O-RAN.WG6.AAL Common API-R003-v08.00

*Technical Specification*

**O-RAN Working Group 6 (Cloudification and Orchestration Workgroup)**

**Acceleration Abstraction Layer Common API**

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

[Chapter 1 Introductory Material 4](#_bookmark0)

[1.1 Scope 4](#_bookmark1)

[1.2 References 4](#_bookmark2)

[1.3 Definitions and Abbreviations 5](#_bookmark11)

* + 1. [Definitions 5](#_bookmark12)
    2. [Abbreviations 5](#_bookmark13)

[Chapter 2 Overview 6](#_bookmark14)

* 1. [General 6](#_bookmark15)
  2. [Document Structure 6](#_bookmark16)

[Chapter 3 AAL Common requirements and Object Model 7](#_bookmark17)

* 1. [AALI-C Mgmt Requirements 7](#_bookmark18)
     1. [Functional requirements for the HAM 8](#_bookmark19)
  2. [AALI-C App Requirements 10](#_bookmark20)
     1. [AALI-C-APP Transport Abstraction Requirements and Operations 10](#_bookmark21)
  3. [AAL Information Model 11](#_bookmark22)
     1. [Overview 11](#_bookmark23)
     2. [AALI-C-Mgmt Information Model 12](#_bookmark24)
        1. [Imported and associated information entities 12](#_bookmark25)
           1. [Associated information entities and local labels 12](#_bookmark26)
        2. [Class Diagram 12](#_bookmark27)
        3. [Class Definitions 14](#_bookmark28)
     3. [AALI-C-App Information Model 25](#_bookmark29)
        1. [Imported and associated information entities 25](#_bookmark30)
        2. [Class Diagram 25](#_bookmark31)
        3. [Class Definitions 27](#_bookmark32)
     4. [Naming convention and managed resource tags 29](#_bookmark33)
        1. [API managed resources tags 29](#_bookmark34)
        2. [API Naming Conventions 30](#_bookmark35)
        3. [Model Conventions 30](#_bookmark36)

[Chapter 4 AALI-C-Mgmt 30](#_bookmark37)

* 1. [AALI-C-Mgmt operations and events 30](#_bookmark38)
  2. [AALI-C-Mgmt use case workflows 32](#_bookmark39)
     1. [HW Accelerator installation/bootstrap use case workflow 32](#_bookmark40)
        1. [Sequence Diagram 33](#_bookmark41)
     2. [end-2end Life Cycle Management use case workflow (Informational) 35](#_bookmark42)
        1. [Sequence Diagram 37](#_bookmark43)
     3. [HW Accelerator Manager (HAM) registration use case workflow 41](#_bookmark44)
        1. [Sequence Diagram 42](#_bookmark45)
  3. [AALI-C-Mgmt API details 43](#_bookmark46)
     1. [Capabilities discovery and configuration 43](#_bookmark47)
        1. [getAalHwAccelStatus 43](#_bookmark48)
        2. [getAalLpuStatus 44](#_bookmark49)
        3. [getAalAccelInfo 45](#_bookmark50)
        4. [setAalAccelConfig 48](#_bookmark51)
        5. [startAalLpu 51](#_bookmark52)
        6. [stopAalLpu 52](#_bookmark53)
        7. [Inventory Notifications 52](#_bookmark54)
        8. [setAalAccelIdentifier 53](#_bookmark55)
        9. [aalHamRegistrationNotification 54](#_bookmark56)
     2. [Infrastructure Monitoring Service 55](#_bookmark57)
        1. [Fault Service Operations 55](#_bookmark58)
        2. [Fault Notifications 56](#_bookmark59)

[Chapter 5 AALI-C-APP 60](#_bookmark60)

* 1. [AALI-C-App Identification & Representation 60](#_bookmark61)
  2. [AALI-C-App operations and events 61](#_bookmark62)
  3. [AALI-C-App Operations and Events Sequence and State Diagram 62](#_bookmark63)
     1. [AALI-C-App State 62](#_bookmark64)
     2. [AALI-C-App API Init Sequence 64](#_bookmark65)
     3. [AALI-C-Transport API Sequence 66](#_bookmark66)
        1. [Allocation of AAL Buffer(s) 66](#_bookmark67)
        2. [Procedure for Buffer(s) Send and Receive 70](#_bookmark68)
  4. [AALI-C-App API details 90](#_bookmark69)
     1. [Capabilities discovery and configuration 90](#_bookmark70)
        1. [initAal 90](#_bookmark71)
        2. [cleanupAal 90](#_bookmark72)
        3. [getAalLpuInfo 91](#_bookmark73)
     2. [AALI-C-App Configuration Services 92](#_bookmark74)
        1. [createAalProfileInstance 92](#_bookmark75)
        2. [setAalProfileInstanceConfig 93](#_bookmark76)
        3. [deleteAalProfileInstance 93](#_bookmark77)
        4. [setAalProfileQueueConfig 94](#_bookmark78)
        5. [getAalProfileQueueConfig 95](#_bookmark79)
        6. [getAalProfileInstanceConfig 96](#_bookmark80)
     3. [AALI-C-App Transport Operations 97](#_bookmark81)
        1. [createBufferPool 97](#_bookmark82)
        2. [allocBuffer 98](#_bookmark83)
        3. [getBufferAddr 100](#_bookmark84)
        4. [getBufferSize 100](#_bookmark85)
        5. [sendBuffer 101](#_bookmark86)
        6. [receiveBuffer 103](#_bookmark87)
        7. [freeBuffer 104](#_bookmark88)
        8. [destroyBufferPool 105](#_bookmark89)
     4. [AALI-C-App Administrative Operations 105](#_bookmark90)
        1. [startAalProfileInstance 105](#_bookmark91)
        2. [stopAalProfileInstance 106](#_bookmark92)
        3. [startAalProfileQueue 107](#_bookmark93)
        4. [stopAalProfileQueue 107](#_bookmark94)

