O-RAN.WG6.AAL-GAnP-R004-v10.00

Technical Specification

O-RAN Working Group 6

(Cloudification and Orchestration Work Group)

O-RAN Acceleration Abstraction Layer

General Aspects and Principles

Copyright © 2024 by the O-RAN ALLIANCE e.V.

The copying or incorporation into any other work of part or all of the material available in this specification in any form without the prior written permission of O-RAN ALLIANCE e.V. is prohibited, save that you may print or download extracts of the material of this specification for your personal use, or copy the material of this specification for the purpose of sending to individual third parties for their information provided that you acknowledge O-RAN ALLIANCE as the source of the material and that you inform the third party that these conditions apply to them and that they must comply with them.

O-RAN ALLIANCE e.V., Buschkauler Weg 27, 53347 Alfter, Germany

Register of Associations, Bonn VR 11238, VAT ID DE321720189

Contents

[List of Figures 4](#_Toc173325918)

[Foreword 5](#_Toc173325919)

[Modal verbs terminology 5](#_Toc173325920)

[1. Scope 6](#_Toc173325921)

[2. References 7](#_Toc173325922)

[2.1 Normative References 7](#_Toc173325923)

[2.2 Informative references 7](#_Toc173325924)

[3. Definitions of terms, symbols and abbreviations 9](#_Toc173325925)

[3.1 Terms 9](#_Toc173325926)

[3.2 Symbols 10](#_Toc173325927)

[3.3 Abbreviations 10](#_Toc173325928)

[4. General Aspects 11](#_Toc173325929)

[4.1 Hardware Acceleration 11](#_Toc173325930)

[4.2 AAL Architecture and concepts 12](#_Toc173325931)

[4.2.1 Overview 12](#_Toc173325932)

[4.2.2 AAL Architecture 13](#_Toc173325933)

[4.2.3 HW Accelerator Manager 15](#_Toc173325934)

[4.2.4 AAL Interfaces 16](#_Toc173325935)

[4.2.5 AAL Deployment in O-Cloud environments 18](#_Toc173325936)

[4.3 Relationship with Standards 20](#_Toc173325937)

[4.3.1 Relationship with ETSI 20](#_Toc173325938)

[5. AAL Interface definition General Principles and Requirements 22](#_Toc173325939)

[5.1 General Principles 22](#_Toc173325940)

[5.1.1 Extensibility 22](#_Toc173325941)

[5.1.2 HW Independence 22](#_Toc173325942)

[5.1.3 Interrupt and Poll Mode 22](#_Toc173325943)

[5.1.4 Discovery and Configuration 22](#_Toc173325944)

[5.1.5 Multiple AAL-LPU Support 22](#_Toc173325945)

[5.1.6 AAL offload capabilities 23](#_Toc173325946)

[5.1.7 Look-aside Acceleration Model 23](#_Toc173325947)

[5.1.8 Inline Acceleration Model 24](#_Toc173325948)

[5.1.9 AAL Application interface Concurrency and Parallelism 25](#_Toc173325949)

[5.1.10 Separation of Control and User Plane AAL Application interface APIs 25](#_Toc173325950)

[5.1.11 Support of Versatile Acceleration Payload 25](#_Toc173325951)

[5.1.12 Support of Different Transport Mechanisms 25](#_Toc173325952)

[5.1.13 AAL API namespace 26](#_Toc173325953)

[5.1.14 Chaining of AAL Profiles 26](#_Toc173325954)

[5.2 High-PHY Profile Specific Principles 28](#_Toc173325955)

[5.2.1 Separation of Cell and Slot Level Parameter Configurations 28](#_Toc173325956)

[5.2.2 SFN/slot-based Synchronization 28](#_Toc173325957)

[5.2.3 Compatibility with O-RAN FH interface 29](#_Toc173325958)

[5.2.4 Inline Profile for High-PHY Stack 29](#_Toc173325959)

[6. AAL-LPU Principles 30](#_Toc173325960)

[6.1 Overview 30](#_Toc173325961)

[6.2 LPU Deployment and Operation 31](#_Toc173325962)

[6.2.1 Example AAL-LPU Mapping 31](#_Toc173325963)

[6.2.2 Statistics 34](#_Toc173325964)

[6.2.3 Memory Management 34](#_Toc173325965)

[6.2.4 Run Time Configurations 34](#_Toc173325966)

[6.2.5 AAL Profile(s) offload, processing status query and processed data retrieval 35](#_Toc173325967)

[6.2.6 AAL-LPU Exposure 35](#_Toc173325968)

[7. AAL Profiles 38](#_Toc173325969)

[7.1 O-DU AAL Profiles 38](#_Toc173325970)

[7.1.1 O-DU Protocol Stack Reference 38](#_Toc173325971)

[7.1.2 O-DU Protocol Stack Reference for mMTC 42](#_Toc173325972)

[7.1.3 O-DU AAL Profile Definitions 46](#_Toc173325973)

[7.2 O-CU AAL Profiles 69](#_Toc173325974)

[Annex (informative): Change History 70](#_Toc173325975)

List of Figures

[Figure 4.1-1: Example illustration of the effect of hardware acceleration on functional compute performance. 12](#_Toc173325976)

[Figure 4.2.2-1: High Level AAL Architecture Diagram 14](#_Toc173325977)

[Figure 4.2.2-2: AAL Resource Relationship and Cardinality 15](#_Toc173325978)

[Figure 4.2.3-1 Example HAM software deployment scenarios 16](#_Toc173325979)

[Figure 4.2.4-1: AAL Application Common and profile APIs 17](#_Toc173325980)

[Figure 4.2.5-1: Accelerator APIs/Libraries in Container (left) and Virtual Machine Implementations (right) 18](#_Toc173325981)

[Figure 4.2.5.1-1: Example AAL Implementation Software deployment contained within O-Cloud Platform Software only 19](#_Toc173325982)

[Figure 4.2.5.1-2: Example AAL Implementation Software Deployment split between O-Cloud Platform Software and NF Deployment 19](#_Toc173325983)

[Figure 4.2.5.1-3: Example AAL Implementation Software Deployment split between O-Cloud Platform Software and NF Deployment in a virtual machine environment 20](#_Toc173325984)

[Figure 5.1.5-1: Logical Representation of AAL Application interface support for multiple AAL-LPUs 23](#_Toc173325985)

[Figure 5.1.7-1: AAL Application interface look-aside acceleration model - Data flow 23](#_Toc173325986)

[Figure 5.1.8-1: AAL Application interface inline acceleration model - Data flow 24](#_Toc173325987)

[Figure 5.1.8-2: User plane dataflow paths in look-aside and inline acceleration architectures. 25](#_Toc173325988)

[Figure 5.1.14-1: Data flow through unchained AAL Profiles 26](#_Toc173325989)

[Figure 5.1.14-2: Data flow through chained AAL Profiles 27](#_Toc173325990)

[Figure 5.1.14-3: Dataflow through chained lookaside HW Accelerator for consecutive Hi-PHY functions. 28](#_Toc173325991)

[Figure 5.1.14-4: Dataflow through chained lookaside HW Accelerator for consecutive and non-consecutive PHY. 28](#_Toc173325992)

[Figure 5.2.4-1: Partial Inline Model for AAL Hi-PHY Profile 29](#_Toc173325993)

[Figure 6.2.1-1: Scenario 1: A single AAL LPU exposes a single AAL Profile Queue used by a single AAL Application. 31](#_Toc173325994)

[Figure 6.2.1-2: A single HW Accelerator supporting two LPU's each assigned to individual AAL Applications 32](#_Toc173325995)

[Figure 6.2.1-3: Two HW Accelerators each supporting a single AAL-LPU assigned to a single AAL Application. 32](#_Toc173325996)

[Figure 6.2.1-4: A single AAL LPU exposes two AAL Profile Queues used by a single AAL Application. 33](#_Toc173325997)

[Figure 6.2.1-5: A single AAL LPU supporting two AAL Profile Instances exposes two AAL Profile Queues used by a single AAL Application. 33](#_Toc173325998)

[Figure 6.2.1-6: AAL-LPU Mapping example showing chained profile support 34](#_Toc173325999)

[Figure 6.2.6.2.1-1: Example AAL-LPU and profile supported 35](#_Toc173326000)

[Figure 6.2.6.2.1-2: Example HW Accelerator configuration 36](#_Toc173326001)

[Figure 6.2.6.2.1-3: Example assignment of AAL-LPUs and supported profiles to POD 36](#_Toc173326002)

[Figure 6.2.6.2.2-1: Example AAL Application configured AAL-Profile-Instances 37](#_Toc173326003)

[Figure 7.1.1-1: O-DU PHY processing blocks for 5G NR Downlink 38](#_Toc173326004)

[Figure 7.1.1-2: O-DU PHY processing blocks for 5G NR Uplink 40](#_Toc173326005)

[Figure 7.1.2-1: O-DU PHY processing blocks for mMTC Downlink. 43](#_Toc173326006)

[Figure 7.1.2-2: O-DU PHY processing blocks for mMTC Uplink 45](#_Toc173326007)

[Figure 7.1.3.2.1-1: AAL\_MU-MIMO\_PRECODER\_WEIGHTS\_CALC 47](#_Toc173326008)

[Figure 7.1.3.2.1-2: Example AAL\_MU-MIMO\_PRECODER\_WEIGHTS\_CALC use. 47](#_Toc173326009)

[Figure 7.1.3.2.2-1: AAL\_FFT 48](#_Toc173326010)

[Figure 7.1.3.2.2-2: AAL\_FFT example for SRS Processing 49](#_Toc173326011)

[Figure 7.1.3.3.1-1: AAL\_PDSCH\_FEC Profile 50](#_Toc173326012)

[Figure 7.1.3.3.2-1: AAL\_PDSCH\_HIGH-PHY Profile 51](#_Toc173326013)

[Figure 7.1.3.3.3-1: AAL\_PDCCH\_HIGH-PHY Profile 52](#_Toc173326014)

[Figure 7.1.3.3.4-1: AAL\_PBCH\_HIGH-PHY Profile 53](#_Toc173326015)

[Figure 7.1.3.3.5-1: AAL\_CSI-RS\_HIGH-PHY Profile 54](#_Toc173326016)

[Figure 7.1.3.3.6-1: AAL\_PT-RS-DL\_HIGH-PHY Profile 55](#_Toc173326017)

[Figure 7.1.3.3.7-1AAL\_DOWNLINK\_ HIGH-PHY Profile 56](#_Toc173326018)

[Figure 7.1.3.4.1-1: AAL\_PUSCH\_FEC Profile 57](#_Toc173326019)

[Figure 7.1.3.4.2-1 : AAL\_PUSCH\_HIGH-PHY Profile 58](#_Toc173326020)

[Figure 7.1.3.4.3.1-1: AAL\_PUCCH\_HIGH-PHY Profile (PUCCH format 0) 59](#_Toc173326021)

[Figure 7.1.3.4.3.2-1: AAL\_PUCCH\_HIGH-PHY Profile (PUCCH format 1) 60](#_Toc173326022)

[Figure 7.1.3.4.3.3-1: AAL\_PUCCH\_HIGH-PHY Profile (PUCCH format 2/3/4) 61](#_Toc173326023)

[Figure 7.1.3.4.4-1: AAL\_PRACH\_HIGH-PHY Profile 62](#_Toc173326024)

[Figure 7.1.3.4.5-1: AAL\_SRS\_HIGH-PHY Profile 63](#_Toc173326025)

[Figure 7.1.3.4.6-1: AAL\_PT-RS-UL\_HIGH-PHY profile 64](#_Toc173326026)

[Figure 7.1.3.4.7-1: AAL\_UPLINK\_ HIGH-PHY Profile 65](#_Toc173326027)

[Figure 7.1.3.5.1-1: AAL\_NPDSCH\_FEC Profile 66](#_Toc173326028)

[Figure 7.1.3.5.2-1: AAL\_NPDCCH\_FEC Profile 67](#_Toc173326029)

[Figure 7.1.3.5.3-1: AAL\_NPBCH\_FEC Profile 68](#_Toc173326030)

[Figure 7.1.3.5.4-1: AAL\_NPUSCH\_FEC Profile 69](#_Toc173326031)

Foreword

This Technical Specification (TS) has been produced by W6 of the O-RAN Alliance.

The content of the present document is subject to continuing work within O-RAN and may change following formal O-RAN approval. Should the O-RAN Alliance modify the contents of the present document, it will be re-released by O-RAN with an identifying change of version date and an increase in version number as follows:

version xx.yy.zz

where:

xx: the first digit-group is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc. (the initial approved document will have xx=01). Always 2 digits with leading zero if needed.

yy: the second digit-group is incremented when editorial only changes have been incorporated in the document. Always 2 digits with leading zero if needed.

zz: the third digit-group included only in working versions of the document indicating incremental changes during the editing process. External versions never include the third digit-group. Always 2 digits with leading zero if needed.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the O-RAN Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in O-RAN deliverables except when used in direct citation.

