



BSI Standards Publication

Aerospace series — Modular and Open Avionics Architectures

Part 005: Software

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

raising standards worldwide™

Full document details: BS EN 4660-005:2011

National foreword

This British Standard is the UK implementation of EN 4660-005:2011.

The UK participation in its preparation was entrusted to Technical Committee ACE/6, Aerospace avionic electrical and fibre optic technology.

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

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

© BSI 2011

ISBN 978 0 580 62445 2

ICS 49.090

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

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 May 2011.

Amendments issued since publication

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

EUROPEAN STANDARD
NORME EUROPÉENNE
EUROPÄISCHE NORM

EN 4660-005

May 2011

ICS 49.090

English Version

**Aerospace series - Modular and Open Avionics Architectures -
Part 005: Software**

Série aérospatiale - Architectures Avioniques Modulaires et
Ouvertes - Partie 005: Software

Luft- und Raumfahrt - Modulare und offene
Avionikarchitekturen - Teil 005: Software

This European Standard was approved by CEN on 26 June 2010.

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

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

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



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: Avenue Marnix 17, B-1000 Brussels

© 2011 CEN All rights of exploitation in any form and by any means reserved
worldwide for CEN national Members.

Ref. No. EN 4660-005:2011: E

Contents

Page

0	Introduction	11
0.1	Purpose.....	11
0.2	Document structure.....	12
1	Scope	12
1.1	Software Architecture Overview	12
1.2	Software Architectural Components	13
2	Normative references	15
3	Terms, definitions and abbreviations.....	16
3.1	Terms and definitions	16
3.2	Abbreviations.....	16
4	System Functions	19
4.1	System Management Function.....	19
4.2	Communication.....	26
4.3	Security Management.....	45
4.4	Module Management	49
4.5	Mass Memory Management.....	50
4.6	Graphics Management	54
4.7	Power Management	56
4.8	Network Management.....	58
4.9	Time Management.....	61
5	Software Architecture Definition.....	65
5.1	MSL	66
5.2	OSL.....	71
5.3	RTBP	107
5.4	Application Layer.....	110
6	Direct Interfaces Definitions.....	117
6.1	APOS.....	117
6.2	MOS.....	189
6.3	SMBP.....	270
6.4	SMOS	296
7	Logical Interfaces Definitions.....	341
7.1	OLI	341
7.2	GLI	348
7.3	SMLI	373
7.4	MLI.....	381
8	Data Type Definitions	439
8.1	IDL	439
8.2	Data Types.....	441
9	Tailoring.....	487

Annex A (normative) AGL	496
A.1 The Concept	496
A.2 Graphical Command Set	496
A.2.1 Overview	496
A.2.2 Command Listings	497
A.2.3 Auxiliary Library (AL) Definition	501
A.2.4 Video Library (VL) Definition	502
A.2.5 Texture Mapping Constraints	503
A.2.6 Display Frame and Synchronisation	505
A.2.7 Command Responses and Delays	505

Figures

Page

Figure 1 — ASAAC Standard Documentation Hierarchy	11
Figure 2 — ASAAC Three Layer Software Architecture	12
Figure 3 — The Software Architecture Model	13
Figure 4 — Hierarchical Organisation of the System Management	20
Figure 5 — GSM Decomposition for RE-Management (Example)	21
Figure 6 — IA Application Control (Example)	22
Figure 7 — GSM Decomposition for Module Management (Example)	23
Figure 8 — Hierarchical Organisation of the AM (Example)	24
Figure 9 — The ASAAC Communication Stack	27
Figure 10 — Types of Data Transfer	29
Figure 11 — Communication Concept	30
Figure 12 — Between AL Communication Routing	31
Figure 13 — ASAAC Message in BMC Data Transfer	33
Figure 14 — Multicast Scheme With a Single TC	34
Figure 15 — Multicast Scheme With Multiple Simple TC's	35
Figure 16 — Data Parallelism	36
Figure 17 — Corner Turn	36
Figure 18 — Corner Turn in Three Dimensions	37
Figure 19 — Illustration of the Involved Services in DSP1	38
Figure 20 — Data Representation	41
Figure 21 — GSM Interfaces	46

Figure 22 — Main Security Related Architectural Components	47
Figure 23 — VC transferring Data Requiring Encryption & Decryption.....	48
Figure 24 — File Handling by a Remote Application	51
Figure 25 — MMM Download onto a CFM with only the MSL.....	52
Figure 26 — CFM Download onto a CFM with only the MSL.....	53
Figure 27 — Application Download	54
Figure 28 — Graphics Concept	55
Figure 29 — Graphics Standard	55
Figure 30 — Application Control.....	57
Figure 31 — PCM Management (Example).....	57
Figure 32 — Configuration of a NSM	60
Figure 33 — Network Configuration Message Format.....	60
Figure 34 — Clock Hierarchy in an ASAAC System.....	63
Figure 35 — Software Architecture.....	65
Figure 36 — ASAAC Software Stack on a CFM	67
Figure 37 — GSM Logical Interface	72
Figure 38 — GSM: External Interfaces.....	73
Figure 39 — Thread State Transition Diagram with sample APOS services.....	81
Figure 40 — Process State Diagram	86
Figure 41 — Example for a 1 to N FIFO	89
Figure 42 — Example for a 1 to N LIFO	89
Figure 43 — OS Error Handling of an Application Error.....	97
Figure 44 — OS Error Handling of an MSL Error Due to a Return of a MOS Service	98
Figure 45 — OS Error Handling of an MSL Error Due to a CBIT Status.....	99
Figure 46 — The OLI.....	101
Figure 47 — Decomposition for OLI	101
Figure 48 — RTBP Tree Concept	108
Figure 49 — Relation of Processes and Threads and VC's.....	111
Figure 50 — Software Architecture Model - Three-Layer Stack (TLS)	189

Figure 51 — MOS Software Architecture Model.....	189
Figure 52 — sendFragmentedTransfer Data Buffer Description	247
Figure 53 — Splitting of Incoming Data with receiveFragmentedTransfer	249
Figure 54 — Different Step Sizes with Fragmented Transfers.....	250
Figure 55 — Root Definition	274
Figure 56 — Function Set Definition	275
Figure 57 — Configuration Set Definition	276
Figure 58 — Process Set Definition.....	278
Figure 59 — VC Set Definition.....	279
Figure 60 — TC Set Definition.....	280
Figure 61 — CFM Set Definition	281
Figure 62 — PE Set Definition.....	282
Figure 63 — Clock Set Definition.....	282
Figure 64 — State Machine Set Definition	284
Figure 65 — General VC Message Format	342
Figure 66 — File Reading Protocol	344
Figure 67 — Remote MLI Download Management Protocol	345
Figure 68 — General SMLI Message Format	374
Figure 69 — General TC Message Format	381
Figure 70 — General MLI Message Format.....	382
Figure 71 — Optional Parameter Element Format	382
Figure 72 — Request PBIT Result Format	384
Figure 73 — Reply PBIT Result Format.....	384
Figure 74 — Request CFM Status Format.....	385
Figure 75 — Reply CFM Status Format.....	386
Figure 76 — Request CFM Info Format.....	389
Figure 77 — Reply CFM Info Format	390
Figure 78 — Test Message Format	390
Figure 79 — Test Message Acknowledge Format.....	391

Figure 80 — Request IBIT Start Format.....	391
Figure 81 — Reply IBIT Start Format	392
Figure 82 — Request IBIT Result Format.....	392
Figure 83 — Reply IBIT Result Format	393
Figure 84 — Load Image Format	394
Figure 85 — Load Image Acknowledge Format.....	396
Figure 86 — Load Routing Table Format	397
Figure 87 — Load Routing Table Acknowledge Format.....	403
Figure 88 — Load Time Configuration Format	404
Figure 89 — Load Time Configuration Acknowledge Format.....	409
Figure 90 — Request AGT Format	410
Figure 91 — Reply AGT Format.....	410
Figure 92 — Ready_For_ALT_Synchro Format.....	411
Figure 93 — Start_ALT_Synchro Format	412
Figure 94 — Request ALT Format.....	412
Figure 95 — Reply ALT Format	413
Figure 96 — Request AGT ALT Format	414
Figure 97 —Reply AGT ALT Format.....	414
Figure 98 — Load Network Configuration Format	416
Figure 99 — NSM Switch Command Format.....	417
Figure 100 — NSM Connection Command Format	418
Figure 101 — NSM Reset Command Format	418
Figure 102 – NSM Execute Command Format	418
Figure 103 — Load Network Configuration Acknowledge Format	419
Figure 104 — Load Network Configuration Format	420
Figure 105 — Reply Network Status Format.....	420
Figure 106 — Load Power Switches Configuration Format	422
Figure 107 — PCM Switch Command Format.....	423
Figure 108 – Power Switch Command Format	424

Figure 109 — Power Switch Reset Format	424
Figure 110 — Power Switch Configuration Acknowledge Format	424
Figure 111 — Request Power Switch Status Format	426
Figure 112 — Reply Power Switches Status Format	426
Figure 113 — General CFM Resource Management Protocol	429
Figure 114 — General Download Management Protocol	432
Figure 115 — General Time Management Protocol	434
Figure 116 — Load Network Configuration Format	440
Figure A.1 — Graphics Concept	496

Tables

Page

Table 1 — Software Layer Independence	13
Table 2 — CBIT Modes	26
Table 3 — Routing Information and Data Transfer	33
Table 4 — IDL Primitive Types	43
Table 5 — IDL Constructive Types	45
Table 6 — Power Switching Services	56
Table 7 — Layers, Process Classes, and Standardised Interfaces	66
Table 8 — List of SMOS Services for RE-CM	74
Table 9 — List of SMOS Services for RE-HM	75
Table 10 — List of SMOS Services for RE-FM	75
Table 11 — List of SMOS Services for RE-SM	76
Table 12 — List of SMOS Services for IA-CM	77
Table 13 — List of SMOS Services for IA-FM	78
Table 14 — List of SMOS Services for AC-FM	79
Table 15 — Transition of Thread States	82
Table 16 — Condition of State Transition	82
Table 17 — Properties of Time Services	93
Table 18 — Resource Parameters of Basic Resource Entities	112

Table 19 — Criticality Classes of APOS Services	113
Table 20 — Safety Restriction Definitions	116
Table 21 — APOS Services	118
Table 22 — APOS File Seek Modes.....	180
Table 23 — Core MOS Services.....	190
Table 24 — Specific Board MOS Services	235
Table 25 — MOS Bespoke Extension Services	251
Table 26 — Overview of All SMBP Services	270
Table 27 — Identifiers Described as ID_ITEM.....	271
Table 28 — Mapping of EBNF Specification with RTBP Concept.....	273
Table 29 — Overview of all SMOS Services	296
Table 30 — MLI Download Types	332
Table 31 — OLI Services	346
Table 32 — GLI Services List.....	349
Table 33 — SMLI Services List.....	374
Table 34 — MLI Services	383
Table 35 — Reply PBIT Status Payload Field Definition.....	385
Table 36 — Reply CFM Status Payload Field Definition	386
Table 37 — GENERIC_CFM Specific Extension to Payload Information	387
Table 38 — NOT_GENERIC_CFM Specific Extension to Payload Information.....	389
Table 39 — Reply CFM Info Payload Field Definition.....	390
Table 40 — Reply IBIT Status Payload Field Definition	392
Table 41 — Reply IBIT Result Payload Field Definition	393
Table 42 — Load Image Payload Field Definition.....	394
Table 43 — Load Image Acknowledge Payload Field Definition	396
Table 44 — Load Routing Table Payload Field Definition	398
Table 45 — Load Routing Table Data Definition.....	399
Table 46 — Data Definition for Interface Configuration.....	399
Table 47 — Data Definition for Transfer Configuration	400

Table 48 — Data Definition for Protocol Configuration.....	402
Table 49 — Data Definition for Destroy Transfer	402
Table 50 — Load Routing Table Acknowledge Payload Field Definition	403
Table 51 — Load Time Configuration Payload Field Definition.....	405
Table 52 — Load Time Configuration Data Definition	406
Table 53 — Data Definition for Clock Configuration.....	406
Table 54 — Data Definition for Federated Clock Configuration	408
Table 55 — Load Time Configuration Acknowledge Payload Field Definition	409
Table 56 — Reply AGT Payload Field Definition	410
Table 57 — Start_ALT_Synchro Payload Field Definition.....	412
Table 58 — Reply ALT Payload Field Definition.....	413
Table 59 — Reply AGT ALT Payload Field Definition	414
Table 60 — Load Network Configuration Payload Field Definition	416
Table 61 — NSM Switch Command Field Encoding	418
Table 62 — Load Network Configuration Acknowledge Payload Field Encoding.....	419
Table 63 — Load Network Configuration Payload Field Encoding	420
Table 64 — Reply Network Status Payload Field Encoding.....	421
Table 65 — Load Power Switches Configuration Payload Field Encoding.....	422
Table 66 — PCM Switch Command Field Encoding	424
Table 67 — Power Switch Configuration Acknowledge Payload Field Encoding	425
Table 68 — Reply Power Switches Status Payload Field Encoding	426
Table 69 — IDL Basic Integer Types.....	439
Table 70 — Interfaces Compliancy Matrix	487
Table 71 — Service Compliancy Matrix.....	488
Table A.1 — ASAAC Graphics Language	497
Table A.2 — Keys Referred to in Table A.1	501
Table A.3 — Auxiliary Functions.....	501
Table A.4 — Video Library Functions.....	502
Table A.5 — Texture Formats.....	503

Foreword

This document (EN 4660-005:2011) has been prepared by the Aerospace and Defence Industries Association of Europe - Standardization (ASD-STAN).