[Annex A (normative): Acceleration Capability Requirements 108](#_bookmark95)

* 1. [Capability requirements used in the placement of an AAL Application to an AF 108](#_bookmark96)

# Chapter 1 Introductory Material

## Scope

This Technical Specification has been produced by the O-RAN Alliance.

The contents of the present document are 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 release date and an increase in version number as follows:

Release x.y.z where:

1. the first digit is incremented for all changes of substance, i.e., technical enhancements, corrections, updates, etc. (the initial approved document will have x=01).
2. the second digit is incremented when editorial only changes have been incorporated in the document.
3. the third digit included only in working versions of the document indicating incremental changes during the editing process.

The present document describes the functional requirements for the Hardware Accelerator Manager (HAM) and interface requirements for the AAL Common API (i.e., AALI-C-Mgmt and AALI-C-App interfaces). It also provides the specification of the AALI-C-Mgmt and AALI-C-App interfaces as well as the information elements exchanged over those interfaces.

## References

1. 3GPP-TR-21.905: “Vocabulary for 3GPP Specifications”.
2. Void
3. O-RAN O-RAN.WG1.O-RAN-Architecture-Description
4. O-RAN O2 Interface General Aspects and Principles
5. O-RAN Cloudification and Orchestration Use Cases and Requirements for O-RAN Virtualized RAN
6. O-RAN O2ims Interface Specification
7. O-RAN Acceleration Abstraction Layer General Aspects and Principles
8. O-RAN Acceleration Abstraction Layer FEC Profiles
9. O-RAN Acceleration Abstraction Layer High-PHY Profiles
10. 3GPP TS 28.620: Telecommunication management; Fixed Mobile Convergence (FMC); Federated Network Information Model (FNIM); Umbrella Information Model (UIM))
11. 3GPP TS 28.622: Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS)
12. 3GPP TS 32.156: " Model repertoire".
13. OAS 3.0 Specification: [OpenAPI Specification](https://swagger.io/specification/v3/)
14. ITU-T X.731: Information Technology - Opens Systems Interconnection: State Management Function
15. IETF RFC 4122: “A Universally Unique IDentifier (UUID) URN Namespace”
16. IETF RFC 3986: IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax"

## Definitions and Abbreviations

### Definitions

For the purposes of the present document, the terms and definitions given in O-RAN Acceleration Abstraction Layer General Aspects and Principles [[7]](#_bookmark5) apply.

UUID Universally Unique Identifier [[15]](#_bookmark9) URI Uniform Resource Identifier [[16]](#_bookmark10)

### Abbreviations

For the purposes of the present document, the terms and definitions given in the O-RAN Acceleration Abstraction Layer General Aspects and Principles [[7]](#_bookmark5) apply. There are no additional abbreviations that are required aside from those in O-RAN Acceleration Abstraction Layer General Aspects and Principles [[7].](#_bookmark5)

# Chapter 2 Overview

## General

The AAL General Aspects and Principles are described in [[7]](#_bookmark5) and the AAL architecture including the AAL interfaces are depicted in figure 4.2-1 in [[7].](#_bookmark5)

The set of interfaces used to expose AAL functionality is the following:

* + 1. AALI-C: AAL-Profile independent set of interfaces addressing all the independent aspects of the underlying AAL Implementation(s) within an O-Cloud platform.
    2. AALI-P: A set of AAL-Profile specific interfaces dependent upon the AAL-Profile selected e.g., FEC. The AALI-P interfaces are specified in FEC[[8]](#_bookmark6) and High-PHY specifications [[9].](#_bookmark7)

Within the AALI-C interface set there are two interfaces:

1. AALI-C-Mgmt: Interface used for common administrative operations, actions, and events between the O- Cloud IMS and the HW Accelerator Manager (HAM) for resources exposed by this interface. The IMS provides the O2 interface towards the SMO. The HAM exposes the AALI-C-Mgmt interface towards IMS.
2. AALI-C-App: Interface used for common operations/actions/events toward an AAL Application for resources exposed by this interface. These operations are between the AAL Implementation and the AAL Application (AAL-App).