# Scope

This document defines O-RAN O-Cloud hardware accelerator interface functions and protocols for the O-RAN AAL interface. The document studies the functions conveyed over the interface, including configuration and management functions, procedures, operations and corresponding solutions, and identifies existing standards and industry work that can serve as a basis for O-RAN work.

# References

## Normative References

References are either specific (identified by date of publication and/or edition number or version number) or non‑specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, O-RAN cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

1. O-RAN WG1 Architecture Description
2. O-RAN WG1 OAM Architecture
3. Void
4. ETSI GS NFV-IFA 002: "Network Functions Virtualization (NFV) Release 2; Acceleration Technologies; VNF Interfaces Specification"

1. [ETSI](https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/019/03.01.01_60/gs_NFV-IFA019v030101p.pdf) GS NFV-IFA 019: "Network Function Virtualization (NFV); Acceleration Technologies; Acceleration Resource Management Interface Specification; Release 3"

1. [5G; NR; Physical Channels and Modulation 3GPP TS 38.211 v15.2.0 Release 15](https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/15.02.00_60/ts_138211v150200p.pdf)
2. [5G; NR; Multiplexing and Channel Coding 3GPP TS 38.212 v15.2.0 Release 15](https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/15.02.00_60/ts_138211v150200p.pdf)
3. [LTE; E-UTRA Physical Channels and Modulation 3GPP TS 36.211 v15.2.0 Release 15](https://www.etsi.org/deliver/etsi_TS/136200_136299/136211/15.02.00_60/ts_136211v150200p.pdf)
4. [LTE; E-UTRA Multiplexing and Channel Coding 3GPP TS36.212 v15.2.1 Release 15](https://www.etsi.org/deliver/etsi_ts/136200_136299/136212/15.02.01_60/ts_136212v150201p.pdf)
5. ETSI GS [NFV-IFA 001](https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-IFA%20001v1.1.1%20-%20GS%20-%20Acceleration%20-%20UCs%20report.pdf): "Network Functions Virtualisation (NFV); Acceleration Technologies; Report on Acceleration Technologies & Use Cases"
6. O-RAN WG6 O2 General Aspects and Principles
7. O-RAN WG6 Cloudification and Orchestration Use Cases and Requirements for O-RAN Virtualized RAN
8. ETSI GS NFV-IFA [011](https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-IFA%20011v4.3.1%20-%20GS%20-%20VNF%20Packaging%20Spec.pdf): "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; VNF Descriptor and Packaging Specification"
9. ETSI GR NFV-IFA 046: "Network Functions Virtualisation (NFV) Release 5; Architectural Framework; Report on NFV support for virtualisation of RAN"
10. O-RAN WG6 AAL Common API R003
11. O-RAN WG4 Control, User and Synchronization (CUS) Plane Specification

## Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non‑specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, O-RAN cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document, but they assist the user with regard to a particular subject area.

1. [Vocabulary for 3GPP Specifications (TR21.905)](https://www.etsi.org/deliver/etsi_tr/121900_121999/121905/04.05.00_60/tr_121905v040500p.pdf)

# Definitions of terms, symbols and abbreviations

## Terms

For the purpose of this document the terms and definitions given in O-RAN WG6 Cloudification and Orchestration Use Cases (UC) and Requirements for O-RAN Virtualized RAN [13], ETSI GS NFV-IFA 002 [4], and the following apply:

**Hardware (HW) Accelerator (HWA)** is a specialized HW implementation that can offload processing from application(s) running on a General-Purpose Processor. The Hardware (HW) Accelerator is a physical Managed Element as defined in [13].

NOTE: Examples of Hardware Accelerators include ASIC, FPGA, DSP and GPU.

NOTE: Throughout this document, the term “Accelerator” and “Hardware (HW) accelerator” are used interchangeably.

**Acceleration Abstraction Layer (AAL)** specifies a common and consistent set of interfaces used by different entities to enable interaction with different types of HW Accelerators within an O-Cloud instance.

**AAL Implementation** is a realization of an AAL including but not limited to the software libraries, drivers and the Hardware Accelerator

**Accelerated Function (AF)** is a representation of a workload building block that an accelerator processes on behalf of an AAL Application within an O-RAN Network Function in an O-Cloud [1].

**AAL Application (AAL-App)** is defined as a workload that can offload Accelerated Functions to AAL-LPU(s).

NOTE: Unless explicitly noted, the term Application refers to an AAL Application in AAL specifications.

NOTE: Unless explicitly noted, the terms Application, NF Application, L2 Application, VNF/CNF, or NF workload accessing the AAL in figures of this present document refers to an AAL Application.

**AAL Profile(s) (AAL-Profile)** specify one or more Accelerated Functions that an accelerator processes on behalf of an AAL Application within an O-RAN Network Function in an O-Cloud [1].

**AAL Operations** actions supported by the AAL interface.

**AAL Profile Instance (AAL-Profile-Instance)** is an executing instance of an AAL profile that can be used by an AAL Application via the AAL interface. The AAL-Profile-Instance executes within an AAL-LPU execution environment.

**AAL Logical Processing Unit (AAL-LPU)** is a logical representation of resources within an instance of a HW Accelerator (example: there can be multiple processing units or subsystems on a hardware accelerator, or resource partitioning (hard – dedicated resources, soft – soft resources) and these can be logically represented as a AAL Logical Processing Unit)

**AAL Profile Queue** is a logical grouping construct and may be used by the AAL Application to group operations together. For example, AAL Profile Queues may access specific resources (compute, I/O) of an AAL-LPU executing specific AAL Profile Instances(s).

**AAL Profile Queue ID** is a unique index used to designate the AAL Profile Queue in function exposed by specific AAL profile.

NOTE: An AAL Profile Queue or an AAL Profile Queue ID does not reflect a HW design or an AAL Implementation specification

**HW Accelerator Manager (HAM)** is an acceleration management function, that provides management capabilities for the HW Accelerator(s) and the AAL-LPU's in the O-Cloud Node. Management capabilities include but not limited to lifecycle management, configuration, updates/upgrades and failure handling of the HWA and the LPUs. The HAM may use the services of an external software repository to securely transfer approved software artifacts, for example FW/SW images, binaries, libraries and configuration data. The specification of a software repository and the communication with the HAM are outside the scope of O-RAN specifications. The HW Accelerator Manager is considered an O-Cloud Platform Software.

## Symbols

Void

## Abbreviations

For the purposes of the present document, the abbreviations given in [i.1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in [i.1] .

AF Accelerated Function

AAL Acceleration Abstraction Layer

AAL-App Acceleration Abstraction Layer - Application

AAL-LPU Acceleration Abstraction Layer - LPU

AALI-C Acceleration Abstraction Layer Interface-Common

AALI-C-Mgmt Acceleration Abstraction Layer Interface-Common-Management

AALI-C-App Acceleration Abstraction Layer Interface-Common-Application

AALI-P Acceleration Abstraction Layer Interface-Profile

BF Beam-forming

CNF Cloudified Network Function

DFT/iDFT Discrete Fourier Transform / Inverse Discrete Fourier Transform

DMS Deployment Management Services

FCAPS Fault, Configuration, Accounting, Performance, Security

FEC Forward Error Correction

FFT/iFFT Fast Fourier Transform / inverse Fast Fourier Transform

HWA Hardware Accelerator

IMS Infrastructure Management Services

LPU Logical Processing Unit

RB Resource Block

SMO Service Management and Orchestration

UC Use Case

VNF Virtual Network Function

# General Aspects

## Hardware Acceleration

In the design of digital computing systems, ranging from general-purpose processors to fully customized hardware, there is a tradeoff between flexibility and efficiency, with efficiency increasing by orders of magnitude when any given application is implemented in hardware. The range of implementation options includes general-purpose processors (GPPs) such as CPUs, more specialized processors such as Graphics Processing Units (GPUs), functions implemented on field-programmable gate arrays (FPGAs), and fixed-functions implemented on application-specific integrated circuits (ASICs). Hardware accelerator is a specialized HW implementation that can offload processing from application(s) running on the General-Purpose Processor (GPP). Any transformation of data or computation can be implemented purely in software running on a generic CPU, or purely in a specialized hardware accelerator, or using a combination of both. The implementation of computing tasks in hardware to improve performance is known as hardware acceleration. The hardware acceleration can be implemented in the form of lookaside or inline mode where in the former case, the host CPU invokes an accelerator for data processing and receives the result after processing is complete, while in the latter case, the accelerator, after being invoked by the host CPU with the request for data processing, completes the processing request of data received from a source node and directly transfers the post-processed data to a destination node, where the source or destination node can be different than host CPU (e.g., an Ethernet Interface). The principle of hardware acceleration and functional offloading in lookaside mode and inline is illustrated in Figure 4.1-1, allowing the application to offload workload to a hardware accelerator and to continue performing other work in parallel- this could be to continue to execute other software tasks in parallel or to sleep and wait for the accelerator hardware to complete. The hardware acceleration boosts application performance in environments with compute-intensive, deeply pipelined, massively parallel operations as shown in Figure 4.1-1. This model requires the support of two operations, one for initiating the offload and another for retrieving the operation once complete.



Figure 4.1-1: Example illustration of the effect of hardware acceleration on functional compute performance.

## AAL Architecture and concepts

### Overview

The goal of the acceleration abstraction layer (AAL) is to specify a common and consistent set of interfaces to enable interactions with the HW Accelerators and facilitates decoupling of an AAL Application, from a specific HW implementation. To accommodate the many different combinations of HW and SW implementation and many different network deployment scenarios, the AAL introduces the concept of an AAL profile which is used to distinguish between the different combinations of accelerated functions to be offloaded.

The AAL specifications shall define the AAL interface between the AAL Application and AAL Implementation in the O-Cloud instance. This includes the APIs, information models, operations and input/outputs used by the AAL Application to interface with the AAL Implementation. When the AAL APIs are utilized:

* AAL Applications that utilize the AAL APIs render the same functionality as described in the O-RAN specifications describing the AAL Application [1].
* AAL Applications that utilize the AAL APIs shall preserve the interface definitions and behave the same way as described in the O-RAN specifications describing those interfaces.
* O-RAN NFs that utilize the AAL APIs shall preserve the synchronization topologies as defined in [16].

In addition, the AAL Specification shall define the requirements for managing the hardware accelerator in the O-Cloud instance. The AAL Implementation itself shall not be defined by the AAL GAnP specification. ETSI GS NFV-IFA 002 [5] defines several abstraction models including pass through and abstracted models that can be used to realize an AAL Implementation.

The AAL specification facilitates the following:

An O-cloud provides the flexibility of deploying multiple software implementations from different software vendors on a common CPU-based (e.g., x86/ARM) platform with hardware accelerators (e.g., FPGA/DSP/ASIC/GPU) for specific functions, and conversely, also allows multiple physical deployment scenarios in terms of centralizing or distributing each network element with the same software implementation.

A disaggregated and cloudified multi-vendor RAN requires common vendor-neutral APIs for managed element discovery, lifecycle management, FM/PM, and orchestration across both PNFs and VNFs in order to function as a cohesive unit that supports key lifecycle use cases such as scale-out, slice management, fault tolerance, and hitless software upgrades.

### AAL Architecture

The end-to-end high-level AAL architecture block diagram is shown in Figure 4.2.5.1‑1. For AAL Applications O1 interface usage is optional. Interaction between DMS and AAL Application is out of the scope of this document. For further understanding of the deployment and operation of AAL-LPUs please refer to section 6.2.



Figure 4.2.2-1: High Level AAL Architecture Diagram

shows a pictorial view of the entity relationship and cardinality between AAL entities that constitute the AAL architecture. It is not meant to be the basis of a class diagram or object model which would normally be the start of an information model. Its purpose is just to help the reader mentally conceptualize the concepts depicted. See clause 4.2.4 of the present document for a high-level description of the AAL-C-Mgmt interface and the AAL-C-App interface. See [15] for a detailed description of the specification of the AAL-C-Mgmt and AAL-C-App interfaces and the operations supported.

AAL Managed Elements and Managed Functions are managed by SMO via the O2 interface exposed by the IMS and DMS using the AALI-C-MGMT interface terminated by the HW Accelerator Manager. Specifically, AAL Managed Elements and Managed Functions do not utilize the O1 interface for management of AAL Managed Elements and Managed Functions.

The HW Accelerator, the HW Accelerator Manager, AAL-LPU and AAL drivers are defined as O-Cloud Platform Software for the purposes of management and orchestration [1]. The O-Cloud Infrastructure and O-Cloud Platform Software use case flows in [12] are applicable for these AAL elements.