After enquiries and votes carried out in accordance with the rules of this Association, this Standard has received the approval of the National Associations and the Official Services of the member countries of ASD, prior to its presentation to CEN.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by November 2011, and conflicting national standards shall be withdrawn at the latest by November 2011.

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

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

0 Introduction

0.1 Purpose

This document is produced under contract ASAAC Phase II Contract n°97/86.028.

The purpose of the ASAAC Programme is to define and validate a set of open architecture standards, concepts & guidelines for Advanced Avionics Architectures (A3) in order to meet the three main ASAAC drivers. The standards, concepts and guidelines produced by the Programme are to be applicable to both new aircraft and update programmes from 2005.

The three main goals for the ASAAC Programme are:

1. Reduced life cycle costs,
2. Improved mission performance,
3. Improved operational performance.

The ASAAC standards are organised as a set of documents including:

- A set of agreed standards that describe, using a top down approach, the Architecture overview to all interfaces required to implement the core within avionics system,
- The guidelines for system implementation through application of the standards.

The document hierarchy is given hereafter: *(in this figure the document is highlighted)*

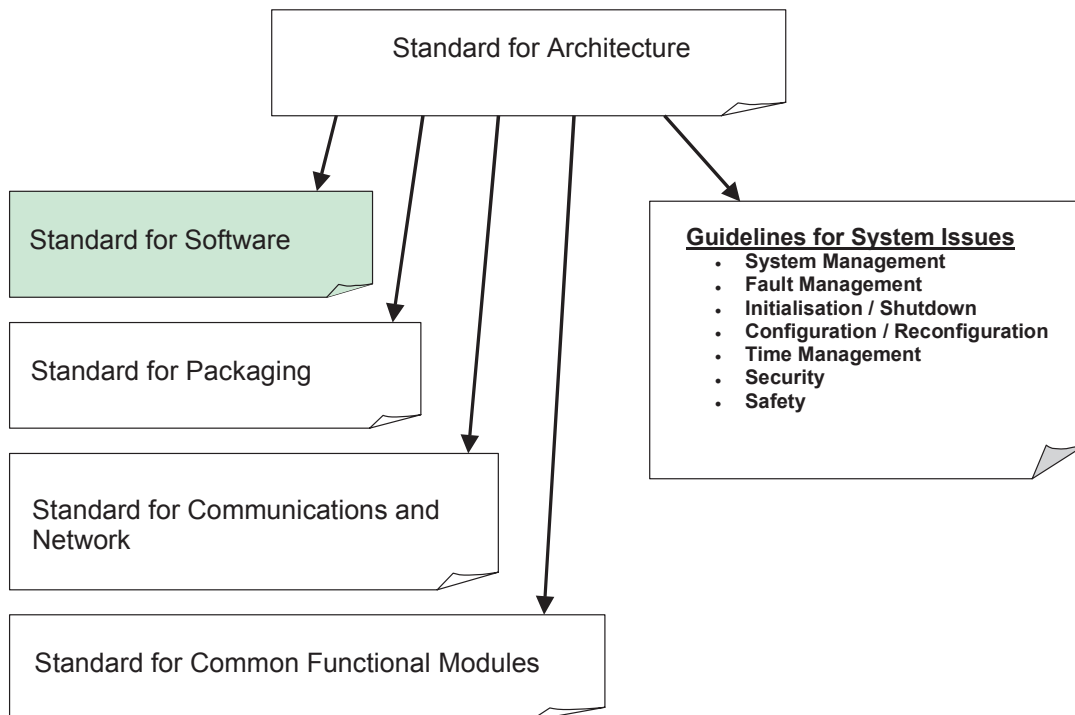


Figure 1 — ASAAC Standard Documentation Hierarchy

0.2 Document structure

The document contains the following sections:

Clause 1, Scope,

Clause 2, Normative references,

Clause 3, Terms, definitions and abbreviations,

Clause 4, System Functions,

Clause 5, Software Architecture Definition,

Clause 6, Direct Interfaces,

Clause 7, Logical Interfaces Definitions,

Clause 8, Data Type Definitions,

Clause 9, Tailoring,

Annex A, AGL.

1 Scope

The purpose of this European Standard is to establish uniform requirements for design and development of software architecture for modular avionics systems as defined per ASAAC.

1.1 Software Architecture Overview

The ASAAC Software Architecture is based on a three-layer stack as shown by a simplified Figure 2.

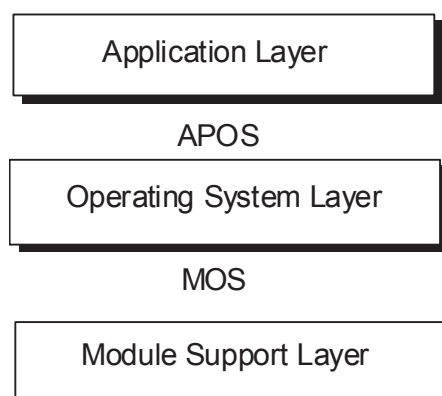


Figure 2 — ASAAC Three Layer Software Architecture

Each layer is described in terms of its dependency/independency on both the aircraft system and the underlying hardware.

Table 1 — Software Layer Independence

Software Layer	Aircraft Dependency	Hardware Dependency
Application Layer (AL)	Dependent	Independent
Operating System Layer (OSL)	Independent	Independent
Module Support Layer (MSL)	Independent	Dependent

1.2 Software Architectural Components

Figure 3 provides an overview of the software architectural components and software interfaces.

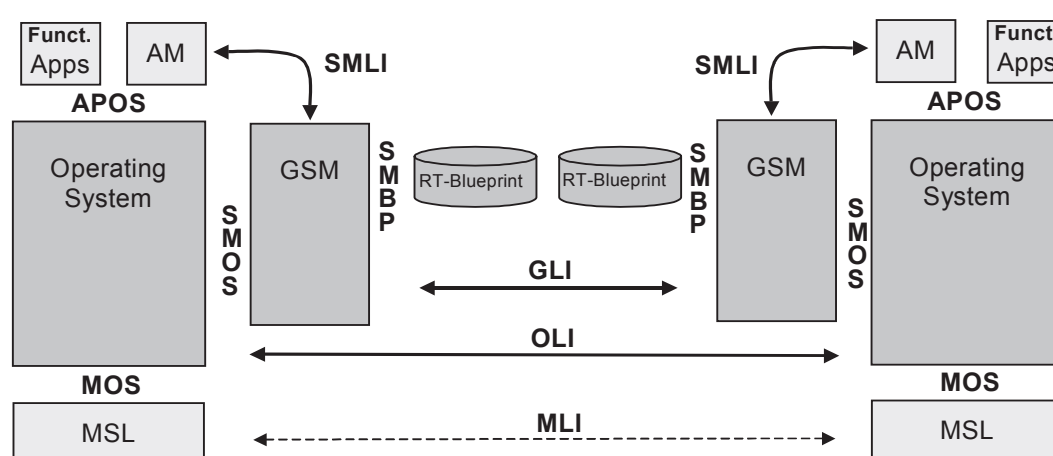


Figure 3 — The Software Architecture Model

1.2.1 Functional Applications

The term "Functional Applications" relates to all functions that handle the processing of operational data, e.g.

- Radar Applications,
- Mission Management,
- Stores Management,
- Vehicle Management System,
- Communication, Navigation and Identification.

1.2.2 Application Management (AM)

AM is responsible for the non-standardised system management, i.e. the AM performs the non-generic system management. As an example, the AM may perform the mission/moding management. The interface between the AM and GSM is the System Management Logical Interface (SMLI) (see 4.1.2).

1.2.3 Operating System (OS)

A Real-Time OS provides the particular part of OSL functionality that controls the real-time behaviour of the Processing Element and its associated resources (see Clause 0).

1.2.4 Generic System Management (GSM)

The GSM is responsible for the management of the core processing (see 4.1.1 and 5.2.1). This functionality is divided into four areas:

- Health Monitoring,
- Fault Management,
- Configuration Management,
- Security Management.

1.2.5 Run-Time Blueprints (RTBP)

The RTBP contain the information (e.g. process description, routing information, fault management data) required to configure and manage the core processing on which it is hosted (see 5.3).

1.2.6 Module Support Layer (MSL)

The MSL encapsulates the details of the underlying hardware and provides generic, technology independent access to low-level resources (see 5.1).

1.2.7 Application to OS Interface (APOS)

The APOS is a direct interface that separates the aircraft dependent software (AL) from the aircraft independent software (OSL). Its purpose is to provide the processes in the AL with a standardised OS independent interface to those services provided by the OS, thus promoting the portability and re-use of application software (see 6.1).

1.2.8 Module Support to OS Interface (MOS)

The MOS is a direct interface that separates the OSL from the hardware dependent software (MSL). Its purpose is to provide the OS with a hardware independent/technology transparent interface to the functionality contained within the MSL. The MOS therefore allows the same OSL software to reside on different implementations of a particular CFM regardless of the underlying hardware (see 6.2).

1.2.9 System Management to Blueprints Interface (SMBP)

This direct interface, encapsulated within the OSL between the GSM and the blueprints, allows the structure and implementation of the blueprints to remain non-standardised, while defining a standardised interface to them (see 6.3).

1.2.10 System Management to OS Interface (SMOS)

This direct interface, encapsulated within the OSL, describes the services provided by the OS to the GSM (see 6.4).

1.2.11 OS Logical Interface (OLI)

The OLI describes the intercommunications between two instantiations of OS's with regard to Virtual Channel (VC) communications and data presentation (see 7.1).

1.2.12 GSM Logical Interface (GLI)

The GLI describes the intercommunications between two instantiations of GSM (see Clause 0). The nature of this inter GSM communication is hierarchical.

1.2.13 System Management Logical Interface (SMLI)

The SMLI standardises a VC based communication protocol between the AM and GSM. AM and the GSM have to cooperate and to do so, they communicate and synchronise themselves via the SMLI (see 7.3).

