p>If the receiving sub-service is to be used, correct input from the large data transfer.receiver instance for input 1 shall be supplied.</p> <p>If the sending sub-service is to be used, correct input from the large data transfer.sender instance for input 2 and 3 shall be supplied.</p>		
Adaptation Step	Ada Package	
<p>Make an instance of the generic Ada package implementing the Large Data Transfer.Service using the below relations between the above inputs and the formal generic parameters.</p> <p>See Data Handling System.Large Data Manager.Large Data Service for inspiration.</p>	<p>Large Data Transfer.Service</p>	
Formal Generic Parameter		Input
<p>Large Data Transfer Receiver Handle TC Packet¹⁴</p>		<p>1</p>

¹⁴ Defaulted by the error handler Receiver Not Supported Handle TC Packet implemented in the parent instance for the large data transfer.

Large Data Transfer Sender Handle TC Packet ¹⁵	2	
Large Data Transfer Sender Deposit TM Packet ¹⁶	3	

11.4.9 Event Action Instance

The following steps shall only be performed if the application process is to provide an event action service (service type 19).

Interpretation	
A specific instance of the Event Action. Service has to be produced based on the specific characteristics of the application process, and knowledge of how it will be operated during the mission.	
Input	
1	Application process ID assigned to current application process.
2	The maximum length of the event action detection list.
3	The maximum length of the list of events and action telecommand in an add event telecommand packet.
4	The maximum length of the list of events in a delete event telecommand packet.
5	The maximum length of the list of events in an enable or disable event telecommand packet.
6	Identification of the telemetry router implemented in section 10.2. The operation Optional Deposit shall be available.
7	Task priority and length of protected queue queueing telecommands up for execution.
8	Task priority and stack size for sporadic task executing telecommands.
9	Task priority of the protected object encapsulating the event action detection list.
Considerations	
<p>The input for the maximum list length in input 3, 4 and 5 shall be less than or equal to the maximum list length for the event action detection list in input 2.</p> <p>Task priorities:</p> <ul style="list-style-type: none"> ● Task priority assigned to queue in input 7 shall exceed the priority of any task depositing telecommands for execution. ● Task priority assigned to object in input 9 shall at least be the ceiling of the task priority assigned to the task in input 8. 	
Adaptation Step	Ada Package

¹⁵ Defaulted by the error handler Sender Not Supported Handle TC Packet implemented in the parent instance for the large data transfer.

¹⁶ Defaulted by the error handler Sender Not Supported Deposit TM Packet implemented in the parent instance for the large data transfer.

<p>Make an instance of the generic Ada package implementing the large data transfer using the below relations between the above inputs and the formal generic parameters. See Data_Handling_System.Event_Action_Manager for inspiration.</p>		Event Action.Service
<i>Formal Generic Parameter</i>	<i>Input</i>	
Application Process ID	1	
Max Detection List Length	2	
Max Add Events	3	
Max Delete Events	4	
Max Action Controls	5	
Optional Deposit	6	
PUS Handler Buffer Priority	7	
PUS Handler Buffer Size		
PUS Handler Interpreter Priority	8	
PUS Handler Interpreter Stack Size		
Detection List Priority	9	

11.5 Telecommand Dispatcher

This step is mandatory providing the link between the service instances created above and the interface provided by the application process.

<i>Interpretation</i>	
<p>The collection of Telemetry & Telecommand Packet Utilization standard service instances developed above as to be combined into one component implementing the application process.</p>	
<i>Input</i>	
1	List of names of software components – i.e. Ada packages – implementing the Telemetry & Telecommand Packet Utilization standard services to be provided by the application process. These are the results of the steps carried out in section 11.4. 'Instantiation of PUS Services'.
2	Name of overall software component – i.e. Ada package – implementing the application process. This was selected during the adaptation of the Onboard Traffic Management service in section 10.2.
3	Application process ID assigned to current application process.
4	Task priority and length of protected queue queueing telecommands up for execution.
5	Task priority and stack size for sporadic task executing telecommands.
<i>Considerations</i>	
None.	



Adaptation Step		Ada Package
<p>Create an <i>Application Process</i> Telecommand Interpreter package providing a Handle PUS Packet Operation.</p> <p>This operation shall accept a telecommand packet, examine the Service Type of the packet, and forward it to the software component from input 1 providing the identified Telemetry & Telecommand Packet Utilization standard service.</p> <p>See Data Handling System.TC Interpreter for inspiration.</p>		
<p>Create an <i>Application Process</i> Dispatcher containing a sporadic task accepting Telecommand packets – or even telemetry packets – and forwarding telecommand packets to the Handle PUS Packet Operation in the <i>Application Process</i> Telecommand Interpreter package and telemetry packets to similar routines e.g. in an onboard storage & retrieval service.</p> <p>This involves instantiation of the generic package Sporadic Task the below relations between the above inputs and the formal generic parameters.</p> <p>See Data Handling System.Dispatcher for inspiration.</p>		Sporadic Task
Formal Generic Parameter	Input	
My Application ID	3	
Event Buffer Task Priority	4	
Sporadic Task Priority	5	
Sporadic Task Stack Size	5	
Sporadic Operation Parameter Type	PUS.PUS Packet	
Sporadic Event Buffer Size	4	
Sporadic Operation	Handle PUS Packet operation from previous step	
<p>Create a package having the name given by input 2.</p> <p>This package shall — as part of its elaboration — invoke the <i>Initialize</i> operations on the Ada packages identified in input 1.</p> <p>This package shall provide one or more of the following operations:</p> <ul style="list-style-type: none"> • <i>Forward TC, Forward TM, or Forward Packet</i> accepting a telecommand packet and forwarding it to the Handle PUS Packet Operation in the <i>Application Process</i> Dispatcher package produced in the previous step. <p>See Data Handling System for inspiration.</p>		

12 Main Program

The main program shall tie all the components making up the data handling system together. This may include components produced in the previous steps as well as other components not related to the Telemetry & Telecommand Packet Utilization standard services.

<i>Interpretation</i>	
The components that have been adapted and instantiated in the previous steps has to be initialised.	
<i>Input</i>	
1	List of application process packages created as a result of section 11.5 'Telecommand Dispatcher'.
<i>Considerations</i>	
None.	
<i>Adaptation Step</i>	<i>Ada Package</i>
Produce a main program performing the following steps: <ul style="list-style-type: none"> • Call CDH Structure Initializer.Initialize to initialise the general logistic framework. • Include context clauses — i.e. 'withs' — including all application process packages from input 1. 	