@startuml

allow\_mixing

hide circle

hide empty members

skinparam class {

FontSize 10

BorderColor black

BackgroundColor lightgrey

ArrowFontName Ariel

ArrowColor black

ArrowFontColor #777777

}

class "AAL Logical Processing Unit" as lpu

abstract class “AAL Profile” as profile

class “AAL Profile Queue” as queue

class “AAL Profile Instance” as profilei

class "AAL Application" as app

class "Hardware Accelerator" as hwa

class "IMS" as ims

class “HW Accelerator Manager” as ham

abstract class “Accelerated Function” as af

ims “1” -down- “0..n” ham: manages

hwa “1” \*-down- “0..n” lpu: resources rep by

lpu “1..n” -right- “0..n” profile: supports

profile “1..n” -up- “1..n” af: specifies >

lpu “1” -- “0..n” profilei: executes

profilei “1” \*-down- “0..n” queue: consists of

profile .down. profilei: is realized by >

app "0..n" -up- "1..n" lpu: uses >

app “1” -right- “0..n” profilei: uses

app “1” -right- “0..n” queue: uses

ham “1” -down- “0..n” hwa: manages

@enduml

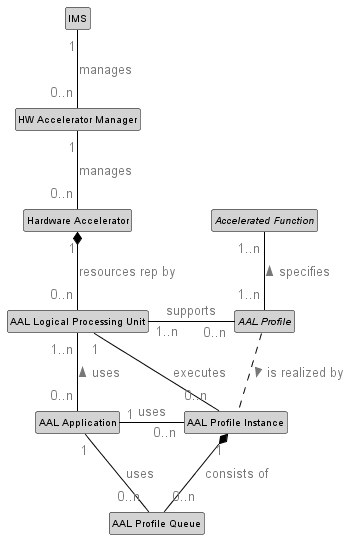


Figure 4.2.2-2: AAL Resource Relationship and Cardinality

### HW Accelerator Manager

The HW Accelerator Manager (HAM) is responsible for exposing a consistent mechanism to the O-Cloud platform for the discovery, lifecycle management, fault, state/status, performance, configuration, updates/upgrades, and error handling of the HW Accelerator(s) that are part of the O-Cloud Platform Hardware. The HW Accelerator Manager exposes the AALI-C-Mgmt interface towards the IMS. The interfaces between the HW Accelerator and the HAM are vendor specific and not in the scope of AAL specifications. The HW Accelerator Manager is managed by IMS as per Figure 4.2.2-1: High Level AAL Architecture Diagram of this document. The identifier of the HW Accelerator Manager used within the IMS is unique within an O-Cloud instance.

The HW Accelerator Manager is under the control of the O-Cloud Infrastructure. It is integrated into the Cloud Infrastructure using the O-Cloud Infrastructure's installation procedures. The HW Accelerator Manager shall be certified by the O-Cloud vendor.

The following list provides a description of capabilities offered by HAM. The list is not exhaustive:

* Discovery and Life Cycle Management:
  + The HW Accelerator Manager provides a mechanism to expose inventory information and capabilities of the physical and logical partitioning of the hardware and software.
  + The HW Accelerator Manager provides the ability to discover the capabilities of the HW Accelerator(s).
* Software/ Firmware upgrade services
  + The HW Accelerator Manager allows the update and/or upgrade of the software for a HW Accelerator(s) on the O-Cloud node. An example of this may include the programming or re-programming of a downloadable firmware or driver upgrades. Updates/Upgrades can be done locally or remotely.
* Configuration
  + The HW Accelerator Manager allows the configuration of the HW Accelerator as prescribed by the IMS through the AALI-C-Mgmt interface. The configuration of the HW Accelerator Manager may include HW Accelerator resource assignment to AAL-LPUs.
* Fault and Performance Monitoring:
  + The HW Accelerator Manager allows exposure of faults, logs and performance measurements toward the IMS.

Deployment Scenarios

The HAM’s software is deployed as part of the O-Cloud Platform software. The installation and update of the HAM’s software, deployed as a part of O-Cloud platform software, is described section 3.1.2 and section 3.1.6 of Cloudification and Orchestration Use Cases and Requirements for O-RAN Virtualized RAN [13].

The HAM may be located on O-Cloud Nodes designated as part of the clusters used to deploy the AAL Applications (NFs) or otherwise as shown below. The figure shows HAM from the perspective of AAL-C-Mgmt interface termination point. The actual software entities (i.e., vendor specific libraries used by the HAM) for managing the HWA can reside in other locations as well.



NOTE 1: The scenarios depicted are considered an exemplification of the most common deployment scenarios. Other locations and modelling approaches are not precluded.

Figure 4.2.3-1 Example HAM software deployment scenarios

### AAL Interfaces

The AAL interface API has two distinct parts, the first part corresponds to a set of common APIs (AALI-C) to address all the profile independent aspects of the underlying AAL Implementation(s) within an O-Cloud platform. The second part corresponds to a set of AAL profile specific APIs (AALI-P) which is specific to each defined AAL profile.

There are two categories of AALI-C interface:

* AALI-C-Mgmt: OAM management from the HAM toward the O-Cloud IMS for acceleration resources exposed by this interface.
* AALI-C-App: Common operations/actions/events towards the RAN AAL Application for resources exposed by this interface.

A candidate set of functionalities for AAL services supported by the AAL common API(s) potentially includes (but is not limited to) the following:

* Inventory Management, Fault, Performance, Configuration Management
* Software/ Firmware upgrade services
* Operations (Query status, Reset/Restart)
* Life Cycle Management of resources exposed by the interface. The AALI-C-Mgmt provides the Life Cycle Management of the HW Accelerator and the AAL-LPU resources.
* Configuration of the state of these AAL-LPU(s) (for example, start, stop, or reset of an AAL-LPU).
* Configuration of various counters and resources associated with AAL-LPU(s) (for example, performance measurements/indicators, performance monitoring metrics, events, faults etc.).
* Discovery of AAL-profile(s) supported by these AAL-LPU(s) and associated configurations etc.
* Abstraction of transport mechanism between the AAL Application and AAL Implementation

The information model and the exact list of operations and actions applicable across AAL-C-Mgmt and AAL-C-App are defined in [15].

The second part of AAL interface corresponds to a set of AAL profile specific APIs (AALI-P) which is specific to each defined AAL profile. The AAL profile can be common across the different AAL Implementation accelerating the same set of AFs. It enables the AAL Application to efficiently offload an AAL profile workload to the AAL Implementation in a consistent way without requiring the HW implementations to expose every single detail of the underlying HW implementation to the AAL Applications. Figure 4.2.4-1 shows examples of the AAL APIs presented to an AAL Application in three different scenarios.

The AALI-C-App & AALI-P APIs are internal to the AAL Application that utilize them.



Figure 4.2.4-1: AAL Application Common and profile APIs

### AAL Deployment in O-Cloud environments



Figure 4.2.5-1: Accelerator APIs/Libraries in Container (left) and Virtual Machine Implementations (right)

The AAL specifications define the AAL interfaces (i.e AAL-C and AAL-P) and the AAL profiles that may be supported. The AAL-C-App interface is used by AAL Application to access the AAL Implementation encompassing HW Accelerator and associated SW libraries, drivers etc. In Figure 4.2.5-1 two deployment scenarios are shown, one with Containers and the other with Virtual Machines.

Figure 4.2.5-1 also shows the O-Cloud Infrastructure Management Services and Accelerator management entity (i.e. the HAM). The orchestration of the HAM as a managed entity is outside the scope of the AAL and shall be specified in the O-RAN WG6 O2 specification [11].

#### AAL Implementation Software Deployment Options

The following figures describe AAL Implementation software deployment options. In Figure 4.2.5.1-1: there is an example where the entire AAL Implementation software is deployed as part of the O-Cloud Platform Software [3], and there is no software component dependency between NF Deployments and O-Cloud Platform Software. While in Figure 4.2.5.1-2: and Figure 4.2.5.1-3: there are other examples where part of the AAL Implementation software is deployed as part of the NF Deployment and part of O-Cloud Platform Software. In these and similar cases there is a software component dependency between the NF Deployment and the O-Cloud Platform Software, as such updating AAL Implementation software in the O-Cloud Platform Software may require an update to the NF Deployment, this is outside the scope of the AAL.

The installation and update of the AAL Implementation software including but not limited to device drivers and libraries, deployed as a part of O-Cloud Platform Software, is described in Cloudification and Orchestration Use Cases and Requirements for O-RAN Virtualized RAN [12], clause 3.1.2 and 3.1.6.

In the diagrams below, ‘Container image(s)’ and ‘VM image(s)’ refer specifically to containers or VMs which are being used to run AAL Applications and do not preclude the existence of other containers or images. These diagrams depict several different possible ways in which the AAL Implementation software may be deployed. The AAL Implementation software may exist wholly or partially inside or outside of the specific containers or VMs which are being used to enclose the AAL Applications.



Figure 4.2.5.1-1: Example AAL Implementation Software deployment contained within O-Cloud Platform Software only



Figure 4.2.5.1-2: Example AAL Implementation Software Deployment split between O-Cloud Platform Software and NF Deployment



Figure 4.2.5.1-3: Example AAL Implementation Software Deployment split between O-Cloud Platform Software and NF Deployment in a virtual machine environment

## Relationship with Standards

The O-RAN AAL interface shall leverage existing standards wherever possible.

### Relationship with ETSI

In [4,5,10], ETSI has specified a generic acceleration and abstraction model as well as acceleration resource management that have served as the basis of this specification.

ETSI GS NFV-IFA 001 [10] provides a classification of acceleration types and describes several related uses cases, for example about compute acceleration when considering Virtual Base Stations (VBS). ETSI GS NFV-IFA 002 [4] specifies an acceleration architectural model for NFV. The acceleration model considers that a single HW device exposes multiple abstract virtual accelerators and multiple instances of these virtual accelerators per VNF. ETSI GS NFV-IFA 002 [4] defines the interfaces between the NFVI and the VNF instances to support this concept. ETSI GS NFV-IFA 019 [5] defines acceleration resource management interfaces between the VIM and the NFVI (e.g., Acceleration Image Management interface). From an NFV orchestration and management perspective, acceleration capabilities are defined as part of the relevant Virtual Compute Resource information elements, i.e., a virtual compute resource (e.g., a VM) can have associated acceleration capabilities from the set of capabilities offered by the acceleration resources. For the relevant descriptors, ETSI GS NFV-IFA 011 [14] details the VirtualComputeDesc information element in VNFD which describes CPU, Memory and acceleration requirements of the virtualisation container (e.g., a VM or a set of OS containers, such as a Pod) realizing a VNFC (i.e., a component of a VNF). For containerized deployments required acceleration capabilities can be described by the extendedResourcesRequests attribute in the OsContainerDesc information element. In addition, in the case of a containerized deployment, requirements about constraints on the placement of the set of one or more OS containers (a Pod) to cluster nodes with certain capabilities, such as acceleration, can also be defined. In both cases (VM-based and OS container-based), the VDU information element also supports an attribute to request for additional capabilities, being acceleration related capabilities as one possible example.

In ETSI NFV the concept of the abstract virtual accelerator has been devised based on the idea of an extensible para-virtualised device. However, the concept has only been defined considering its use in hypervisor-based virtualization systems and is not ad-hoc applicable to the case of containerized deployments without further analysis. Also, there is no relevant ETSI NFV stage 3 work which can be referenced for the interfaces described in ETSI GS NFV-IFA 002 [4] and ETSI GS NFV-IFA 019 [5].This specification builds upon the concepts described by the aforementioned ETSI NFV specifications but also goes beyond to consider the advances introduced by cloud native solutions, for example by introducing the concept of LPUs. .

ETSI GR NFV-IFA 046 [15] profiles the NFV-MANO acceleration abstraction framework and compares it to the O-RAN acceleration abstraction solution developed in the context of O-RAN, from which several challenges are further described. A set of recommendations about future work in the NFV framework are derived in the same referenced group report, some considering an alignment between the two designs to make ETSI NFV specifications more expressive to support additional functionality.

# AAL Interface definition General Principles and Requirements

## General Principles

The set of generic and profile-specific features of the AAL interface between the AAL application and the AAL implementation described in the following subsections are defined from an AAL Application point of view and are related to the AAL-C-App general principles and requirements.

### Extensibility

O-RAN has defined the functions that can be accelerated by the cloud platform based on 3GPP specifications and O-RAN deployment scenarios. However, the AAL should not limit innovation of future implementations and should evolve as the specification requires. To that end, the AAL Application interface shall be extensible to accommodate future revisions of the specification.

### HW Independence

AAL Application interface shall be independent of the underlying AAL Implementation.

### Interrupt and Poll Mode

The AAL Application interface shall support multiple design choices for AAL Application vendors and shall not preclude an AAL Application/HW Accelerator vendor from adopting/supporting an interrupt-driven design or poll-mode design or any combination of both. As such, the AAL Application interface shall support both interrupt mode, poll mode and any combination of interrupt and poll modes for the data-path AAL Application interface.

### Discovery and Configuration

The AAL Application interface shall enable AAL Application to discover and configure AAL-LPU(s). The AAL Application interface shall allow an AAL Application to discover what physical resources have been assigned to it from the upper layers and then to configure said resources for offload operations.

### Multiple AAL-LPU Support

There may be scenarios where multiple AAL-LPUs (either implementing the same or different AAL profile(s) are assigned to a single AAL Application, which uses one or more of these AAL-LPU(s) as needed. The AAL Application interface shall support an AAL Application using one or more AAL-LPU(s) at the same time, as shown in Figure 5.1.5-1.