1.2.14 Module Logical Interface (MLI)

This logical interface (communication protocol) defines the logical interactions between modules to meet the module interoperability and system buildability requirements (see 7.4).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

EN 4660-001, *Aerospace series — Modular and Open Avionics Architectures — Part 001: Architecture*

EN 4660-002, *Aerospace series — Modular and Open Avionics Architectures — Part 002: Common Functional Modules*

EN 4660-003, *Aerospace series — Modular and Open Avionics Architectures — Part 003: Communications/Network*

EN 4660-004, *Aerospace series — Modular and Open Avionics Architectures — Part 004: Packaging*

ASAAC2-GUI-32450-001-CPG Issue 01, *Final Draft of Guidelines for System Issues* ¹⁾

- *Volume 1 — System Management.*
- *Volume 2 — Fault Management.*
- *Volume 3 — Initialisation and Shutdown.*
- *Volume 4 — Configuration / Reconfiguration.*
- *Volume 5 — Time Management.*
- *Volume 6 — Security.*
- *Volume 7 — Safety.*

Common Object Request Broker Architecture: *Core Specification Version 3.0 - Editorial update formal/02-12-06, OMG.*

ISO/IEC 14977:1996(E), *EBNF specification.*

ISBN 0-201-63276-4, *Open GL Reference Manual.*

3 Terms, definitions and abbreviations

For the purposes of this document, the following terms, definitions and abbreviations apply.

3.1 Terms and definitions

Use of “shall”, “should” and “may” within the standards observe the following rules:

- The word 'SHALL' in the text expresses a mandatory requirement of the standard.
- The word 'SHOULD' in the text expresses a recommendation or advice on implementing such a requirement of the standard. It is expected that such recommendations or advice be followed unless good reasons are stated for not doing so.
- The word 'MAY' in the text expresses a permissible practice or action. It does not express a requirement of the standard.

3.2 Abbreviations

AC	Aircraft
AGL	ASAAC Graphics Tag Language
AGT	Absolute Global Time
AL	Application Layer

1) In preparation at the date of publication of this standard.

ALT	Absolute Local Time
AM	Application manager
APOS	Application to OS [interface]
ASAAC	Allied Standard Avionics Architecture Council
ATM	Asynchronous Transfer Mode
BIT	Built-In Test
BMC	Between Module Communication
CBIT	Continuous BIT
CDR	Common Data Representation
CFM	Common Functional Module
CM	Configuration Management
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
DMC	Distributed Multicast Communication
DPM	Data Processing Module
EBNF	Extended Backus-Naur Form
EW	Electronic Warfare
FC	Fibre Channel
FIFO	First In First Out
FM	Fault Management
GLI	Generic System Management Logical Interface
GSM	Generic System Management
HM	Health Monitoring
HW	Hardware
IA	Integration Area
IBIT	Initiated BIT
ID	Identification
IDL	Interface Definition Language
IF	Interface

IMA	Integrated Modular Avionics
IMC	Intra Module Communication
IPC	Intra Processor Communication
IPEC	Intra PE Communication
LC	Logical Configuration
LSB	Least Significant Byte
MC	Master Clock
MLI	Module Logical Interface
MMM	Mass Memory Module
MOS	MSL to OS [interface]
MRC	Master Reference Clock
MSB	Most Significant Byte
MSL	Module Support Layer
MSU	Module Support Unit
N/A	Not Applicable
NC	Network Channel
NII	Network Independent Interface
NIU	Network Interface Unit
NSM	Network Support Module
NW	Network
OLI	Operating System Logical Interface
OMG	Object Management Group
OS	Operating System
OSL	Operating System Layer
PBIT	Power-Up BIT
PCM	Power Conversion Module
PE	Processing Element
PSE	Power Supply Element
PU	Processing Unit

QoS	Quality of Service
RE	Resource Element
RC	Remote Clock
RF	Radio Frequency
RLT	Relative Local Time
RTBP	Runtime Blueprints
RU	Routing Unit
SCU	Switch Control Unit
SM	Security Management
SMBP	System Management to Blueprints Interface
SMLI	System Management Logical Interface
SMOS	System Management to OS Interface
SPM	Signal Processing Module
SW	Software
TC	Transfer Connection
TLS	Three Layer Stack
VC	Virtual Channel
VL	Video Library

4 System Functions

This section describes the system context and the mapping of system functions onto software architectural components.

4.1 System Management Function

NOTE For requirements on the System Management including detailed examples refer to the respective volumes of ASAAC2-GUI-32450-001-CPG Issue 01.

The System Management is responsible for managing the ASAAC system during initialisation, all operational phases in flight and on ground, and system shutdown until power-off. Thus, its tasks include:

- Control of the system initialisation, reconfiguration, and shutdown processes,
- Identification, masking, filtering, and localisation of errors,
- Provision of security related services.

The System Management is comprised of two functions located on the application and OSL's of the ASAAC Three-Layer-Stack model (Figure 3):

- The Application Management function (AM): Aircraft dependent, HW independent,
- The GSM function (GSM): Aircraft independent, HW independent.

The underlying principles of this architecture are the separation between hardware and aircraft dependent layers and the separation between avionics functions represented by functional applications and system management functions.

The System Management function is organized hierarchically on three level types (Figure 4) covering the following functionalities:

- Aircraft (AC) level,
- Integration Area (IA) level,
- Resource Element (RE) level.

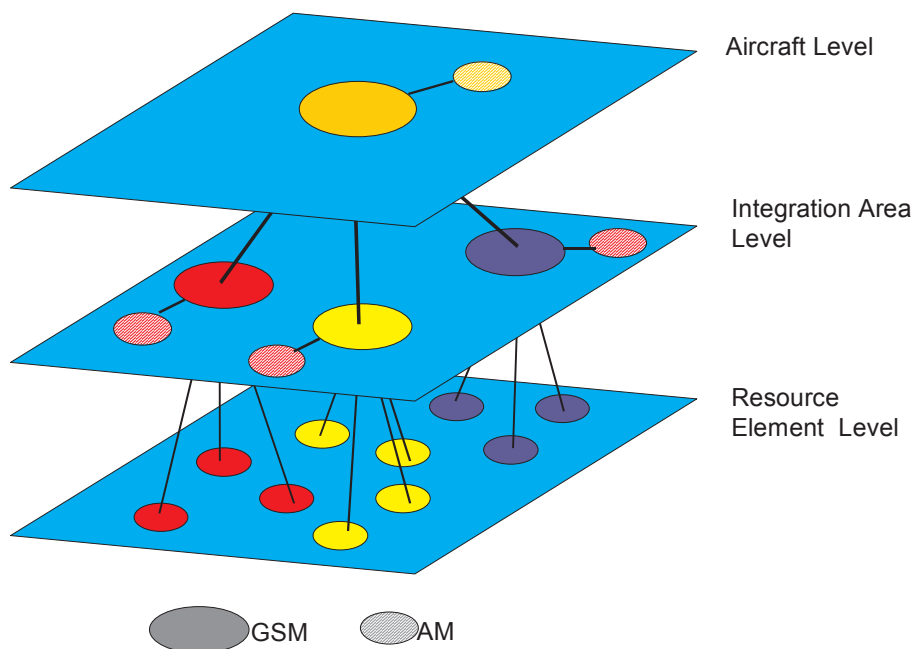


Figure 4 — Hierarchical Organisation of the System Management

Each level has dedicated characteristics:

AC Level:

- The AC level is a single system management entity responsible for controlling/monitoring the entire IMA core.

IA Level:

- The IA level is a logical grouping of closely integrated functional applications with their resources. This grouping need not be static, but may be created and deleted dynamically during a mission.
- Each IA controls one or more RE's.
- IA's may internally be organised hierarchically, i.e. a system may include one or more IA levels. In this case, the lowest-level IA must control one or more PE's.
- A system may be designed so that it does not include an IA level. In this case, there is one AC level, and one or more RE levels.

RE Level:

- The resource element is the lowest addressable level in the system management hierarchy responsible for managing the functionality of a single Processing Element (PE).

4.1.1 GSM Function

The GSM function GSM is responsible for the management and control of all resources and management of the ASAAC system behaviour via the use of RTBP (see 5.3).

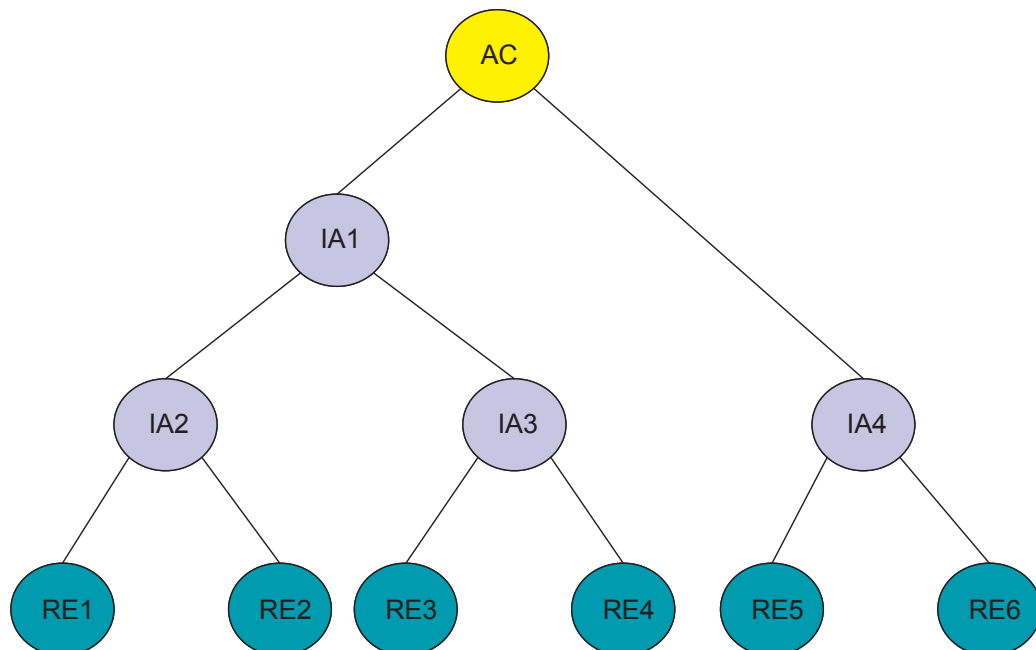


Figure 5 — GSM Decomposition for RE-Management (Example)

Functions:

The GSM comprises four functions:

- Configuration Management:
Set-up of the initial configuration, reconfiguration management.
- Health Monitoring:
Monitoring of the health status, error collection, filtering, and transmission to the FM.
- Fault Management:
Masking, filtering, and localisation of faults including the processing of corrective actions.
- Security Management:
Implementation of the system security policy: Authentication, decryption, and encryption.

Hierarchical Organisation:

The following figures illustrate possible mappings of resources to the system management hierarchy:

- RE management (Figure 5): An AC manager controlling 2 IA managers IA1 and IA4; IA1 controlling the IA managers IA2 and IA3; IA2, IA3, and IA4 each controlling 2 RE's

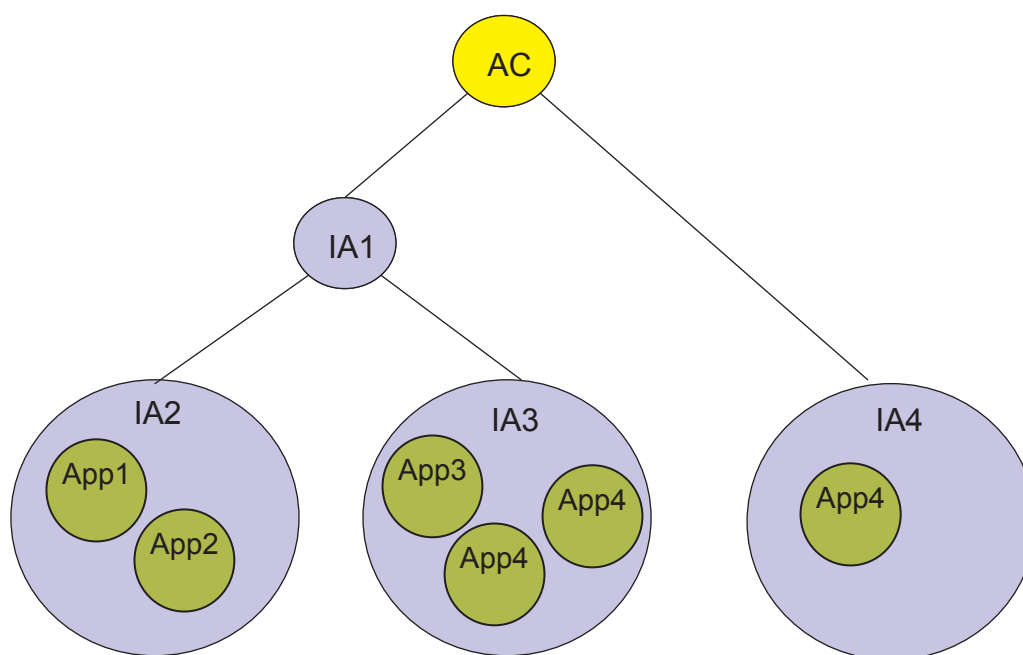


Figure 6 — IA Application Control (Example)

- Application configuration control (Figure 6): An AC manager controlling IA1 and IA4; IA1 controlling IA2 and IA3; IA2 controlling the applications App1 and App2; IA3 controlling the applications App3 and App4 (redundant); IA4 controlling the redundant application App4.

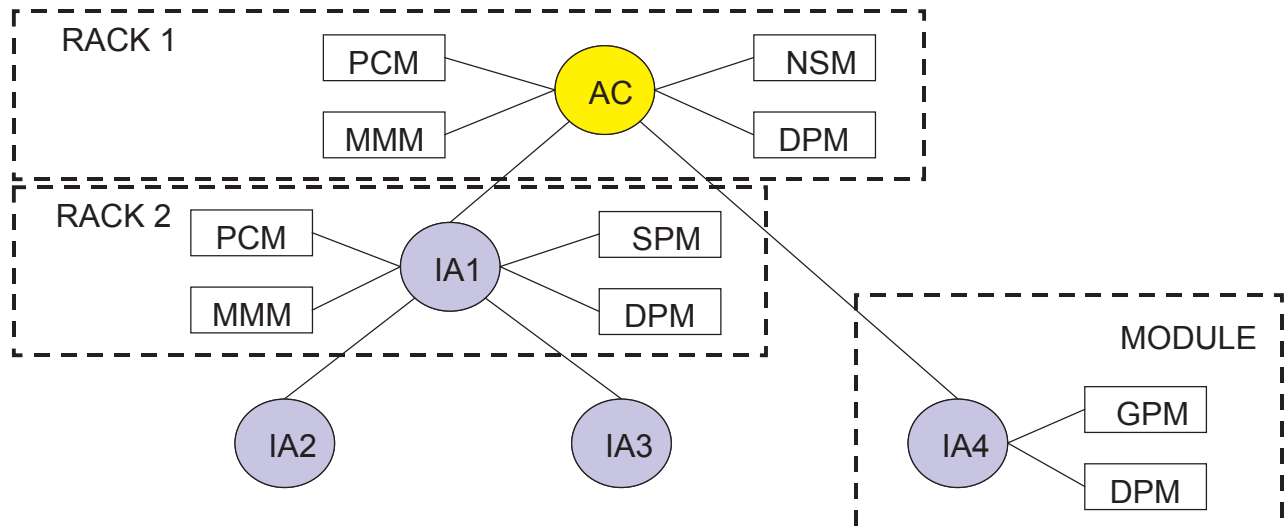


Figure 7 — GSM Decomposition for Module Management (Example)

- Application configuration control (Figure 7): An AC manager controlling IA1 and IA4; IA1 controlling IA2 and IA3; IA2 controlling the applications App1 and App2; IA3 controlling the applications App3 and App4 (redundant); IA4 controlling the redundant application App4.

Configuration Data:

The configuration data is obtained from the RTBP via SMBP. The reconfiguration is defined through dedicated sequences obtained via SMBP.

Initialisation and Shut-Down:

Initialisation and shut-down is performed on three different levels:

- Application,
- System,
- Module.

4.1.2 AM Function

The AM function is responsible for the management and control of all AC dependent functions (functional applications) on the Application Layer (AL). It acts as an interface between the functional applications and a dedicated instantiation of the GSM.

Hierarchical Organisation:

The AM should only be located on the AC- and IA-levels, as the RE level is resource-oriented, whereas the AC and IA levels are function-oriented.

An example for the hierarchical organisation of the AM showing the assignment of functional applications to IA's is depicted in Figure 8:

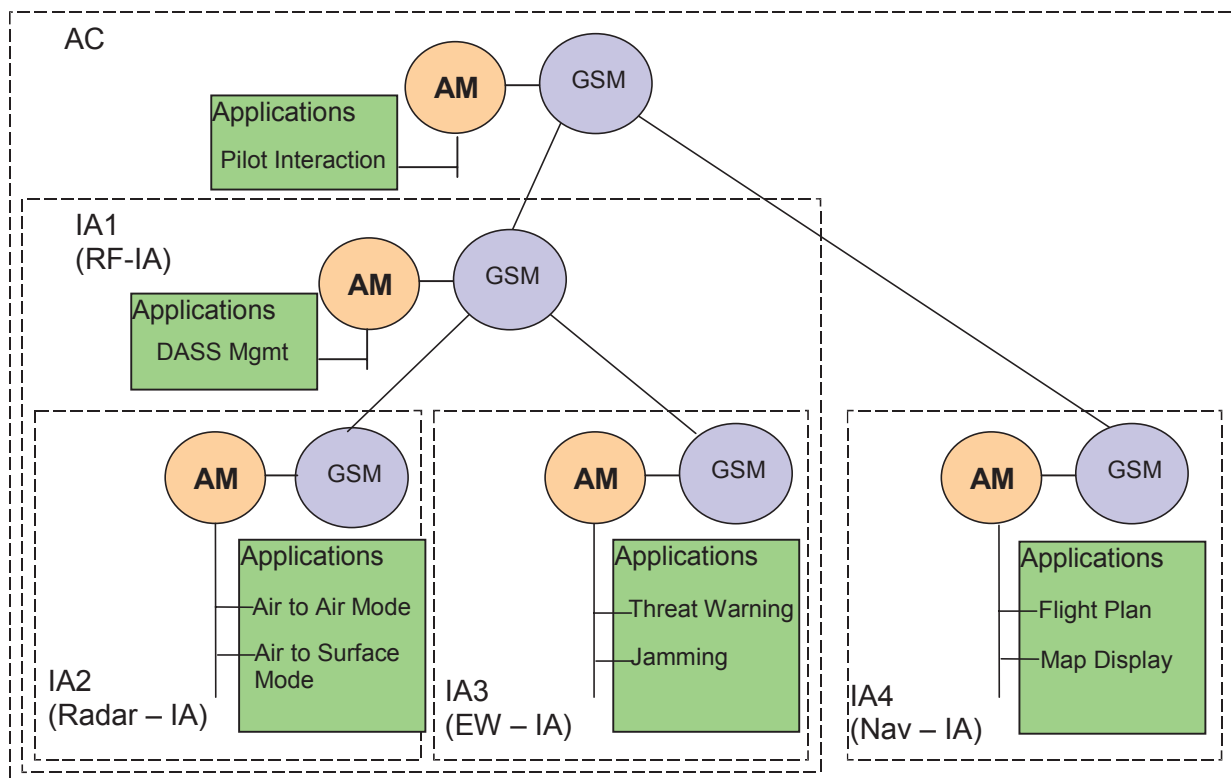


Figure 8 — Hierarchical Organisation of the AM (Example)

Internal Interfaces:

The standardised internal interface of the AM is the System Management Logical Interface (SMLI.) The SMLI includes a request-response protocol for the change of the logical configuration.

External Interfaces:

There are no standardised external interfaces of the AM. All external interfaces are application-dependent.

4.1.3 Error Handling

ASAAC compliant systems require that software developers write their functional application code to interface with the underlying OS using the standardised service calls that comprise the APOS interface (6.1). However, it is possible at run-time for an APOS service not to perform correctly and to actually return an error status to the calling Application Process. This might be due to a real fault in the underlying system or by misuse of the APOS interfaces themselves (e.g. posting a semaphore before it has been created). In either case the fault is handled through a standardised process (see ASAAC2-GUI-32450-001-CPG Issue 01) in which the precise error identification is passed to the Health Monitoring function within the GSM. Any error handling shall be subject to the decisions made by the fault management function. In handling the error, the fault management function may delegate the error handling back to a functional Application Process by invoking the error handler thread of the Application Process. In this case, the complete error information shall be accessible to this error handler thread.

The error information shall be accessible to the application itself, but used for debugging purposes only.

Exceptions to this rule are timeouts and resources, which are managed by the application. Note however that functional Application Processes shall handle situations where a called APOS service has timed out. In this case, the application calling a service shall be informed by means of a return value.

4.1.4 Built-In Test

The BIT Services provide the ability to execute module built-in tests and read their results. The built-in-test component provides access to all built-in-test routines available on the module. There are three different types of built-in test:

- Power-up built-in-test (PBIT),
- Continuous built-in-test (CBIT),
- Initiated built-in-test (IBIT).

The OS provides the GSM with Services related to the BIT Management at the SMOS interface that are paired with services at the MOS interface:

- Get PBIT Result: Retrieves the stored PBIT result,
- Start CBIT: Runs the CBIT processing and then returns. It allows a specific type of test to be run, or all tests to be run,
- Get CBIT Result: Retrieves the CBIT result,
- Start IBIT: Starts the IBIT processing,
- Get IBIT Result: Retrieves the stored IBIT result.

4.1.4.1 Power-up Built-In Test (PBIT)

PBIT is used to check the state of the module hardware as part of the boot process. The tests are run autonomously as part of the MSL before any control is applied from outside the module. The result of these tests is recorded in the MSL for retrieval by a GSM on a controlling module via the MLI. It is also available via a MOS/SMOS call to the local GSM.

4.1.4.2 Continuous Built-In Test (CBIT)

CBIT is used to continuously check the health of the module during normal operation. CBIT is non-intrusive. The tests can be run either:

- Autonomously, if no processor support is required to perform the test,
- Under the command of the GSM, if processor support is required to perform the test.

Test results can also be obtained using two mechanisms:

- Callback,
- Polled, either as part of the calling mechanism or as a separate call.

The various combinations of CBIT behaviour are described below:

Table 2 — CBIT Modes

Run method	Result Method	Behaviour
Autonomous	Callback	CBIT runs autonomously and does not require control from outside the MSL. When a test fails, the indication of this is flagged to the OSL using a callback. The service getCbitResult is then used to retrieve the detailed information about the failure.
Autonomous	Polled	CBIT runs autonomously and does not require control from outside the MSL. When a test fails, the result is stored internally in the MSL. No indication is given to the OSL. GetCbitResult is then used periodically to retrieve any failure information. If no failure has occurred, no action is taken. If a failure has occurred, the detailed information about the failure is returned.
Commanded	Callback	CBIT runs under the control of the OSL. When a test fails, the indication of this is flagged to the OSL using a callback. GetCbitResult is then used to retrieve the detailed information about the failure.
Commanded	Polled	CBIT runs under the control of the OSL. The time allowed to perform CBIT each time the service startCbit is called, is MSL specific. When a failure is detected, GetCbitResult is then used to retrieve the detailed information about the failure.

4.1.4.3 Initiated Built-In Test (IBIT)

IBIT is used to check the state of the module hardware as part of the fault management process. It performs a comprehensive test of the module in order to help during fault localisation. The tests can be run remotely under the control of a GSM on a controlling module via the MLI, or via a MOS/SMOS call (startIbit) from the local GSM when it is available. IBIT can be destructive in its operation. This means that the current configuration of the module cannot be guaranteed when the tests have been completed. Care must therefore be taken to ensure the system is not compromised when IBIT is used. Also, in the case of destructive testing, its use should be restricted to invocation via the MLI. The result of these tests is recorded in the MSL for retrieval by a GSM on a controlling module via the MLI or via a MOS/SMOS call (getIbitResult) from the local GSM if it started IBIT.