Figure 5.1.5-1: Logical Representation of AAL Application interface support for multiple AAL-LPUs

### AAL offload capabilities

The AAL Application interface in supporting different AAL profiles and AAL Implementations shall support different offload architectures including look-aside, inline, and any combination of both. An AAL Implementation shall support one or more of these offload architectures depending on the supported AAL profile(s).

### Look-aside Acceleration Model

The AAL Application interface shall support look-aside acceleration model where the AAL Application invokes a HW Accelerator for data processing and receives the result after processing is complete. A look-aside architecture, illustrated in Figure 5.1.7-1, allows the AAL Application to offload AF(s) specified by AAL profiles(s) to a HW Accelerator and continue to perform other work in parallel—this could be to continue to execute other software tasks in parallel or to sleep and wait for the HW Accelerator to complete. This model requires the AAL Application interface to support two operations, one for initiating the offload and another for retrieving the output data once complete.



Figure 5.1.7-1: AAL Application interface look-aside acceleration model - Data flow

### Inline Acceleration Model

The AAL Application interface shall support inline acceleration model where the AAL Application, after invoking a HW Accelerator for offloading AF(s) specified by AAL profile(s), does not necessarily retrieve the post processed data. Unlike the look-aside acceleration model where data source/sink is always the AAL Application (i.e., the HW Accelerator always receives the data to be processed from the AAL Application and returns the post processed data to the same), a HW Accelerator operating in inline acceleration mode receives/returns data from/to a different source/destination endpoint than the AAL Application, depending on the direction of data flow (e.g., in downlink (DL) direction versus uplink (UL) direction). Figure 5.1.8-1 shows one possible implementation of an inline acceleration model.



Figure 5.1.8-1: AAL Application interface inline acceleration model - Data flow

In Figure 5.1.8-1, “Tx” refers to the transmission of the data from the HW Accelerator through an egress port (e.g., an Ethernet interface) to a destination node (e.g., O-RU), while “Rx” refers to the reception of data through an ingress port (e.g., Ethernet interface) to the HW Accelerator from a source node (e.g O-RU).

While the look-aside architecture (in DL) shall support dataflow from the CPU to the HW Accelerator and back to the CPU before being sent to the egress port (front-haul interface), the inline architecture (in DL) shall support data flow from the CPU to the HW Accelerator and directly from the HW Accelerator to the egress port (front-haul interface), instead of being sent back to the CPU. The typical user plane data flows for accelerating the O-DU High-PHY functions for the look-aside and inline architectures are as follows.

Look-aside architecture user plane dataflow

CPU ↔ HW Accelerator ↔ CPU ↔ front-haul: for a set of consecutive PHY functions offload (e.g., FEC)

CPU ↔ HW Accelerator ↔ CPU ↔ HW Accelerator ↔…↔ CPU ↔ front-haul: for a set of non-consecutive PHY functions offload

Inline architecture user plane dataflow

CPU ↔ HW Accelerator ↔ front-haul: for a set of consecutive PHY functions offload (up to the end of the PHY pipeline)

Figure 5.1.8-2: illustrates one possible implementation of the look-aside and inline architectures. While a set of PHY-layer functions are offloaded to the HW Accelerator for look-aside acceleration, the entire end-to-end High-PHY pipeline is offloaded to the accelerator for inline acceleration.



Figure 5.1.8-2: User plane dataflow paths in look-aside and inline acceleration architectures.

### AAL Application interface Concurrency and Parallelism

To enable greater flexibility and design choice by AAL Application vendors, the AAL Application interface shall support multi-threading environment allowing an AAL Application to offload acceleration requests in parallel from several threads.

### Separation of Control and User Plane AAL Application interface APIs

For efficiency and flexibility of AAL Implementation, AAL Application interface shall support separation of control and user plane APIs with appropriate identifiers, as required by different AAL profiles.

### Support of Versatile Acceleration Payload

Range of payload sizes can vary widely, depending on the specific layer of the RAN protocol stack from which the workload for AAL profile(s) is offloaded to a HW Accelerator. AAL Application interface API shall be flexible to support various ranges of payload sizes as required by different AAL profiles.

### Support of Different Transport Mechanisms

The transport between an AAL Application and an AAL Implementation can be of different types (e.g., based on shared memory, PCIe interconnect, over ethernet). The AAL Application interface shall support abstraction of these various transport mechanisms between the AAL Application and the AAL Implementation.

### AAL API namespace

For convenience of AAL Implementation, the AAL shall follow a unique name space for all AAL API functions.

### Chaining of AAL Profiles

AAL profiles specify one or multiple AFs in specific order offloaded to the HWA. To take advantage of multiple AFs offloaded to the same or different HWAs, the AAL Implementation can permit the chaining of AAL Profile(s) executing the AF or set thereof. In such a case, the AAL Implementation redirects the output of preceding AAL Profile as input to the subsequent AAL Profile belonging to the same or different LPU(s) on the same or different HWA(s). Such a chaining of AAL Profiles allows data to be transferred from one AAL Profile to another AAL Profile without the intervention of the AAL Application thereby reducing transfer latencies. Chained AAL Profile Instances can reside on the same or different LPUs and HWAs. However, in this version of specification, only considers the chaining of AAL Profiles on single AAL-LPU, and the HAM announces chained AAL Profiles which can be satisfied on the single AAL- LPU and HWA. Other cases where the chained AAL Profile straddles across multiple AAL-LPUs and HWAs are for further study.

The use of chaining two AAL Profiles is optional for the AAL Implementation and depends on the AAL Profiles requested by the AAL Application. During CNF deployment as per K8s profile, or VNF deployment as per ETSI profile, AAL Implementation can chain the AAL Profiles to create a chained AAL Profile with more AFs implemented as part of the chained AAL Profile. The chained AAL Profile should conform to the AAL Profile requested by the AAL Application, i.e., the chained AAL Profile has same properties towards the AAL Application as well as the IMS and the DMS. It is the responsibility of the AAL Implementation to announce that the chained profile as an AAL Profile as defined in AAL specifications to which the chain conforms. The first AAL Profile in the chain takes data from the AAL Application and the last AAL Profile returns the transformed data to the AAL Application.

A chained AAL Profile Instance is valid only if the two or more AAL Profiles form semantically correct sequence of consecutive blocks in the AF processing chain and no intervention is required by the AAL Application for control or data exchanged between the two AAL Profiles. Additionally, the output format of the preceding AAL Profile should match with the input format of the succeeding AAL Profile.

AAL Profile chaining may be applicable to both inline and look-aside acceleration. The data flows in these cases are depicted in Figure 5.1.14-3: and Figure 5.1.14-4:.



Figure 5.1.14-1: Data flow through unchained AAL Profiles



NOTE 1: In Figure 5.1.14-1 and Figure 5.1.14-2 above, the chained AAL Profile Instance is a data stream transformer that works upon the input data stream to produce output stream. Input data and parameters are passed on from preceding AAL Profile to next. These are provided by the AAL Application in case of the first AAL Profile. The difference is that in Figure 5.1.14-1:, the AAL Application would call a series of AAL Profile Instances, while in **Error! Not a valid bookmark self-reference.**, the AAL Application calls only one AAL Profile Instance, which happens to be the chained AAL Profile instance.

NOTE 2: The mechanisms for transfer of data and parameters to each of the AAL Profile Instances in unchained mode and to the chained AAL Profile Instances follow the AAL-C-App interface’s transport as defined in AAL Common API Specifications [14]. The AAL Profiles in the chained mode shall continue to follow the AAL-C-App Interface specified in [14]

NOTE 3: The AAL-Implementation can configure the chaining of AAL-Profiles during its initialization in order to avoid disruption in the processing chain while the AAL Application is in service.

Figure 5.1.14-2: Data flow through chained AAL Profiles

The chaining of AAL Profiles is possible across the HWAs and LPUs, however, it is beyond the current scope of document and for future studies.

The chained look-aside acceleration architecture follows the same dataflow models as look-aside acceleration functions with subtle difference. The first and the last blocks in the chain interfacing the CPU to HW Accelerator support respectively dataflow from the CPU to the accelerator and then from accelerator back to the CPU. All the intermediate blocks support dataflow to and from the preceding and succeeding accelerated functions when those are configured as links in the chain. When not configured as chained accelerated functions, these should support dataflow as in look-aside acceleration.

Look-aside chained architecture user plane dataflow.

CPU ↔ HW Accelerator (AF1) ↔ HW Accelerator (AF2) ↔ … ↔ HW Accelerator (AFn) ↔ CPU ↔ front-haul: for offload of a set of consecutive PHY functions (e.g., FEC)

CPU ↔ HW Accelerator (AF1) ↔… ↔HW Accelerator (AFn) ↔ CPU ↔ HW Accelerator ↔…↔ CPU ↔ front-haul: for offload of a set of non-consecutive PHY functions

Figure 5.1.14-3: illustrates the data flow through the chained HW Accelerated functions within an accelerator when configured for consecutive PHY function offload while Figure 5.1.14-4 represents it for non-consecutive PHY function offload.



Figure 5.1.14-3: Dataflow through chained lookaside HW Accelerator for consecutive Hi-PHY functions.



Figure 5.1.14-4: Dataflow through chained lookaside HW Accelerator for consecutive and non-consecutive PHY.

## High-PHY Profile Specific Principles

The set of features of AAL described in the following subsections are relevant for inline High-PHY AAL profiles (profile names with suffix ‘\_HIGH-PHY’) defined in Chapter 5

### Separation of Cell and Slot Level Parameter Configurations

In general, “cell-specific” (typically static or semi-static in nature) parameters change less frequently than “slot-specific” (typically dynamic in nature, specific to PHY channels/signals) parameters associated with inline, High-PHY AAL profiles. Hence, for optimizing signalling overhead, the AAL Application interface shall support configuration of “cell-specific” and “slot-specific” parameters to the AAL Implementation using separate AAL Application interface API functions. It is noteworthy that the cell/slot specific configurations can include both control and user planes.

### SFN/slot-based Synchronization

The AAL Application interface shall support system frame number (SFN) based or slot-based synchronization between the AAL Application and the AAL Implementation supporting inline, high-PHY AAL profiles.

### Compatibility with O-RAN FH interface

The functional split between the O-DU and the O-RU remains as defined by the WG4 OFH interface [17] and the AAL Application interface API shall be compatible with O-RAN FH interface (7.2-x split) to enable communication between the O-DU AAL Application and O-RU via AAL Implementation as required by inline, High-PHY AAL profile(s).

### Inline Profile for High-PHY Stack

Inline data flow for the AAL Profile that specifies a complete stack of High-PHY functions implies that the set of Accelerated Functions is constituted by the entire U-plane processing of the High-PHY channels or signals (with 7-2x PHY functional split) and the IQ data (in DL) or decoded bits (in UL) (post processing) and these are transferred directly from the HW Accelerator to the Fronthaul interface (in DL) or to the Layer 2 (in UL).

Inline acceleration for AAL Profiles that specify a partial stack of the High-PHY for UL and for DL Accelerated Functions is also possible; in such case only parts of the U-Plane, i.e., not all the functions of the High-PHY stack of a given channel, may be offloaded to a HW Accelerator. For inline acceleration of a partial High-PHY stack, the IQ data or decoded bits are also transferred directly from the HW Accelerator to the Fronthaul (in DL) and to the non-accelerated part of the High-PHY stack (in UL). The set of Accelerated Functions that are offloaded in Inline mode for acceleration of a partial High-PHY stack would be AAL Profile specific. The support of AAL Profiles that specify the support of partial High-PHY stack is for future study.



Figure 5.2.4-1: Partial Inline Model for AAL Hi-PHY Profile

# AAL-LPU Principles

## Overview

This section discusses about AAL-LPU(s) presented to AAL Applications using the AAL Application interface. An AAL-LPU should not be confused with a physical HW Accelerator. Within a process address space each AAL-LPU shall abstract the AAL Application from underlying HW Accelerator.

The following list provides a description of capabilities and characteristics of AAL-LPUs:

* An AAL-LPU maps to a single HW Accelerator. An AAL-LPU can be identified uniquely within a HW Accelerator.
* A HW Accelerator may support 1 to N AAL-LPU’s.
* Each AAL-LPU shares the resources of the associated HW Accelerator with other AAL-LPU(s) mapped to the same HW Accelerator. AAL-LPU can also represent a hard partition of the HW Accelerator where resources are dedicated to the partition.
* Mapping of HW Accelerator resources to AAL-LPU shall be configurable from O2 interface
* An AAL-LPU may support more than one AAL profile. For each supported AAL profile, an AAL-LPU may execute 0 to N AAL-Profile-Instances.
* An AAL-LPU can be assigned to a single POD or VM. Multiple LPU's can be assigned to a POD or VM.
* An AAL-LPU can provide service to 0 or more AAL Applications within a POD or VM.
* AAL-LPU is a virtual Managed Element as defined in [13].