4.2 Communication

4.2.1 ASAAC Communication Model

4.2.1.1 Principle

The ASAAC Communication stack (see Figure 9) shall be supported by:

- VC's (provided by OSL),
- Transfer Connections (TC) (provided by MSL, hardware independent),
- Network Channels (NC) (provided by MSL, hardware dependent).

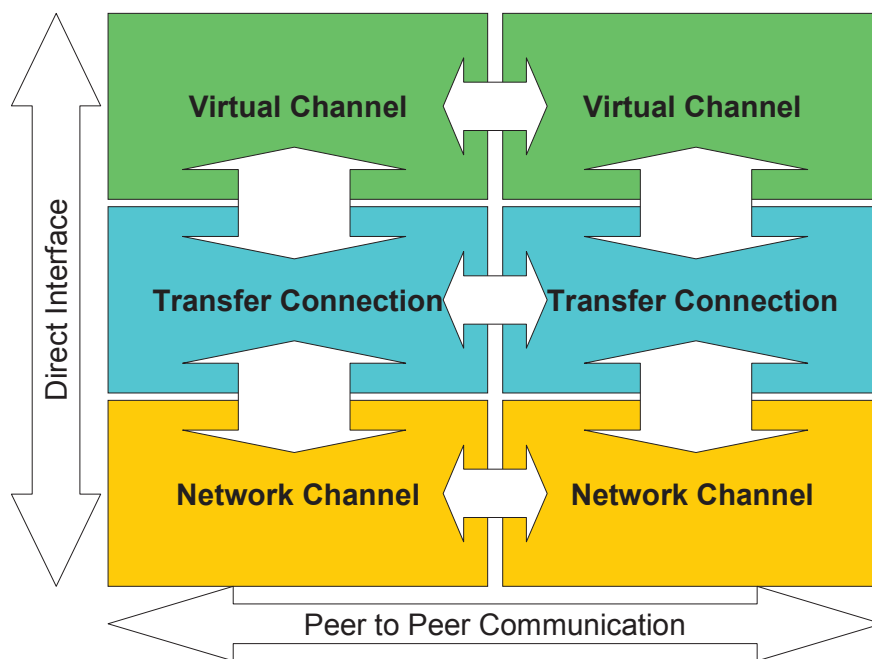


Figure 9 — The ASAAC Communication Stack

The ASAAC Communication shall support:

- One sender to one receiver (1:1),
- Multicast (one sender to n receivers (1:N), the case one sender to one receiver is a sub-set of the previous one (1:1)),
- Distributed multi-cast (applicable to signal processing applications ($M: N$)).

4.2.1.2 VC

Inter-process communication is based on VC's. VC's show the following properties:

- Unidirectional,
- Message-oriented (i.e. one message definition is assigned to a VC),
- Managed by OSL (creation, deletion, routing),
- Predictable in terms of time and resource consumption.

The concept allows a single transmitting process to send data to one or more receiving processes. A receiving process may be resident on the same Processing Element, the same CFM or even a different CFM to the sending process. The sending process has no knowledge of any receiving process; it merely outputs certain data onto a particular VC. Similarly, a receiving process has no knowledge of the sending process; it merely receives certain data from certain VC's.

The source and destination processes, the data items to be transmitted between them, and the VC's, over which they are transmitted, are defined during system design. During run-time this information is provided by the RTBP. Consequently, for a given system configuration, the set of VC's used is fixed and provides a reference against which audit data can be generated.

Each VC shall be associated with a specific message. Therefore, the following properties of messages are also associated with the VC:

- Data representation: one message shall be described in the blueprints,
- Security (encryption, decryption): the VC is marked or not,
- Modularity: an application must be seen as a message server. During application design, there is no knowledge about how many users shall be supposed to receive the message.

4.2.1.3 TC

The basic communication link offered by the NII is the Transfer Connection (TC). Data transfer via TC's has the following properties:

- The TC is unidirectional.
- The OS manages the TC in terms of creation, deletion and routing.
- A TC shall be capable to be used by either one or many VC's
- A TC supports streaming communication mode.

4.2.1.4 NC

The ASAAC Standard does not establish any requirement against the NC's. The NC's shall be managed within the MSL.

Blueprint data shall configure the properties of a NC. These are implementation dependent data and therefore shall be transparent to the OSL software.

4.2.2 Types of Data Transfer

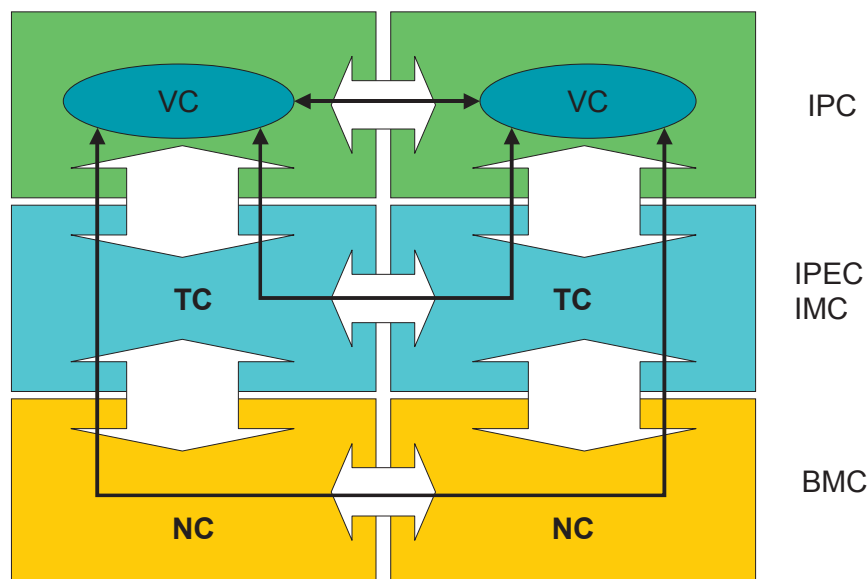


Figure 10 — Types of Data Transfer

Figure 10 shows the 4 types of communication, based on their transfer scope:

- Intra Processor Communication (IPC)
is the communication within a processor and is handled at OSL level by VC.
- Intra Processing Element Communication (IPEC)
is the communication between processors within a PE and is handled at MSL level by TC.
- Intra Module Communication (IMC)
is the communication between PE's within a Module and is handled at MSL level by TC.
- Between Module Communication (BMC)
is the communication between CFM's and is handled at OSL level by a VC and at MSL level by a TC and a NC.

4.2.3 Communication Configuration

The ASAAC Communication shall be configurable based on blueprint data (see 5.3).

The scope of a local VC Id is limited to a single Application Process. Its value is determined by the implementation of the Application Process it is associated with. This value is unique within that Application Process. It is independent of the design of the other Application Processes.

GSM uses blueprint data on both sending and receiving sides, and shall initiate the OS to create communication objects based on blueprint information in order to configure the communication:

- The OS creates an instance of a VC. A global VC Id, which is unique within the System, identifies a VC. It shall be identical on both sending and receiving sides.

- The OS attaches an Application Process local VC to a system global VC.
- The OS creates an instance of a TC. A global TC Id, which is unique within the System, identifies a TC. It shall be identical on both sending and receiving sides.
- The OS maps a VC onto an appropriate TC.

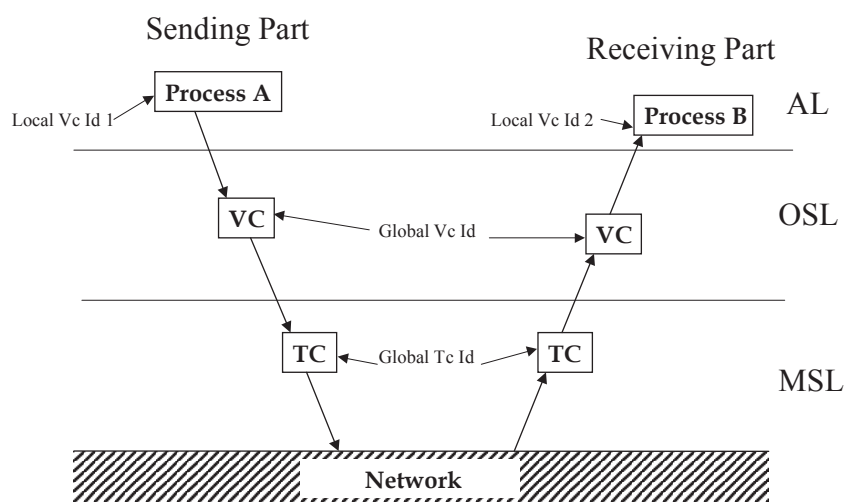


Figure 11 — Communication Concept

The definition of NC's shall be part of the TC parameter set, as NC's are associated to the definition of a TC. Therefore, NC's are not handled as a separate entity.

4.2.4 Communication Protocols

4.2.4.1 Introduction

The ASAAC Communication stack defines the protocols for establishing a communication between ASAAC layers.

4.2.4.2 VC to VC

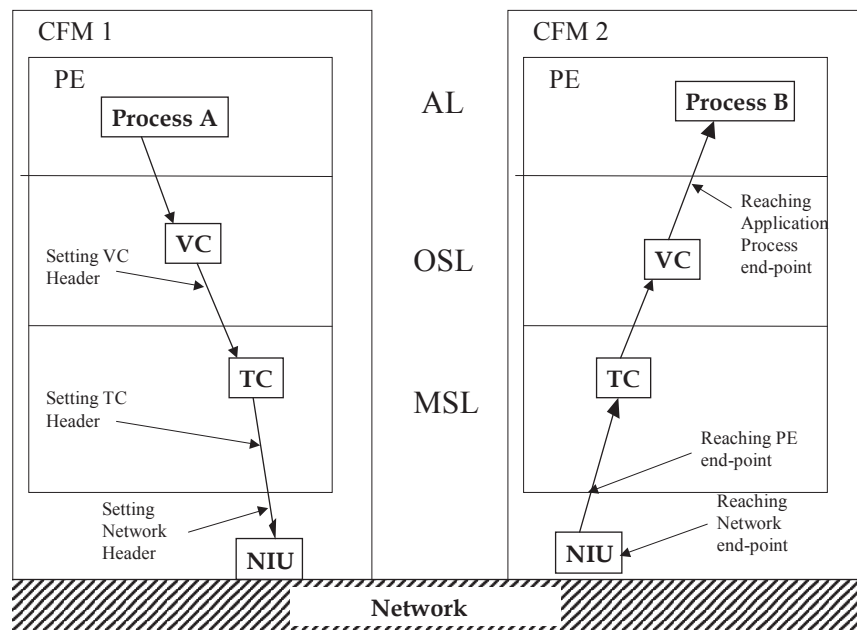


Figure 12 — Between AL Communication Routing

A communication between processes onto two different CFM's can use the following mechanisms:

- Process-to-Process: The VC allows a Process within a PE to be reached.
- PE-to-PE: The TC allows a PE within a CFM to be reached.
- CFM-to-CFM: The NC allows a CFM onto the Network to be reached.

The VC-to-VC supports communication between:

- Application processes,
- GSM's, the inter-GSM communication is a standardised logical interface, namely GLI, see Clause 0,
- Application Manager and GSM, which is a standardised logical interface, namely SMLI, see 7.3.

4.2.4.3 TC to TC

The access and control of TC's shall be at either the boundary between the OSL and MSL, namely the NIU, or within the MSL.

As described above, TC-to-TC communication supports BMC, IMC and IPEC communication. The TC header is necessary for the IPEC and IMC. The TC and Network headers are both necessary for BMC.

It shall be noticed that the inter-MSL communication is a standardised logical interface, namely MLI, see 7.4.

4.2.4.4 Raw VC Transfer

Raw VC transfer is needed when an Application Process communicates with an MSL on another PE or with non-core devices, an example of this is the Graphics Management, see 4.6.

An application sending a raw VC:

- The sending Process uses a VC, which is mapped onto a TC,
- The APOS VC message is provided as it is, without attaching a VC header,
- A CFM or a non-core device will receive the TC. It will be decoded as required.

When a raw VC is received:

- The CFM or non-core device as required encodes the TC payload,
- A raw VC is identified by means of its properties as defined by the blueprint VC information,
- When recognised by the OS as a raw VC, it shall be processed without removing the VC header,
- The usage of a raw VC implies that every TC carrying a raw VC is restricted to a single VC.

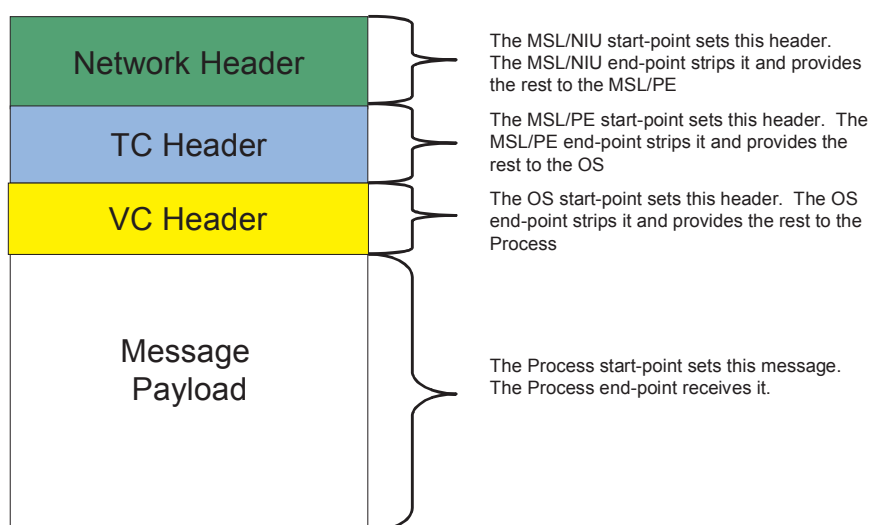
4.2.4.5 Routing Headers

In order to route a message between start-point and end-point, information is needed:

- A header that contains information on its features,
- An identifier that describes each message layer. This identifier allows the identification of the routing information in internal tables.

Each header shall provide information to reach an end-point (see Figure 13):

- The Network Header provides information to reach the targeted CFM,
- As several TC's may use the NC, the TC Header shall identify the TC. This identification allows the targeted PE to be reached,
- As several VC's may use the TC, the VC Header shall identify the VC. This identification allows the targeted Process to be reached.

**Figure 13 — ASAAC Message in BMC Data Transfer**

The layer within the ASAAC Communication stack on which the transfer is happening determines the set of routing data required. E.g. for IPC only a VC header is sufficient, if only VC communication functions are involved.

The TC Header and the VC Header are standardised. Table 3 defines their applicability to the data transfer modes. The Network header is not standardised, as it is technology dependent.

Table 3 — Routing Information and Data Transfer

Data Transfer	Network	TC	VC
BMC	Yes	Yes	Yes
IMC	No	Yes	Yes
IPEC	No	Yes	Yes
IPC	No	No	Yes

4.2.5 Multicast

The ASAAC Communication shall support the Multicast communication. This capability allows a sender to be connected to many receivers.

The multicast capability shall be supported at VC and TC level of the ASAAC Communication stack.

As there are multiple receivers, this possibly leads to a combination of several types of data transfer (IPC, IPEC, IMC and BMC) at the same time.

The VC Multicast has a unique identifier. The sender process has no knowledge of its Multicast behaviour. In IPC that VC shall be attached to all receiver processes.

In case of the VC using TC (IPEC, IMC and BMC), there are two possibilities:

- Using a single TC providing Multicast capability,
- Using multiple simple TC's.

4.2.5.1 Single Multicast TC

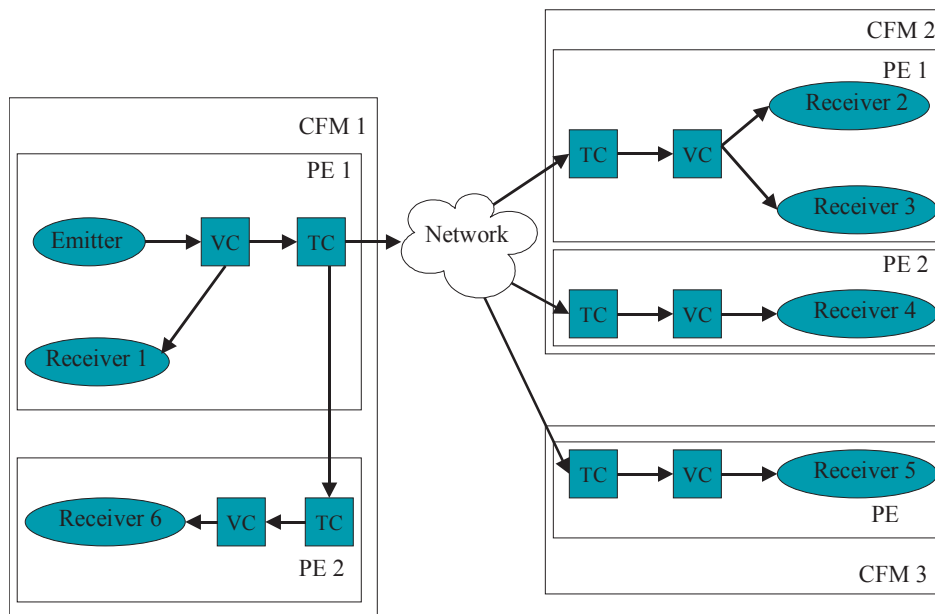


Figure 14 — Multicast Scheme With a Single TC

In this example the TC Identifier is unique. According to the figure above, the types of data transfer are (from the emitter to):

- Receiver 1: IPC, the VC is attached as outgoing to the emitter, and as incoming to the receiver,
- Receiver 2 and Receiver 3: BMC, the incoming TC is attached to a VC that incoming VC is attached to both receivers,
- Receiver 4: BMC, it shall be noticed that the TC on CFM2/PE1 and CFM2/PE2 has the same identifier. It means that the MSL shall be able to handle multiple end-points with the same identifier,
- Receiver 5: BMC,
- Receiver 6: IMC, the outgoing TC from PE1 goes to both the network and PE2. It is a case of a handling of a TC with multiple end-points, as well.

Actually, there are two methods for crossing the Network.

1. The MSL at NIU level duplicates the TC message in several messages with the Network Header corresponding to the end-point onto the Network.
2. The MSL at NIU sends a unique TC message and a Network header indicating the Multicast behaviour. Then the Network resource handles the Multicast and provides the message to the end-points.

4.2.5.2 Multiple Simple TC's

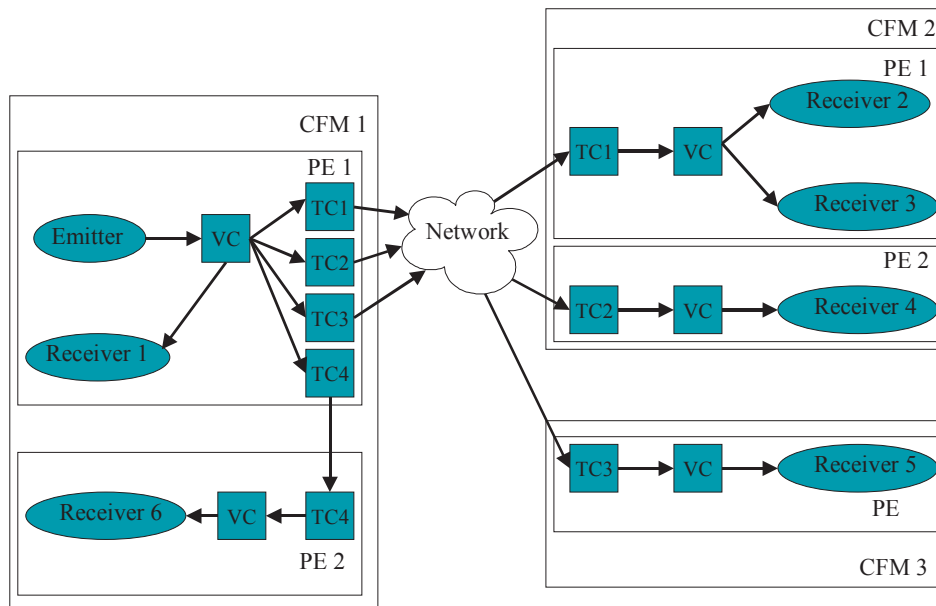


Figure 15 — Multicast Scheme With Multiple Simple TC's

At the sending side the Multicast VC is attached to several TC's. At OS level the VC Message is sent out onto several TC's.

4.2.6 Distributed Multicast

4.2.6.1 Introduction

The Distributed Multicast Communication is only applicable to Signal Processing. The Distributed Multicast Communication is an N:M communication (N senders to M receivers) using both multicasting and fragmentation.

Distributed Multicast is involved when it is necessary to use data parallelism for decreasing the time processing of a sequence of Signal Processing operations. The input data are separated in subparts that are processed in the same time by the several processors. Different processors process data, but the executed algorithm is identical.

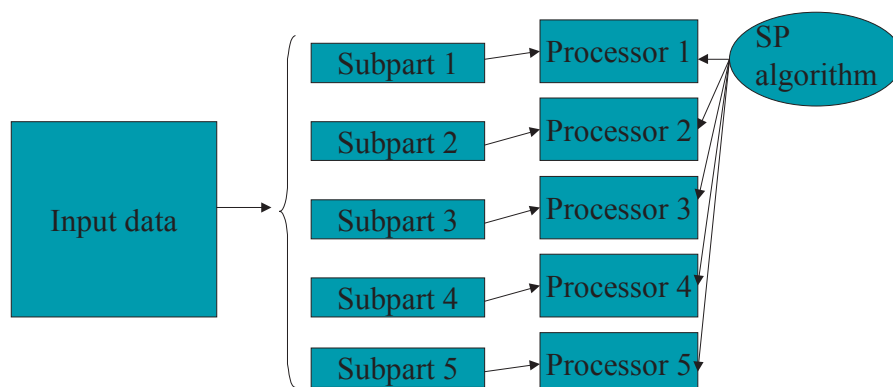


Figure 16 — Data Parallelism

The distributed multicast allows the re-distribution of data spread over several processors. One typical case for using distributed multicast is the corner turn. This function inverses the rows and columns of a table, as it is shown below.

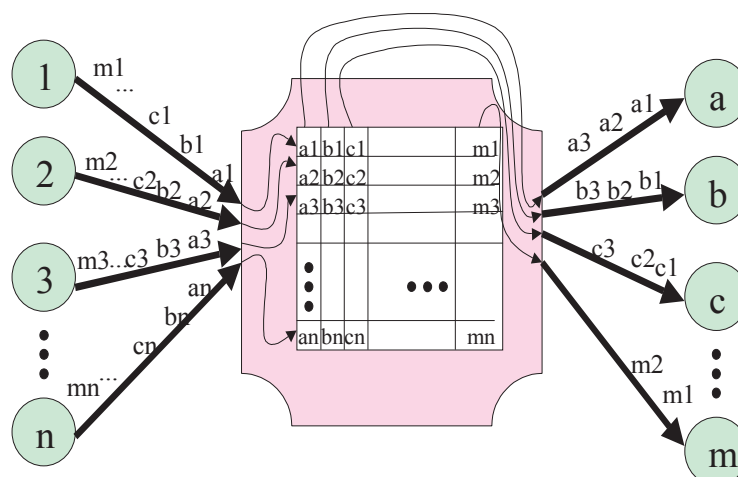


Figure 17 — Corner Turn

In the figure above, $[1..n]$ are senders and $[a..m]$ are receivers. Actually, the matrix to be inverted is virtually constructed by the communication mechanism.