Depending on HW design and implementation choice, a HW Accelerator may accelerate multiple profiles or offer support for sharing HW Accelerator resources between multiple threads, processes, VMs, PODs. For this reason, a second abstract construct known as AAL Profile Queue can optionally be used to

* distinguish between multiple supported AAL profiles per AAL-LPU
* prioritize access to AAL-LPU resources
* group operation requests
* allow parallel access through AAL Application interface for multiple threads

As an abstract construct, an AAL Profile Queue does not reflect a HW design specification or requirement.

From the AAL Application point of view, an AAL Profile queue is exposed by an AAL Profile Instance

* an AAL Profiles Instances exposes one or more AAL Profile Queues
* The AAL Profile Queue optionally also supports priority, allowing the AAL Application to schedule jobs of different priorities.

NOTE: An AAL Profile Queue can be used by an AAL Application to share AAL-LPU resources between threads/cores belonging to the same process address space

NOTE: An AAL Application may use multiple AAL Profile Queues to access different AAL Profile Instances supported by an AAL-LPU

## LPU Deployment and Operation

### Example AAL-LPU Mapping

The following Section contains example deployments mapping AAL-LPUs to AAL Applications. The labels ‘profile-instanceID’ and ‘queueID’ in the following diagrams denote AAL Profile Instance object handle and identifier of an AAL Profile Queue respectively.

* **Scenario 1: Basic implementation:** A HW Accelerator supports a single AAL-LPU which exposes a single AAL-Profile-Instance for one AAL Application to use



Figure 6.2.1-1: Scenario 1: A single AAL LPU exposes a single AAL Profile Queue used by a single AAL Application.

**Scenario 2: Basic Multi Application Support:** A HW Accelerator supports multiple AAL LPUs for multiple AAL Applications



Figure 6.2.1-2: A single HW Accelerator supporting two LPU's each assigned to individual AAL Applications

* **Scenario 3: Multiple Accelerator Support:** Mapping example showing multiple HW Accelerators assigned to a single AAL Application



Figure 6.2.1-3: Two HW Accelerators each supporting a single AAL-LPU assigned to a single AAL Application.

* **Scenario 4: Multi Queue Support:** AAL LPU mapping showing multiple AAL Profile Queue support



Figure 6.2.1-4: A single AAL LPU exposes two AAL Profile Queues used by a single AAL Application.

* **Scenario 5: Multi Profile Support:** Mapping example showing multi-function support



Figure 6.2.1-5: A single AAL LPU supporting two AAL Profile Instances exposes two AAL Profile Queues used by a single AAL Application.

* **Scenario 6: Chained Profile Support:** Mapping example showing a multi-profile chained AAL Profile Instance on an AAL-LPU



Figure 6.2.1-6: AAL-LPU Mapping example showing chained profile support

### Statistics

The AAL Application interface shall provide an AAL Application with general statistics upon request. Statistics may include but not limited to operation counts and error counts.

### Memory Management

O-RAN network functions (O-DU, O-CU, etc.) will be responsible for input, output and operation structure memory allocation and freeing, using AAL defined memory management functions. All other AAL Application memory is not required to use the AAL memory management functions.

Device Drivers are free to manage their own internal memory, DMA implementation as needed, the AAL specification does not add any memory requirements to device driver.

Each AAL Implementation shall define its own memory requirements and implement its own memory backing if needed.

Each AAL Implementation may define its own operation memory structure and allocation if needed.

### Run Time Configurations

Operations are requested to the AAL-LPU to execute specific HW Accelerated Function(s). Each operation shall be represented by an operation struct that shall define all necessary metadata, configurations and information required for the operation to be processed on an AAL-LPU. The operation structs shall define the operation type to be performed, including an operation status and reference to the AAL profile specific operation data which can vary in size and content depending on the AAL profile. Each AAL profile shall define its own operation structure for its specific functions.

### AAL Profile(s) offload, processing status query and processed data retrieval

An AAL Application aggregates one or more AAL profile(s) and offload to the AAL Implementation using a single AAL Application interface API invocation. As one example, for High-PHY AAL profiles defined in clause7, multiple AAL profiles (where an AAL profile refers to a PHY channel/signal for one or more than one cell(s) and one or more than one UE(s)) scheduled within a slot can be aggregated and offloaded to an AAL-LPU by the AAL Application using a single AAL Application interface API invocation.

The processing status of offloaded/enqueued AAL profile(s) can be queried by the AAL Application in an ‘asynchronous’ manner, i.e., not necessarily in the same order in which the AAL profile(s) are offloaded. In case the AAL Application retrieves the post-processed data from the AAL Implementation, a ‘processing status query’ request can be bundled with a ‘processed data retrieval/dequeue’ request. In general, status query and dequeue request corresponding to multiple enqueue requests can be bundled together by the AAL Application and invoked through a single AAL Application interface API function.

### AAL-LPU Exposure

#### Overview

After using the HW Accelerator Manager to create the AAL-LPU(s) and the profiles supported, we can then expose it to the AAL Application (e.g. in KubernetesⓇ it will be part of the POD environment variables). The goal of this section is to abstract the way the AAL-LPU and its supported profiles are exposed to the AAL Application in order to achieve AAL Application portability. Hence, the AAL-LPU and its profile(s) shall be exposed the AAL Application in an abstracted descriptor.

As of today many implementations refer to the AAL-LPU by its PCI address and describe a single profile for the entire HW Accelerator, meaning exposure of one single profile for all LPUs supported. This section provides step-by-step example that shows how the AAL-LPU could be exposed to abstract the PCIe address and the profile(s) supported by the AAL-LPU. Although the below example is a KubernetesⓇ example, the outcome of it is independent of orchestration technology supported.

#### Example

##### AAL-LPU Configuration

1. A given HW Accelerator can support 16 AAL-LPUs and FEC LDPC, PHY profiles but the default setup of the AAL-LPUs is FEC:



Figure 6.2.6.2.1-1: Example AAL-LPU and profile supported

1. Now assume HW Accelerator Manager configured the HW Accelerator as follow:

* **LPU-1** supports FEC version 01, PHY and LDPC version 01 profiles
* **LPU-2** supports LDPC version 02 profile
* **LPU-3** supports LDPC version 02 profile
* **LPU-4** supports LDPC version 02 profile



Figure 6.2.6.2.1-2: Example HW Accelerator configuration

1. Assigning it to a the a vDU POD and exposing the two LPU/Profiles needed by vDU (in green):



Figure 6.2.6.2.1-3: Example assignment of AAL-LPUs and supported profiles to POD

We can see that the AAL-LPU resources requested in the manifest is translated in the POD environment to three AAL-LPU addresses plus the strings that describe the profiles supported by the LPU. Please note that the AAL Application will be able to query the AAL-LPUs and their supported profiles via the AAL-C-App API.

##### AAL-LPU Control by the AAL Application

The figure below shows how the AAL Application for a vDU can create several PHY profiles on LPU-1, for example, PHY Profile-1 handles cell-1 and PHY Profile-2 handles cell-2, as well as running different LDPC profiles on different LPUs and all in the same HW Accelerator. As mentioned before the AAL Application can query what profiles supported by reading the POD environment variables or by querying it using the AAL-C-App API.



Figure 6.2.6.2.2-1: Example AAL Application configured AAL-Profile-Instances

##### AAL-LPU resource tracking

It is a key to be able to track the AAL-LPU as a resource for providing the AAL Application with acceleration resources it needs otherwise the attempt to create the NF will fail due to insufficient resources available.

There are few notes related to the tracking of the AAL-LPU resources in relation to the example’s above in :

* If another POD needs to use the LDPC profile, Kubernetes will allow it as 2 out of 3 were used (LPU-4 supports it).
* If another POD needs to use FEC and PHY profiles (FEC&PHY), Kubernetes will reject it as only one LPU is configured this way.
* We need to consider if a logic is needed to track the AAL-LPU availability from the SMO. For example, a total of ten AAL-LPU/FEC profiles available in a cluster, three is being used by a deployment, seven left for a new deployment. The seven available could be revealed to the user or automation at the SMO level when considering additional deployments.

The conclusion is that the SMO shall be able to track the AAL-LPU as a resource and its availability.

# AAL Profiles

An AAL profile specifies a set of Accelerated Functions that a Hardware Accelerator processes on behalf of an AAL Application within an O-RAN Cloudified Network Function (e.g. O-DU, O-CU etc.). Accordingly, AAL profiles can be categorized as O-DU AAL profiles, O-CU AAL profiles and so on. The following sections describe these different AAL profile categories in further details.

## O-DU AAL Profiles

An O-DU AAL profile can specify a set of Accelerated Functions within the O-DU protocol stack. These functions may belong to a single layer (e.g., PHY) or span across multiple layers (e.g., PHY and MAC) within O-DU. The current O-DU AAL profiles being studied by O-RAN WG6 are focusing on Accelerated Functions from PHY layer of O-DU.

### O-DU Protocol Stack Reference

Figure 7.1.1-1: illustrates the building blocks for processing various O-DU PHY layer Downlink (DL) channels and signals (with 7.2-x functional split between O-DU and O-RU) defined by 3GPP in [6] & [7] as part of 5G NR specification.



Figure 7.1.1-1: O-DU PHY processing blocks for 5G NR Downlink

The O-DU PHY layer in downlink consists of the following physical channels and reference signals:

• Physical Downlink Shared Channel (PDSCH) and associated Demodulation Reference Signal (PDSCH DM-RS).

• Physical Downlink Control Channel (PDCCH) and associated Demodulation Reference Signal (PDCCH DM-RS).

• Synchronization Signal Block (SSB) consisting of

o Physical Broadcast Channel (PBCH) and associated DMRS (PBCH DM-RS).

o Primary Synchronization Signal (PSS).

o Secondary Synchronization Signal (SSS).

• Channel State Information-Reference Signal (CSI-RS) and Tracking Reference Signal (TRS).

• Phase Tracking Reference Signal (PT-RS) for DL.

The downlink physical channels (PDSCH, PDCCH, PBCH) carry information originating from higher layers (i.e. layer 2 and above).

The downlink physical layer processing of data channel (PDSCH) carrying transport blocks consists of the following steps:

TB CRC attachment: Error detection is provided on each transport block (TB) through a Cyclic Redundancy Check (CRC). Refer to Subclause 7.2.1 in [7] for details.

CB segmentation and CRC attachment: The transport block is segmented when it exceeds the code block (CB) size specified by 3GPP [7]. Code block segmentation and code block CRC attachment are performed according to Subclauses 7.2.3 and 5.2.2 of [7].

LDPC encoding: Refer to Subclauses 7.2.4 and 5.3.2 in [7] for details.

Rate matching: Refer to Subclauses 7.2.5 and 5.4.2 in [7] for details.

CB concatenation: Refer to Subclauses 7.2.6 and 5.5 in [7] for details.

Scrambling: Refer to Subclause 7.3.1.1 in [6] for details.

Modulation: Refer to Subclause 7.3.1.2 in [6] for details.

Layer mapping: Refer to Subclause 7.3.1.3 in [6] for details.

RE mapping: Refer to Subclause 7.3.1.5 and 7.3.1.6 in [6] for details on Resource Element (RE) mapping.

The downlink physical layer processing of control channel (PDCCH) carrying Downlink Control Information (DCI) consists of the following steps:

CRC attachment: Error detection is provided on DCI transmissions through a Cyclic Redundancy Check (CRC). Refer to Subclause 7.3.2 in [7] for details.

Polar encoding: Refer to Subclauses 7.3.3 and 5.3.1 in [7] for details.

Rate matching: Refer to Subclauses 7.3.4 and 5.4.1 in [7] for details.

Scrambling: Refer to Subclause 7.3.2.3 in [6] for details.

Modulation: Refer to Subclause 7.3.2.4 in [6] for details.

RE mapping: Refer to Subclause 7.3.2.5 in [6] for details.

The downlink physical layer processing of broadcast channel (PBCH) carrying maximum one transport block consists of the following steps:

PBCH payload generation: Refer to Subclause 7.1.1 in [7] for details.

Scrambling: Refer to Subclause 7.1.2 in [7] for details.

TB CRC attachment: Refer to Subclause 7.1.3 in [7] for details.

Polar encoding: Refer to Subclauses 7.1.4 and 5.3.1 in [7] for details.

Rate matching: Refer to Subclauses 7.1.5 and 5.4.1 in [7] for details.

Data scrambling: Refer to Subclause 7.3.3.1 in [6] for details.

Modulation: Refer to Subclause 7.3.3.2 in [6] for details.

RE mapping: Refer to Subclause 7.3.3.3 in [6] for details.

The downlink physical signals (DM-RS, PSS, SSS, CSI-RS/TRS, PT-RS) correspond to a set of resource elements used by the physical layer but does not carry information originated from higher layers (i.e. layer 2 and above).

Reference Signals (DM-RS, CSI-RS/TRS, PT-RS) and Synchronization Signals (PSS/SSS) are generated using the following steps:

Sequence Generation and Modulation: Refer to Subclauses 7.4.1.1.1 (PDSCH DM-RS), 7.4.1.3.1 (PDCCH DM-RS), 7.4.1.4.1 (PBCH DM-RS), 7.4.1.5.2 (CSI-RS/TRS), 7.4.1.2.1 (PT-RS), 7.4.2.2.1 (PSS) and 7.4.2.3.1 (SSS) in [6] for details.