Each sender shall apply a distribution law for addressing data to each receiver. A distribution law is a linear law that defines how to fragment an emitted buffer in order to generate the different buffers to be sent to each receiver.

Each receiver shall apply a collection law for addressing data from each sender. A collection is a linear law that defines how to build a buffer with the received fragmented buffers from several senders.

The N:M communication or distributed Multicast Communication may be considered from:

- The sending part as a 1:M multicast communication, where the M receivers get a different fragment,
- The receiving part as a N:1 communication, where N senders give a fragment.

The corner turn may be used in a context that needs several dimensions. Data in a radar application is organised in multi dimensional arrays. In the following radar example, a value is attached to a particular antenna beam, a particular pulse, and a particular range gate. This leads using a corner in three dimensions (pulse, range gate, beam).

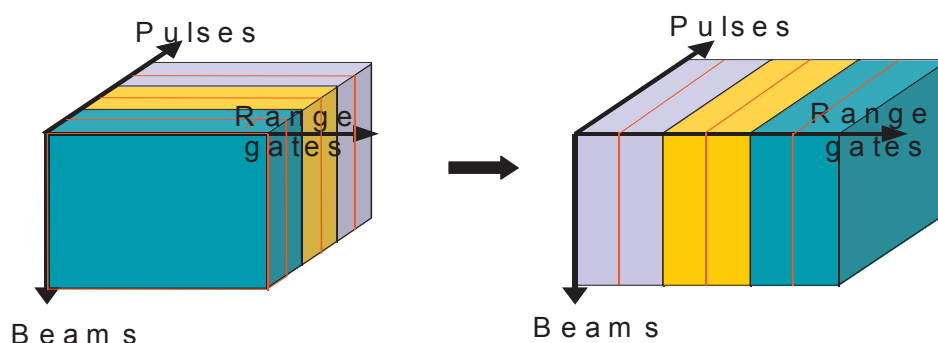


Figure 18 — Corner Turn in Three Dimensions

The slices represent a part of the cube that is processed by one processor. The slices may be overlapped.

4.2.6.2 Requirements

The distributed multicast shall provide the following features:

- A Processor involved in distributed multicast may be a sender and a receiver for this transaction,
- Each sender and receiver may be located in one or several Processing Elements, which may be located in one or several SPM,
- For each sending Processor, one distribution law per receiving Processor shall be defined,
- For each receiving Processor, one collection law per sending Processor shall be defined,
- The signal processing application shall be able to process a fragmentation in one, two or three dimensions,
- Each message sent on a TC dedicated to the Distributed Multicast Communication shall be preceded by the TC identifier for routing to the receiver, and fragment identifier for reconstitution,

4.2.6.3 Distributed Multicast Communication in TLS

The present section introduces how data are sent then received viewed from one processor.

The Application shall use a VC for sending data. This VC shall be specified as a Raw VC. This VC is attached to a TC specified as a DMC TC.

The OS shall execute the MOS service dedicated to DMC purpose: sendFragmentedTransfer.

The distribution of the DMC TC is performed within the MSL. The individual fragments shall be assigned to separate TC's. Each receiver shall be assigned a TC. The TC identifier identifies the receiver.

The receiving MSL shall collect TC's with TC Identifier specified as DMC TC, and reconstitute the message based on Fragment Identifier, which identifies the sender. When the buffer reconstitution is completed, it is retrieved by the MOS service receiveFragmentedTransfer. The MOS service may use the same TC Id as the sending point.

The OS shall execute the MOS service receiveFragmentedTransfer for passing data to the Application through a Raw VC.

Figure 19 introduces a corner turn example for DMC processing on one Processor.

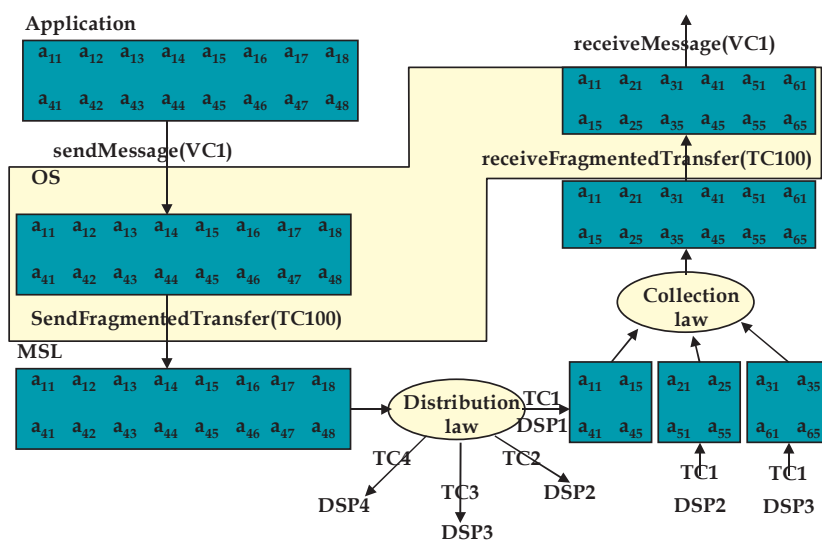


Figure 19 — Illustration of the Involved Services in DSP1

4.2.6.4 Message Fragmentation

The message fragmentation processes data selection follows a linear law that provides either a distribution or collection law depending on whether it is applied respectively on the sending or the receiving side.

In the corner turn example data addresses shall be computed according the following C-like formula:

```
for( a3 = 0; a3<N3; a3++ ) {
  for( a2 = 0; a2<N2; a2++ ) {
    for( a1 = 0; a1<N1; a1++ ) {
      DataAddress = Address0 + a1*Inc1 + a2*Inc2 + a3*Inc3 ;
    }
  }
}
```

Where:

- Address₀ = initial Address of the internal buffer where the fragmentation is started,
- N_x: Data length of the X dimension,
- a_x: Current Index of the X dimension,
- Inc_x: Increment number of the X dimension,
- x: Dimension number 1 (first dimension) 2 (second dimension) 3 (third dimension).

If the fragmentation is processed in the context of two dimensions: N₃ and Inc₃ are equal to zero.

4.2.7 Streaming

Streaming data are continuous data. Two basic applications are video or audio purposes. Streaming data shall only be handled at the TC Level. This stream of data may contain identified blocks of sub-data, e.g. video data may contain individual frames.

To allow the system to keep up with this stream, data shall be transferred from the Network Interface Unit (NIU) to the Processing Unit (PU) with the same rate as data arrived at the Network Interface unit. In case this requirement is not fulfilled, the stream is interrupted, which has to be handled as an error

For a streaming TC, the NIU will be provided with information to allow the direct transfer of data to the PU. The implementation of the MSL, which encompasses the NIU and the PU, is project dependent. Therefore, the configuration data specifying the Streaming behaviour shall be implementation dependent.

4.2.8 Data Representation

Several ASAAC components of a CFM are able to send data to components of remote CFM's. These components are in MSL, or OSL, or AL levels and may involve data representation mechanisms.

The different types of message, which may be submitted to a data representation, are:

- MSL Level:
 - TC supporting VC Messages,
 - TC supporting MLI Messages.
- OSL Level:
 - VC supporting Application, GLI and SMLI Messages,
 - VC supporting OLI Messages.
- APOS client level:
 - Application Messages as typed Messages,
 - Application Messages as raw Messages.

The data representation impacts each ASAAC layer.

4.2.8.1 MSL Level

The data representation is not configurable at MSL level. It means that all parts of a message that is under the responsibility of the MSL, are hard-coded.

- TC supporting VC Messages: The TC header structure is fixed and is known, the payload message is not modified and is given to the OS.
- TC supporting MLI Messages: The structure of all MLI Messages, header and payload message, is known. The payload message may be processed and given to the OS in the expected format. (I.e. If the CFM status is requested, it shall be replied in a format, which shall be understandable by the hosting processor). For more details, refer to MLI 7.4.

4.2.8.2 OSL Level

4.2.8.2.1 Principle

The data representation may be applied to VC Header if requested.

The data representation shall not be applied to raw messages.

The data representation shall be applied to typed message if requested.

The data representation shall not be applied to VC between processes on the same PE.

The data representation shall be based on a subset of the OMG CDR. Big-endian orientation is selected.

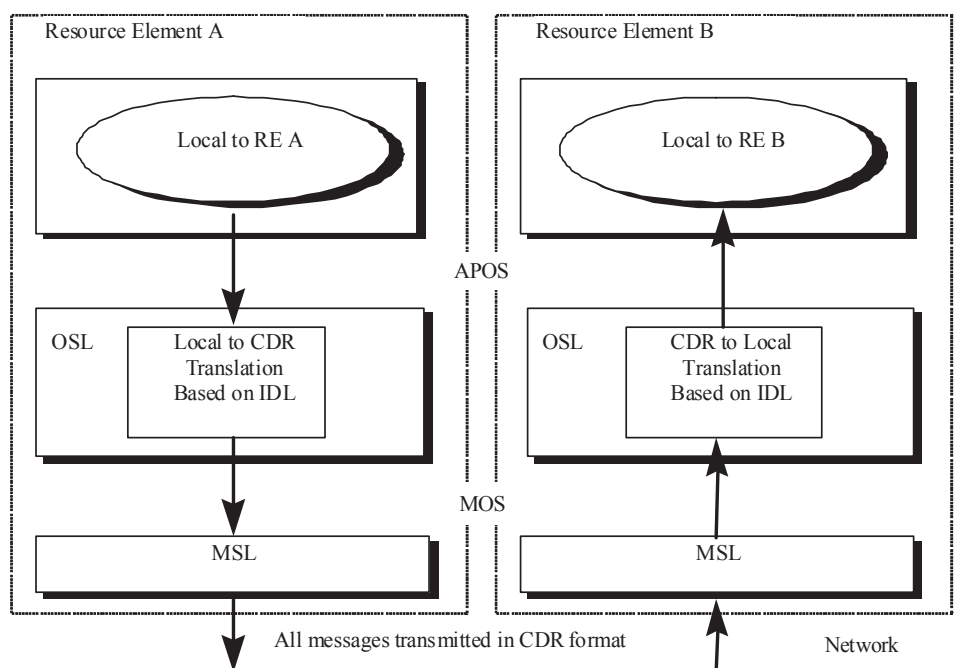


Figure 20 — Data Representation

For typed messages, the OS on information contained in blueprints does a local translation of a message. They translate the message:

- At the sending side, into the OMG CDR format. See Common Object Request Broker Architecture, 15.3,
- At the receiving side, into the format understandable by the hosting processor.

Figure 20 shows the continuation of translation actions made on a message sent from resource element A and received by resource element B.

4.2.8.2.2 Data Representation Configuration

Blueprints contain data representation information:

- A flag in the SMBP VcToTcMappingDescription structure indicates whether the data representation mechanism is required. When this flag is set, all VC's on that TC shall be processed, otherwise no processing shall be applied. In Multicast communication all receivers shall have the same features.
- The SMBP VcDescription structure:
 - A flag indicating whether the message is typed or raw,
 - If the message is specified as typed, a provider-dependent representation of the message data type in the VcDescription following the IDL message definition.

The GSM shall configure the data representation processing of the OS with the blueprint information.

4.2.8.2.3 OLI Messages

The OLI Messages are typed messages. Their description is specified in IDL format. As they are standardised, they are not described in the blueprints.

For more detail, refer to 7.1.

4.2.8.3 APOS Clients

The possible APOS clients are the processes in the AL or in the GSM. APOS clients do not manage data representation and shall not be aware of such translation actions. APOS clients send typed messages or raw messages through VC's.

4.2.8.3.1 Raw Message

For raw message (stream of bytes), application data shall send and receive without interpretation. In this case, data representation mechanisms shall not be applied.

4.2.8.3.2 Typed Message

In order to apply data representation on typed message APOS clients shall be built according to some constraints:

- Message data type definitions are based on a subset of IDL types,
- Programming languages and compiler options used for building APOS clients processes and OS components shall be compliant to generate the standardised memory alignment

4.2.8.3.2.1 Definition of Typed Messages

Message data type definitions shall be based on a subset of IDL and shall contain:

- Simple types,
- Constructed types.

The table below specifies the subset of IDL primitive types used by simple types or by constructed types.

Table 4 — IDL Primitive Types

IDL Type	Octet alignment	CDR Representation		
octet	1	Uninterpreted 8-bit		
short unsigned short	2	octet	Big-endian	
		0	MSB	
		1	LSB	
long unsigned long	4	octet	big-endian	
		0	MSB	
		1		
		2		
		3	LSB	
long long unsigned long long	8	octet	big-endian	
		0	MSB	
		1		
		2		
		3		
		4		
		5		
		6		
		7	LSB	
float	4	IEEE 754-1985		
		octet	big-endian	
		0	s	e1
		1	e2	f1
		2	f2	
		3	f3	
double	8	IEEE 754-1984		
		octet	big-endian	

IDL Type	Octet alignment	CDR Representation		
		0	s	e1
		1	e2	f1
		2	f2	
		3	f3	
		4	f4	
		5	f5	
		6	f6	
		7	f7	
long double	16	IEEE 754-1984		
		octet		big-endian
		0	s	e1
		1	e2	
		2	f1	
		3	f2	
		4	f3	
		5	f4	
		6	f5	
		7	f6	
		8	f7	
		9	f8	
		10	f9	
		11	f10	
		12	f11	
		13	f12	
		14	f13	
		15	f14	

The table below specifies the constructed types. The APOS client shall use types as described in the IDL constructed type column. The OS ensures the data representation and shall encode as it is described in the CDR Encoding column.

Table 5 — IDL Constructive Types

IDL Constructed type	CDR Encoding
struct	Components encoded in order of the definition. A structure may be composed of primitive types and constructed types.
array	Encoded array elements in sequence. The encoded element may be either a primitive type or a constructed type. An array shall be associated to a unique type of element.
sequence	Number of elements followed by the encoded sequence elements in sequence. The number of elements shall be a simple unsigned integer type. The encoded element may be either a primitive type or a constructed type. A sequence shall be associated to a unique type of element.
string	Length of the string followed by the sequence of characters encoded in single-byte form, including a null character. The length of the string shall be a simple unsigned integer type. An empty string has a length of one. The length of the string shall be a simple integer type. The encoded elements shall be an octet.
union	Discriminant tag followed by the representation of the selected member. The discriminant tag shall be a simple unsigned integer type. The selected member may be either a primitive type or a constructed type.

It is possible to merge several IDL constructed types with either a fixed size or a variable size as long as they are encapsulated in a structure and the additional information that is added by the CDR encoding is also an element of the structure. In the Data representation specification of the message, this additional information shall be a key word depending of the type of the IDL constructed types.

- Sequence: Number of element shall be a part of the structure specified by a key word,
- String: Length of string shall be a part of the structure specified by a key word,
- Union: Discriminant tag shall be a part of the structure specified by a key word.

4.3 Security Management

Security Management is the GSM function that is responsible for the secure transfer of protectively marked data throughout the system. The key responsibilities of Security Management are the:

- Management and distribution of keys for authentication and encryption as required,
- Provision of classification and clearance data to the reference monitoring function,
- Implementation of authentication and software encryption algorithms,
- Compilation of a Security Audit Log - Logging and collating.

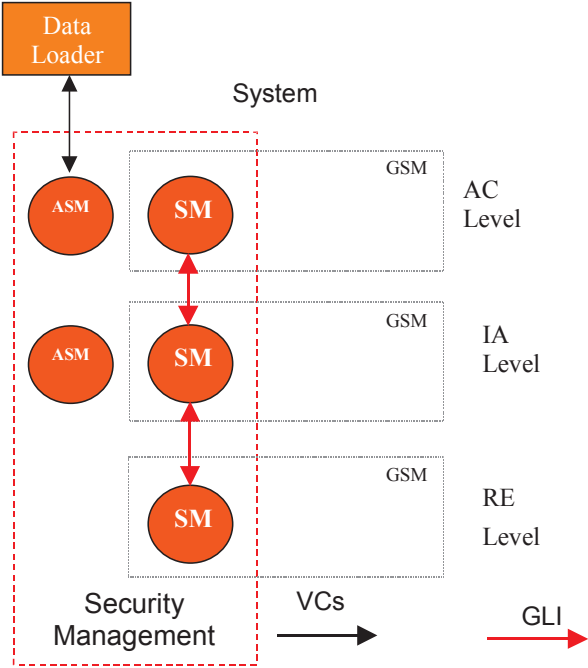


Figure 21 — GSM Interfaces

The Security Management Function is actually implemented in the form of a platform specific part, the Application Security Management (as part of AM), and the hierarchy of Generic System Security Management.

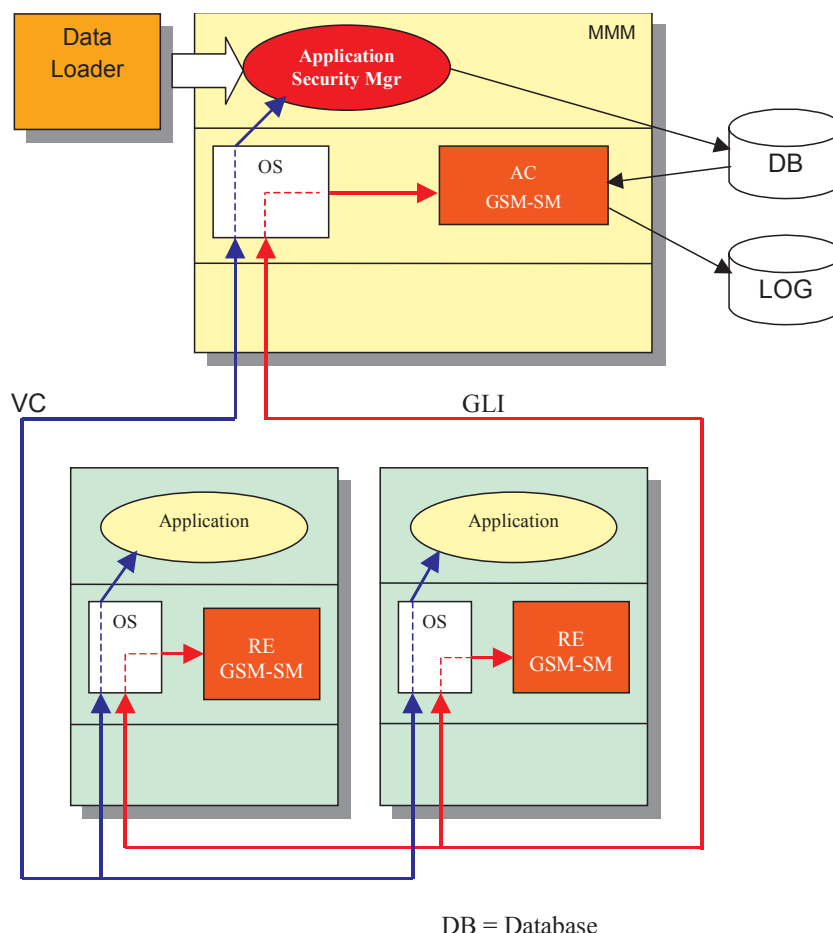


Figure 22 — Main Security Related Architectural Components

4.3.1 Application Security Management

The Application Security Management is the part of the Security Management Function that is provided specifically for a particular platform and therefore it resides above the APOS alongside the Application Manager.

Its purpose is twofold:

- To act as an interface between the data loader and the rest of the IMA System,
- To act as an interface between the database holding the protectively marked data and the functional applications which are to process the protectively marked data.

NOTE The data loader and its interface are located outside the ASAAC core.

4.3.2 Generic Security Management

The GSM-SM is that part of the Security Management Function that is applicable to all platforms.

Its purpose is twofold:

- To enable the secure transmission of protectively marked data through the use of encryption/decryption and authentication,
- To write detected attempted security breaches to the Security Log.

4.3.3 Encryption/Decryption and Authentication

The OS intercepts data being transferred or received and determines by the means of a look-up table as to whether the data being transferred is protectively marked. If it is, the OS is responsible for sending it to the GSM-SM, passing the data and the VC Id, so based on blueprint information the appropriate action can be performed on it. E.g.: Encryption, Decryption, and Authentication.

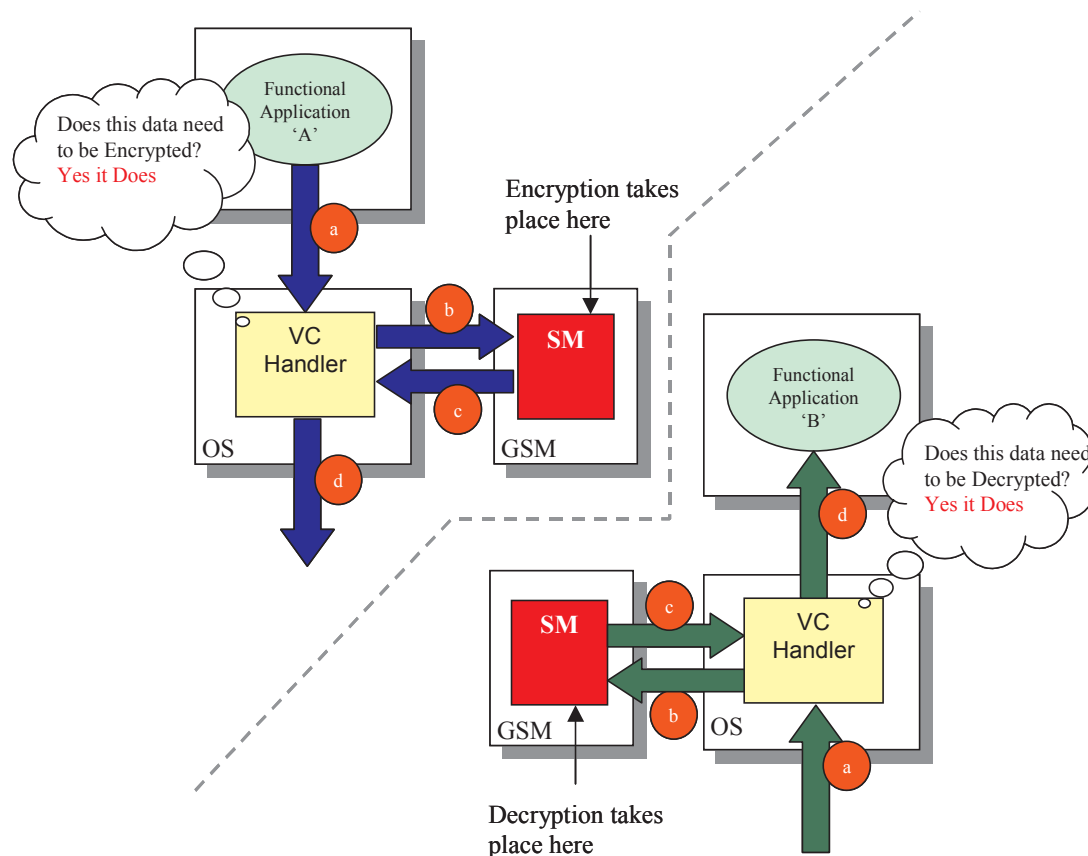


Figure 23 — VC transferring Data Requiring Encryption & Decryption

The set of protectively marked messages is dependent of system mode and defined by means of blueprints. As part of the initialisation/configuration process, the GSM-SM resident on RE's processing protectively marked data undergo a set of predefined actions. The final action of the GSM-SM in this mode changing process shall be to call the SMOS service getPMDData.