RE mapping: Refer to Subclauses 7.4.1.1.2 (PDSCH DM-RS), 7.4.1.3.2 (PDCCH DM-RS), 7.4.1.4.2 (PBCH DM-RS), 7.4.1.5.3 (CSI-RS/TRS), 7.4.1.2.2 (PT-RS), 7.4.2.2.2 (PSS) and 7.4.2.3.2 (SSS) in [6] for details.

An O-DU AAL profile for 5G NR downlink shall specify a set of accelerated functions corresponding to one or more than one physical downlink channel(s) and/or physical downlink signal(s).

In addition to the processing blocks mentioned above, each of these downlink physical channels/signals may include some additional functional blocks (e.g. precoding, IQ compression) which are implementation specific and may also depend on system configurations/capabilities (for example, whether a O-DU is connected to a CAT-A/CAT-B O-RU). Each of these physical channels/signals can be implemented with/without these optional functional blocks. The AAL Application interface shall expose to the AAL Application whether these functional blocks are supported or not within the AAL Implementation.

Figure 7.1.1-2: illustrates the building blocks for processing various O-DU PHY layer Uplink (UL) channels and signals (with 7.2-x functional split between O-DU and O-RU) defined by 3GPP [6] as part of 5G NR specification.



Figure 7.1.1-2: O-DU PHY processing blocks for 5G NR Uplink

The O-DU PHY layer in uplink consists of the following physical channels and reference signals:

• Physical Uplink Shared Channel (PUSCH).

• Physical Uplink Control Channels (PUCCH) with formats 0/1/2/3/4.

• Physical Random-Access Channel (PRACH).

• Sounding Reference Signal (SRS).

• Phase Tracking Reference Signal (PT-RS) for UL.

The uplink physical channels (PUSCH, PUCCH, PRACH) carry information originating from higher layers (i.e. layer 2 and above).

The uplink physical layer processing of shared channel (PUSCH) carrying uplink data with or without Uplink Control Information (UCI) consists of the following steps at the receiver (O-DU):

RE de-mapping: Refer to Subclauses 6.3.1.6, 6.3.1.7 and 6.4.1.1.3 of [6] for details on RE mapping at the transmitter.

Channel estimation and equalization: up to O-DU implementation.

Transform precoding (IDFT): optional, only required for DFT-s-OFDM waveform. Refer to Subclause 6.3.1.4 of [6] for details on transform precoding (if applicable) applied at the transmitter.

Demodulation: Refer to Subclause 6.3.1.2 in [6] for details on modulation applied at the transmitter.

Descrambling: Refer to Subclause 6.3.1.1 in [6] for details on scrambling applied at the transmitter.

CB de-concatenation: Refer to Subclause 6.2.6 in [7] for details on CB concatenation applied at the transmitter.

Rate de-matching: Refer to Subclause 6.2.5 in [7] for details on rate matching applied at the transmitter.

LDPC decoding: Refer to Subclause 6.2.4 in [7] for details on LDPC encoding applied at the transmitter.

CB de-segmentation and CB CRC check: Refer to Subclause 6.2.3 in [7] for details on CB segmentation and CB CRC attachment applied at the transmitter.

TB CRC check: Refer to Subclause 6.2.1 in [7] for details on TB level CRC attachments applied at the transmitter.

The uplink physical layer processing for control channel (PUCCH) carrying UCI depends on PUCCH formats.

PUCCH format 0 processing consists of the following steps at the receiver (O-DU):

RE de-mapping: Refer to subclause 6.3.2.3.2 of [6] for details on RE mapping applied at the transmitter.

Sequence detection: The transmitted sequence (refer to Subclause 6.3.2.3 in [6] for details) is detected at O-DU using a non-coherent detector, since PUCCH format 0 does not carry any DM-RS. The detailed design is up to O-DU implementation.

PUCCH format 1 processing consists of the following steps at the receiver (O-DU):

RE de-mapping: Refer to Subclauses 6.3.2.4.2 and 6.4.1.3.1.2 of [6] for details on RE mapping applied at the transmitter.

Channel estimation and equalization: up to O-DU implementation.

Demodulation: Refer to Subclause 6.3.2.4.1 in [6] for details on modulation applied at the transmitter.

PUCCH formats 2/3/4 processing consists of the following steps at the receiver (O-DU):

RE de-mapping: Refer to Subclauses 6.3.2.5.3 and 6.4.1.3.2.2 (format 2); 6.3.2.6.5 and 6.4.1.3.3.2 (formats 3/4) of [6] for details on RE mapping applied at the transmitter.

Channel estimation and equalization: up to O-DU implementation.

Transform precoding (IDFT): optional, only required for DFT-s-OFDM waveform. Refer to Subclause 6.3.2.6.4 of [6] for details on transform precoding (applicable for formats 3/4) applied at the transmitter.

Demodulation: Refer to Subclause 6.3.2.5.2 (format 2) and 6.3.2.6.2 (formats 3/4) in [6] for details on modulation applied at the transmitter.

Descrambling: Refer to Subclause 6.3.2.5.1 (format 2) and 6.3.2.6.1 (formats 3/4) in [6] for details on scrambling applied at the transmitter.

Rate de-matching: Refer to Subclause 6.3.1.4 in [7] for details on rate matching applied at the transmitter.

Polar/Block decoding: Refer to Subclause 6.3.1.3 in [7] for details on Polar/Block encoding applied at the transmitter.

CRC check: Refer to Subclause 6.3.1.2 in [7] for details on CRC attachment applied at the transmitter.

The uplink physical layer processing for random access channel (PRACH) carrying preamble consists of the following steps at the receiver (O-DU):

RE de-mapping: Refer to Subclause 6.3.3.2 in [6] for details on RE mapping applied at the transmitter.

Root sequence correlation: Perform correlation operation between root sequence and received signals. Refer to Subclause 6.3.3.1 in [6] for details on root sequence generation.

IFFT: perform the inverse Fast Fourier Transform (iFFT) operation on the received signal(s).

Noise estimation: perform the noise estimation operation.

Peak search: detect the peak for different root sequences.

Preamble detection and Timing Advance (TA) or delay estimation: determine the preamble sequence(s) received and the corresponding timing advance estimate(s).

The uplink physical signals (SRS, PT-RS) do not carry any information from the higher layers (i.e. layer 2 and above).

The Sounding Reference Signal (SRS) in uplink is received at O-DU using the following steps:

RE de-mapping: Refer to Subclauses 6.4.1.4.3 and 6.4.1.4.4 in [6] for details on RE mapping applied at the transmitter.

Sequence detection and Channel estimation: Up to O-DU implementation. Refer to 6.4.1.4.2 in [6] for details on SRS sequence generation at the transmitter. Channel condition in uplink is estimated at the O-DU based on the processing of received SRS.

The Phase-Tracking Reference Signal (PT-RS) in uplink is received at the O-DU using the following steps:

RE de-mapping: Refer to Subclause 6.4.1.2.2 in [6] for details on RE mapping applied at the transmitter.

Sequence detection: Up to O-DU implementation. Refer to Subclause 6.4.1.2.1 in [6] for details on sequence generation at the transmitter.

An O-DU AAL profile for 5G NR uplink shall specify a set of accelerated functions corresponding to one or more than one physical uplink channel(s) and/or physical uplink signal(s).

In addition to the processing blocks mentioned above, each of these uplink physical channels/signals may include an additional functional block, viz. IQ decompression, which is implementation specific and may depend on system configuration/capability. Each of these physical channels/signals can be implemented with/without this optional functional block. The AAL Application interface shall expose to the AAL Application whether these functional blocks are supported or not within the AAL Implementation.

### O-DU Protocol Stack Reference for mMTC

Figure 7.1.2-1 illustrates the building blocks for processing various O-DU PHY layer Downlink (DL) channels and signals (with 7.2-x functional split between O-DU and O-RU) defined by 3GPP in [8] & [9] as part of 4G/5G NR specification.



Figure 7.1.2-1: O-DU PHY processing blocks for mMTC Downlink.

The O-DU PHY layer in downlink consists of the following physical channels and reference signals:

• Narrow-band Physical Downlink Shared Channel (NPDSCH).

• Narrow-band Physical Downlink Control Channel (NPDCCH).

• Narrow-band Physical Broadcast Channel (NPBCH).

• Narrow-band Primary Synchronization Signal (NPSS).

• Narrow-band Secondary Synchronization Signal (NSSS).

• Narrow-band Reference Signal (NRS) and Narrow-band Position Reference Signal (NPRS).

• Narrow-band Wake-Up Signal (NWUS)

The Narrow-band downlink physical channels (NPDSCH, NPDCCH, NPBCH) carry information originating from higher layers (i.e. layer 2 and above).

The Narrow-band downlink physical layer processing of data channel (NPDSCH) carrying transport blocks consists of the following steps:

TB CRC attachment: Error detection is provided on each transport block (TB) through a Cyclic Redundancy Check (CRC). Refer to Subclause 6.4 in [9] for details.

CRC attachment: CRC attachment is performed according to Subclauses 6.4 of [9]

Tail-biting Convolutional coding: Refer to Subclauses 6.2 and 5.1.3.1 in [9] for details.

Rate matching: Refer to Subclauses 6.4 in [9] for details.

Scrambling: Refer to Subclause 10.2.5.2 in [8] for details.

Modulation: Refer to Subclause 10.2.5.3 in [8] for details.

Layer mapping: Refer to Subclause 10.2.5.3in [8] for details.

RE mapping: Refer to Subclause 10.2.5.5in [8] for details on Resource Element (RE) mapping.

The downlink physical layer processing of control channel (PDCCH) carrying Downlink Control Information (DCI) consists of the following steps:

CRC attachment: Error detection is provided on DCI transmissions through a Cyclic Redundancy Check (CRC). Refer to Subclause 6.4 in [9] for details.

Tail-biting Convolutional coding: Refer to Subclauses 6.2 and 5.1.3.1 in [9] for details.

Scrambling: Refer to Subclause 10.2.5.2 in [8] for details.

Modulation: Refer to Subclause 10.2.5.3 in [8] for details.

Layer mapping: Refer to Subclause 10.2.5.3 in [8] for details.

RE mapping: Refer to Subclause 10.2.5.5 in [8] for details on Resource Element (RE) mapping.

The downlink physical layer processing of broadcast channel (NPBCH) carrying maximum one transport block consists of the following steps:

NPBCH payload generation: Refer to Subclause 6.4.1 in [9] for details.

TB CRC attachment: Error detection is provided on each transport block (TB) through a Cyclic Redundancy Check (CRC). Refer to Subclause 6.4 in [9] for details.

Scrambling: Refer to Subclause 10.2.5.2 in [8] for details.

Modulation: Refer to Subclause 10.2.5.3 in [8] for details.

Layer mapping: Refer to Subclause 10.2.5.3 in [8] for details.

RE mapping: Refer to Subclause 10.2.5.5 in [8] for details on Resource Element (RE) mapping.

The downlink physical signals (NRS, NPSS, NSSS, NPRS, NWUS) correspond to a set of resource elements used by the physical layer but does not carry information originated from higher layers (i.e., layer 2 and above).

Reference Signals and Synchronization signals (NPSS/NSSS) are generated using the following steps:

Sequence Generation and Modulation and RE mapping: Refer to Subclauses, 10.2.6B (NWUS), 10.2.7.1 (NPSS) and 10.2.7.2 (NSSS), 10.2.6 (NRS), 10.2.6A (NPRS) in [8] for details.

An O-DU AAL profile for 4G NR downlink shall specify a set of accelerated functions corresponding to one or more than one physical downlink channel(s) and/or physical downlink signal(s).

In addition to the processing blocks mentioned above, each of these downlink physical channels/signals may include some additional functional blocks (e.g., precoding, IQ compression) which are implementation specific and may also depend on system configurations/capabilities (for example, whether a O-DU is connected to a CAT-A/CAT-B O-RU). Each of these physical channels/signals can be implemented with/without these optional functional blocks. The AAL Application interface shall expose to the AAL Application whether these functional blocks are supported or not within the AAL Implementation.

Figure 7.1.2-2 illustrates the building blocks for processing various O-DU PHY layer Uplink (UL) channels and signals (with 7.2-x functional split between O-DU and O-RU) defined by 3GPP in [8] & [9] as part of 4G/5G NR specification.



Figure 7.1.2-2: O-DU PHY processing blocks for mMTC Uplink

The O-DU PHY layer in uplink consists of the following physical channels and reference signals:

• Narrow-band Physical Uplink Shared Channel (NPUSCH).

• Narrow-band Physical Random-Access Channel (NPRACH).

The uplink physical channels (NPUSCH, NPRACH) carry information originating from higher layers (i.e., layer 2 and above).

The uplink physical layer processing of shared channel (NPUSCH) carrying uplink data with or without Uplink Control Information (UCI) consists of the following steps at the receiver (O-DU):

RE (de)mapping: Refer to Subclauses 5.3.4 of [8] for details on RE mapping at the transmitter/receiver.

Channel estimation and equalization: up to O-DU implementation.

Transform precoding (IDFT): optional, only required for DFT-s-OFDM waveform. Refer to Subclause 5.3.3A of [8] for details on transform precoding (if applicable) applied at the transmitter.

Demodulation: Refer to Subclause 5.3.2 in [8] for details on modulation applied at the transmitter.

Descrambling: Refer to Subclause 5.3.1 in [8] for details on scrambling applied at the transmitter.

Rate de-matching: Refer to Subclause 5.1.4.1 in [9] for details on rate matching applied at the transmitter.

Turbo decoding: Refer to Subclause 5.1.3.2 in [9] for details on Turbo decoding applied at the transmitter.

CRC check: Refer to Subclauses 5.1.1 in [9] for details on TB and CB level CRC attachments applied at the transmitter.

The uplink physical layer processing for shared channel (NPUSCH) carrying UCI depends on NPUSCH formats.

An O-DU AAL profile for 4G/5G NR uplink shall specify a set of accelerated functions corresponding to one or more than one physical uplink channel(s) and/or physical uplink signal(s).

In addition to the processing blocks mentioned above, each of these uplink physical channels/signals may include an additional functional block, viz. IQ decompression, which is implementation specific and may depend on system configuration/capability. Each of these physical channels/signals can be implemented with/without this optional functional block. The AAL Application interface shall expose to the AAL Application whether these functional blocks are supported or not within the AAL Implementation.

### O-DU AAL Profile Definitions

O-DU AAL profiles are defined below with future specification(s) to define the AAL Application interface for each profile.

#### Profile Definitions General Guidelines

##### Naming

As discussed above O-DU AAL profiles are specific to one or more physical channel(s) or signal(s) as such should follow the naming guidelines

* O-DU AAL profiles shall be prefixed with “AAL\_”
* O-DU AAL profiles when specific to a single channel or signal shall include the channel or signal in the name e.g. “AAL\_PUSCH”
* O-DU AAL profiles when common across multiple channels or signals shall not include the channel or signal name, instead just reference the Accelerated Function(s), e.g. AAL\_RE-MAPPING
* O-DU AAL profiles that include a subset of the functional blocks within a channel or signal shall include a functional description after the channel name e.g. AAL\_PUSCH\_CHANNEL\_ESTIMATION

##### Data Flow

O-DU AAL profiles shall specify the data flow supported by the AAL Profile.

Look aside data flow implies the remaining functions not included in the Profile that comprise the channel or signal are implemented on the AAL Application or other entity associated with the AAL Application and not implemented in the HW Accelerator.

Inline data flow implies that the signal (with a 7-2x PHY functional split) and the IQ data from the AAL Application (in DL) or the decoded bits (in UL) (post processing) are transferred directly from the HW Accelerator to the Fronthaul interface (in DL) or to the AAL Application (in UL).

The profile shall specify if the data flow includes only user plane, or only control plane, or both.

#### O-DU AAL Profiles

##### AAL\_MU-MIMO\_PRECODER\_WEIGHTS\_CALC



Figure 7.1.3.2.1-1: AAL\_MU-MIMO\_PRECODER\_WEIGHTS\_CALC

The AAL\_MU-MIMO\_PRECODER\_WEIGHTS\_CALC is used by AAL Application to offload beamforming (precoding) weight calculation to the hardware accelerator (HWA) in look-aside acceleration mode. The AAL Application shall provide HWA with all the information required to calculate precoding weights.

This profile is implemented as a look aside accelerator.

The below figure shows an example use of the AAL\_MU-MIMO\_PRECODER\_WEIGHTS\_CALC in an O-DU with AAL\_DOWNLINK\_HIGH-PHY and AAL\_UPLINK\_HIGH-PHY



Figure 7.1.3.2.1-2: Example AAL\_MU-MIMO\_PRECODER\_WEIGHTS\_CALC use.

##### AAL\_FFT



Figure 7.1.3.2.2-1: AAL\_FFT

The AAL\_FFT is used by application to offload FFT/iFFT processing to the hardware accelerator (HWA) in look- aside acceleration mode. The application shall provide HW Accelerator with all the information required to perform the FFT operations. The AAL\_FFT Profile can be used for 3GPP specification 38.211 clause 5.3, and clause 6.4.1.4. The below list and Figure 7.1.3.2.2-1: AAL\_FFT highlights the set of accelerated functions that define the AAL\_FFT Profile.

* Zero Padding
* iDFT
* Windowing
* DFT
* De padding

The below figure shows an example use of the AAL\_FFT Profile for accelerating the SRS processing in an O-DU. The highlighted blocks show the set the accelerated functions.



Figure 7.1.3.2.2-2: AAL\_FFT example for SRS Processing

This profile is implemented as a look aside accelerator.

#### O-DU AAL Profiles for Downlink

##### AAL\_PDSCH\_FEC

Figure 7.1.3.3.1-1 highlights the set of accelerated functions that define the AAL\_PDSCH\_FEC Profile. These include:

* CRC Generation
* LDPC Encoding
* PDSCH Rate Matching

The AAL\_PDSCH\_FEC Profile is implemented as a look aside accelerator. The AAL\_PDSCH\_FEC Profile will support both Transport Block and Code Block operations.



Figure 7.1.3.3.1-1: AAL\_PDSCH\_FEC Profile

##### AAL\_PDSCH\_HIGH-PHY

Figure 7.1.3.3.2-1 highlights the set of accelerated functions that defines the AAL\_PDSCH\_HIGH-PHY Profile, which includes the processing of PDSCH TB(s) and associated DM-RS.

The set of accelerated functions associated with the processing of PDSCH TB(s) is as follows:

• TB CRC attachment

• CB segmentation and CRC attachment

• LDPC encoding

• Rate matching

• CB concatenation

• Scrambling

• Modulation

• Layer mapping

• Precoding [[1]](#footnote-2)

• RE mapping

• IQ compression1

The set of accelerated functions associated with the processing of PDSCH DM-RS is as follows:

• PDSCH DM-RS sequence generation

• Modulation

• Precoding1

• RE mapping

• IQ compression1

The AAL\_PDSCH\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.3.2-1: AAL\_PDSCH\_HIGH-PHY Profile

##### AAL\_PDCCH\_HIGH-PHY

Figure 7.1.3.3.3-1 highlights the set of accelerated functions that defines the AAL\_PDCCH\_HIGH-PHY Profile, which includes the processing of PDCCH DCI and associated DM-RS.

The set of accelerated functions associated with the processing of PDCCH TB(s) is as follows:

• CRC attachment

• Polar encoding

• Rate matching

• Scrambling

• Modulation (QPSK)

• Precoding1

• RE mapping

• IQ compression1

The set of accelerated functions associated with the processing of PDCCH DM-RS is as follows:

• PDCCH DM-RS sequence generation

• Modulation

• Precoding1

• RE mapping

• IQ compression1

The AAL\_PDCCH\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.3.3-1: AAL\_PDCCH\_HIGH-PHY Profile

##### AAL\_PBCH\_HIGH-PHY

Figure 7.1.3.3.4-1 highlights the set of accelerated functions that defines the AAL\_PBCH\_HIGH-PHY Profile, which includes the processing of PBCH TB and associated DM-RS, PSS and SSS, or in other words, the processing of SSB.

The set of accelerated functions associated with the processing of PBCH TB is as follows:

• PBCH payload generation

• Scrambling

• TB CRC attachment

• Polar encoding

• Rate matching

• Data scrambling

• Modulation (QPSK)

• Precoding1

• RE mapping

• IQ compression1

The set of accelerated functions associated with the processing of PBCH DM-RS/PSS/SSS is as follows:

• PDCCH DM-RS/PSS/SSS sequence generation

• Modulation

• Precoding1

• RE mapping

• IQ compression1

The AAL\_PBCH\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.3.4-1: AAL\_PBCH\_HIGH-PHY Profile

##### AAL\_CSI-RS\_HIGH-PHY

Figure 7.1.3.3.5-1 highlights the set of accelerated functions that defines the AAL\_CSI-RS\_HIGH-PHY Profile, which includes the following:

• CSI-RS sequence generation

• Modulation

• Precoding1

• RE mapping

• IQ compression1

The AAL\_CSI-RS\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.3.5-1: AAL\_CSI-RS\_HIGH-PHY Profile

##### AAL\_PT-RS-DL\_HIGH-PHY

Figure 7.1.3.3.6-1 highlights the set of accelerated functions that defines the AAL\_PT-RS-DL\_HIGH-PHY Profile, which includes the following:

• PT-RS sequence generation

• Modulation

• Precoding1

• RE mapping

• IQ compression1

The AAL\_PT-RS-DL\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.3.6-1: AAL\_PT-RS-DL\_HIGH-PHY Profile

##### AAL\_DOWNLINK\_HIGH-PHY

Figure 7.1.3.3.7-1 highlights the set of accelerated functions that defines the AAL\_DOWNLINK\_HIGH-PHY Profile. This profile includes the aggregation of all the individual downlink channel profiles as follows:

**PDSCH**

* **Data**: see list of accelerated functions associated with the processing of PDSCH TB(s), per section 7.1.3.3.2
* **DM-RS**: see list of accelerated functions associated with the processing of PDSCH DM-RS, per section 7.1.3.3.2.
* **PT-RS**: see list of accelerated functions listed in section 7.1.3.3.6.

**PDCCH**:

* **Data**: see list of accelerated functions associated with the processing of PDCCH TB(s), per section 7.1.3.3.3
* **DM-RS**: see list of accelerated functions associated with the processing of PDCCH DM-RS, per section 7.1.3.3.3

**SSB**:

* PSS + SSS: see list of accelerated functions associated with the processing of PSS/SSS, per section 7.1.3.3.4
* PBCH DM-RS: see list of accelerated functions associated with the processing of PBCH DM-RS, per section 7.1.3.3.4
* PBCH: see list of accelerated functions associated with the processing of PBCH TB(s), per section 7.1.3.3.4

**CSI-RS:**

* see list of accelerated functions listed in section 7.1.3.3.5

The AAL\_DOWNLINK\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.3.7-1AAL\_DOWNLINK\_ HIGH-PHY Profile

#### O-DU AAL Profiles for Uplink

##### AAL\_PUSCH\_FEC

Figure 7.1.3.4.1-1 highlights the set of accelerated functions that define the AAL\_PUSCH\_FEC Profile. These include:

* PUSCH Rate De-matching
* LDPC Decoder
* CRC Check

The AAL\_PUSCH\_FEC Profile is implemented as a look aside accelerator. The AAL\_PUSCH\_FEC Profile will support both Transport Block and Code Block operations.



Figure 7.1.3.4.1-1: AAL\_PUSCH\_FEC Profile

##### AAL\_PUSCH\_HIGH-PHY

Figure 7.1.3.4.2-1 highlights the set of accelerated functions that defines the AAL\_PUSCH\_HIGH-PHY Profile, which includes the processing of PUSCH data (with or without UCI).

The set of accelerated functions associated with the processing of PUSCH data is as follows:

• IQ decompression1

• RE de-mapping

• Channel estimation

• Channel equalization

• Transform precoding (optional- only required for DFT-s-OFDM waveform)

• Demodulation

• Descrambling

• Rate de-matching

• LDPC decoding

• CRC check

The AAL\_PUSCH\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.4.2-1 : AAL\_PUSCH\_HIGH-PHY Profile

##### AAL\_PUCCH\_HIGH-PHY

Figure 7.1.3.4.3.1-1, Figure 7.1.3.4.3.2-1 and Figure 7.1.3.4.3.2-1 highlight the set of accelerated functions that defines the AAL\_PUCCH\_HIGH-PHY Profile, which includes the processing of UCI.

The set of accelerated functions associated with the processing of PUCCH UCI depends on the PUCCH format being configured by the AAL Application.

###### PUCCH format 0

The set of accelerated functions associated with the processing of PUCCH UCI using PUCCH format 0 is as follows:

• IQ decompression1

• RE de-mapping

• Sequence detection



Figure 7.1.3.4.3.1-1: AAL\_PUCCH\_HIGH-PHY Profile (PUCCH format 0)

###### PUCCH format 1

The set of accelerated functions associated with the processing of PUCCH UCI using PUCCH format 1 is as follows:

• IQ decompression1

• RE de-mapping

• Channel estimation

• Channel equalization

• Demodulation



Figure 7.1.3.4.3.2-1: AAL\_PUCCH\_HIGH-PHY Profile (PUCCH format 1)

###### PUCCH format 2/3/4

The set of accelerated functions associated with the processing of PUCCH UCI using PUCCH format 2/3/4 is as follows:

• IQ decompression1

• RE de-mapping

• Channel estimation

• Channel equalization

• Transform precoding (optional- only required for DFT-s-OFDM waveform)

• Demodulation

• Descrambling

• Rate de-matching

• Polar/Block decoding

• CRC check



Figure 7.1.3.4.3.3-1: AAL\_PUCCH\_HIGH-PHY Profile (PUCCH format 2/3/4)

The AAL\_ PUCCH\_ HIGH -PHY profile is executed in inline acceleration mode.

##### AAL\_PRACH\_HIGH-PHY

Figure 7.1.3.4.4-1 highlights the set of accelerated functions that defines the AAL\_PRACH\_HIGH-PHY Profile.

The set of accelerated functions associated with the processing of PRACH preamble is as follows:

• IQ decompression1

• RE de-mapping

• Root sequence generation and correlation

• IFFT

• Noise estimation

• Peak search for power delay profile

• Preamble detection and delay/timing advance estimation



Figure 7.1.3.4.4-1: AAL\_PRACH\_HIGH-PHY Profile

The AAL\_PRACH\_HIGH-PHY Profile is executed in inline acceleration mode.

##### AAL\_SRS\_HIGH-PHY

Figure 7.1.3.4.5-1 highlights the set of accelerated functions that defines the AAL\_SRS\_HIGH-PHY Profile.

The set of accelerated functions associated with the processing of SRS is as follows:

• IQ decompression1

• RE de-mapping

• Channel estimation



Figure 7.1.3.4.5-1: AAL\_SRS\_HIGH-PHY Profile

The AAL\_SRS\_ HIGH-PHY Profile is executed in inline acceleration mode.

##### AAL\_PT-RS-UL\_HIGH-PHY

Figure 7.1.3.4.6-1 highlights the set of accelerated functions that defines the AAL\_PT-RS-UL\_HIGH-PHY Profile.

The set of accelerated functions associated with the processing of PT-RS-UL sequence is as follows:

• IQ decompression1

• RE de-mapping

• Sequence detection



Figure 7.1.3.4.6-1: AAL\_PT-RS-UL\_HIGH-PHY profile

The AAL\_PT-RS-UL\_HIGH-PHY profile is executed in inline acceleration mode.

##### AAL\_UPLINK\_HIGH-PHY

Figure 7.1.3.4.7-1 highlights the set of accelerated functions that defines the AAL\_UPLINK\_HIGH-PHY Profile, this profile includes the aggregation of all the individual uplink channel profiles as follows:

**PUSCH**:

* **Data:** see list of accelerated functions associated with the processing of PUSCH data, per section 7.1.3.4.2
* **DM-RS**: see list of accelerated functions listed in section 7.1.3.4.6, implemented to process DM-RS IQ samples.
* **PT-RS:** see list of accelerated functions listed in section 7.1.3.4.6

**PUCCH:**

* **Format 0:** see list of accelerated functions listed in section 7.1.3.4.3.1
* **Format 1:** 
  + **UCI:** see list of accelerated functions listed in section 7.1.3.4.3.2
  + **DM-RS:** see list of accelerated functions listed in section 7.1.3.4.6, implemented to process DM-RS IQ samples.
* **Formats 2/3/4:**
  + **UCI:** see list of accelerated functions listed in section 7.1.3.4.3.3
  + **DM-RS:** see list of accelerated functions listed in section 7.1.3.4.6, implemented to process DM-RS IQ samples.

**PRACH:**

* see list of accelerated functions listed in section 7.1.3.4.4

**SRS:**

* see list of accelerated functions listed in section 7.1.3.4.5

The AAL\_UPLINK\_HIGH-PHY Profile is executed in inline acceleration mode.



Figure 7.1.3.4.7-1: AAL\_UPLINK\_ HIGH-PHY Profile

#### O-DU AAL Profiles for mMTC

##### AAL\_NPDSCH\_FEC

Figure 7.1.3.5.1-1: highlights the set of accelerated functions that define the AAL\_NPDSCH\_FEC Profile. These include:

• CRC Generation

• Tail-Biting Convolutional Coding

• Rate Matching

The AAL\_NPDSCH\_FEC Profile is implemented as a look aside accelerator.



Figure 7.1.3.5.1-1: AAL\_NPDSCH\_FEC Profile

##### AAL\_NPDCCH\_FEC

Figure 7.1.3.5.2-1 highlights the set of accelerated functions that define the AAL\_NPDCCH\_FEC Profile. These include

• CRC Generation

• Tail-Biting Convolutional Coding

• Rate Matching

The AAL\_NPDCCH\_FEC Profile is implemented as a look aside accelerator.



Figure 7.1.3.5.2-1: AAL\_NPDCCH\_FEC Profile

##### AAL\_NPBCH\_FEC

Figure 7.1.3.5.3-1 highlights the set of accelerated functions that define the AAL\_NPBCH\_FEC Profile. These include

• CRC Generation

• Tail-Biting Convolutional Coding

• Rate Matching

The AAL\_NPBCH\_FEC Profile is implemented as a look aside accelerator.



Figure 7.1.3.5.3-1: AAL\_NPBCH\_FEC Profile

##### AAL\_NPUSCH\_FEC

Figure 7.1.3.5.4-1 highlights the set of accelerated functions that define the AAL\_NPUSCH\_FEC Profile. These include

• CRC Generation

• Turbo Decoding

• Rate Matching

The AAL\_NPUSCH\_FEC Profile is implemented as a look aside accelerator.



Figure 7.1.3.5.4-1: AAL\_NPUSCH\_FEC Profile

## O-CU AAL Profiles

The O-CU AAL profiles shall be part of further study for O-RAN WG6.

Annex (informative):   
Change History

|  |  |  |
| --- | --- | --- |
| Date | Revision | Description |
| 2024.07.19 | 09.00.05 | Minor editorial fix to figure 4.2.2-1 archicture diagram, change O1dms to O1. |
| 2024.07.04 | 09.00.04 | Editorial changes to update figures throughout the document.  Updated the number of the LPU’s on architecture diagram. Terminate IMS interface to HAM in example HAM software deployment scenarios. |
| 2024.07.04 | 09.00.03 | Clause 4.2.3 add requirement stating that the scope of the HAM identifier is within an O-Cloud instance. |
| 2024.07.04 | 09.00.02 | Adding picture for inline example in General Aspetcs section as it can be confusing for reader as inline is missing from general discription. |
| 2024.05.09 | 09.00.01 | Editorial changes for removal of “cloudifed RAN” which is not definied on higher levels |
| 2024.03.22 | 08.00.06 | Updated with minor editorial fixes for publication. |
| 2024.03.07 | 08.00.05 | Modified high level architecture diagram with deletion of the interface between DMS and NF so that readers are not confused about API which is not existent yet and is out of scope of this document. |
| 2024.03.07 | 08.00.04 | Updates from CR [ERI-2024.1.23-WG6-CR-0046-AAL GAP CADS reference correction.docx](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/ERI-2024.1.23-WG6-CR-0046-AAL%20GAP%20CADS%20reference%20correction.docx?api=v2)  1. “VOID” the reference to the CADS TR in the normative reference section.  2. Change the reference to CADS for O-Cloud Platform Software to a reference of the WG1 OAD TS.  3. Change the reference to CADS for the synchronization topologies to the WG4 CUS TS. |
| 2024.03.07 | 08.00.03 | 1. Correct definitions of Accelerated Function and AAL Profile to remove the constraint that it be a “Cloudified Network Function” (that term is no longer defined anyway).  2. Add clarification text to figure 4.2-1 to point out that support of O1 by the AAL Application is not a requirement.  3. Move the text at the bottom of the terms and conditions section to section 4.2. The text is unchanged.  4. Update Figure 4.2-5 to change “NF Application” to “AAL Application”. |
| 2024.01.18 | 08.00.02 | To improve readability and improve clarity, existing text was moved to different clauses and additional clarification points were considered from [DCM-2023.10.24-WG6-GAP-updates\_section\_2.2\_v03](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/DCM-2023.10.24-WG6-GAP-updates_section_2.2_v03.docx?api=v2) |
| 2024.01.18 | 08.00.01 | Updated diagrams to align format. |
| 2023.11.28 | 08.00 | Published from version 07.00.09 |
| 2023.10.14 | 07.00.09 | Editorial Updates, fix broken reference. |
| 2023.10.11 | 07.00.08 | Update format to latest O-RAN TS template |
| 2023.10.11 | 07.00.07 | Add text that describes the DMS to AAL Application interface. <https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/QCM-2023.10.24-WG6-CR-0048-Add_AAL_DMS_Description.docx?api=v2>  Editorial updates. |
| 2023.10.11 | 07.00.06 | Avoid mixing definitions with characteristics and requirements, clause 1.3.1 and 4.1. Updated definition section to provide additional clarity. |
| 2023.09.11 | 07.00.05 | Editorial updates in clauses 1.2, 1.3.2, 2.1, 3.1,4.2 <https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/DCM-2023.10.24-WG6-GAP_editorial_updates_v03.docx?api=v2> |
| 2023.09.11 | 07.00.04 | Removed reference to alarms which are no longer in scope of this document. |
| 2023.09.11 | 07.00.03 | Add example diagrams of how AAL Software drivers are deployments, added clarification on how AAL SW drivers are installed and managed.  <https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/INT-10-06-23-WG6-CR-18-AAL-GAnP-driver-deployments-clarification.docx?api=v2> |
| 2023.09.11 | 07.00.02 | Captured the architectural aspects for use of AAL APIs without impacting the O-RAN architecture. <https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/ERI.AO-2023.08.03-WG6-CR-006-AAL%20GAP%20Agreements%20to%20support%20open%20multi-vendor%20Interfaces%20for%20AAL%20APIs.docx?api=v2> |
| 2023.09.11 | 07.00.01 | Add description of a software repository use by HAM in section 1.3.1. <https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/ERI-2023.08.08-WG6-CR-005-AAL%20GAP%20SW%20Repo%20Clarifications.docx?api=v2> |
| 2023.14.07 | 06.00.07 | Implemented CR from <https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/INT-VMW-06-30-23-WG6-CR-AAL-GAnP-Editorial-rearrange-principles-and-arch_v1.docx?api=v2> Move 2.5 to chapter 3 and move 3.1 & 3.2 to chapter 2 to better align with chapter contents. |
| 2023.14.07 | 06.00.06 | Implemented CR from [SLA-2023.05.16-ORAN.WG6-CR 03-AAL-GAnP-R003-v06.00.doc](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/SLA-2023.05.16-ORAN.WG6-CR%20003-AAL-GAnP-R003-v06.00.docx?api=v2). State that HAM is part of and is integrated along with O-Cloud Platform software |
| 2023.30.06 | 06.00.05 | Implemented CR from [VMW-2023.05.31-WG6-CR-AAL%20Architecture%20in%20GAnP.docx](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/VMW-2023.05.31-WG6-CR-AAL%20Architecture%20in%20GAnP.docx?api=v2) |
| 2023.30.06 | 06.00.04 | Implemented CR from [SLA-2023.05.17-ORAN.WG6.CR%20004-AAL-GAnP-R003-v06.00.docx](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/SLA-2023.05.17-ORAN.WG6.CR%20004-AAL-GAnP-R003-v06.00.docx?api=v2) |
| 2023.30.06 | 06.00.03 | Implemented CR from [DCM-2023.05.17-WG6-AAL-ETSI](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/DCM-2023.05.17-WG6-AAL-ETSI%20NFV%20description%20update%20in%20Gap.docx?api=v2) |
| 2023.30.06 | 06.00.02 | Implemented CR from [QCM-2023.04-24-WG6-CR-0034-AAL-GAnP-Clarify-AAL-Mgmt-Functions.docx](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/QCM-2023.04-24-WG6-CR-0034-AAL-GAnP-Clarify-AAL-Mgmt-Functions.docx?api=v2) |
| 2023.27.04 | 06.00.01 | Implemented CR from [ERI-202304.02-WG6\_CR\_AAL-GAnP Editorial.docx](https://oranalliance.atlassian.net/wiki/download/attachments/2210398415/ERI-202304.02-WG6_CR_AAL-GAnP%20Editorial.docx?api=v2%20%20) and various editorial comments from [July Train WIKI](https://oranalliance.atlassian.net/wiki/spaces/COWG/pages/2813329465/July+2023+AAL+CR+Comment+Wiki) |
| 2023.10.03 | 05.00.04 | Definition of new FFT Profile |
| 2023.10.03 | 05.00.03 | Describe the chaining of AAL Profile Instnaces in Sec 4.1.6 AAL-LPU Principles |
| 2023.10.03 | 05.00.02 | Modification of Figure 2.7 and 2.8 in GAnP, changes to sections 2.5.2 and 5.1.3.1.2.. |
| 2022.09.12 | 05.00.01 | Implemented CR from QCM-24 Editorial modifications to correct AAL terms |
| 2022.11.07 | 04.00.04 | Implemented CR from INT-0014 (example AAL-LPU mapping diagrams) |
| 2022.11.07 | 04.00.03 | Implemented CR from QCM-0013 (ER Diagram) |
| 2022.11.07 | 04.00.02 | Implemented CR from QCM-0020 (AAL App Definition) |
| 2022.10.25 | 04.00.01 | Implemented CRs NOK-0001 (AALProfile\_v02) |
| 2022.09.02 | 04.00 | Final version 04.00 |

1. Configurable functional block, depends on implementation and/or system configuration [↑](#footnote-ref-2)