The present document specifies the list of AALI-C-Mgmt and AALI-C-App interface operations and events, description with input/outputs, information models, object models, state transition diagrams and requirements.

NOTE: The scope of the interfaces specification in this document is between the O-Cloud IMS and the HAM as well as between the AAL Implementation and the AAL Application. Interactions between the IMS and SMO is part of the O2 IMS specification. [[6]](#_bookmark4)

## Document Structure

This present document is structured as follows:

Chapter 2 presents the overview and main purpose of this specification. Chapter 3 presents a list of requirements and object model.

Chapter 4 presents the initialization and flow, operations and events associated with AALI-C-Mgmt. Chapter 5 presents the initialization and flow, operations and events associated with AALI-C-App. Annex A identifies the common acceleration capability requirements used for placing AFs.

# Chapter 3 AAL Common requirements and Object Model

## AALI-C Mgmt Requirements

The following table described the AAL-C-Mgmt interface requirements. The HAM shall support producing the AAL- C-Mgmt interface. The consumer of the interface is IMS.

**Table 3.1-1: AALI-C-Mgmt Interface Requirements**

|  |  |  |
| --- | --- | --- |
|  | **Description** | **Additional Info (TBD)** |
| [REQ-AALI-C-MGMT1a] | The AALI-C-Mgmt interface shall support unique identifiers for the HW Accelerators within an O-Cloud. |  |
| [REQ-AALI-C-MGMT1b] | The AALI-C-Mgmt interface shall support unique identifiers for the AAL-LPUs within an O-Cloud. |  |
| [REQ-AALI-C-MGMT2] | The AALI-C-Mgmt interface shall support querying inventory and capabilities information of the HW Accelerator. Inventory and capabilities information are related to the HW Accelerators and AAL- LPUs. |  |
| [REQ-AALI-C-MGMT3] | VOID |  |
| [REQ-AALI-C-MGMT4] | The AALI-C-Mgmt interface shall provide fault information for the HW Accelerator and AAL-LPUs. AALI-C-Mgmt interface shall support the autonomous sending of notification when a fault has been raised. AAL-C-Mgmt interface shall also be able to reply to on demand queries of faults (e.g., active faults). |  |
| [REQ-AALI-C-MGMT5] | The AALI-C-Mgmt interface shall provide logs for the HW Accelerator and AAL-LPUs. Logs can be related to faults, debugging info etc. The AAL-C-Mgmt interface shall support operations like starting and stopping the logs collection and support the logs retrieval. |  |
| [REQ-AALI-C-MGMT6] | AALI-C-Mgmt interface shall support providing performance information for the HW Accelerator and AAL-LPUs. The AAL-C- Mgmt interface shall support the autonomous sending of notification with PM info (e.g., periodically or due to an event). AAL-C-Mgmt interface shall also be able to reply to on demand queries for PM data. |  |
| [REQ-AALI-C-MGMT7] | The AALI-C-Mgmt interface shall provide state management services for HW Accelerator and AAL-LPUs. Examples of state management operations are support querying the status and state information, autonomous notification of state changes, restart/reset HW Accelerator, Power Cycle HW Accelerator, Start AAL-LPU and Stop AAL-LPU. |  |
| [REQ-AALI-C-MGMT8] | The AALI-C-Mgmt interface shall provide configuration management services. Examples of operations related to configuration management are the following:   * Configure the number of AAL-LPUs to be supported. * Configure the Profiles to be support on the AAL-LPUs. * Retrieve AAL-LPU capabilities. * Configure performance collection interval. |  |
| [REQ-AALI-C-MGMT9] | The AALI-C-Mgmt interface shall support the installation and upgrade of the HW Accelerator Drivers/FW. |  |

|  |  |  |
| --- | --- | --- |
| [REQ-AALI-C-MGMT10] | Void |  |
| [REQ-AALI-C-MGMT11] | Void |  |
| [REQ-AALI-C-MGMT12] | Void |  |
| [REQ-AALI-C-MGMT13] | Void |  |
| [REQ-AALI-C-MGMT14] | Void |  |
| [REQ-AALI-C-MGMT15] | The AALI-C-Mgmt interface shall provide the ability to report the default values for attributes associated with AAL-Profiles that are supported by the HW Accelerator. |  |
| [REQ-AALI-C-MGMT16] | The AALI-C-Mgmt interface shall report the capabilities associated with AAL-Profiles that are supported by the HW Accelerator. |  |

### Functional requirements for the HAM

The following tables describe functional requirements for the HAM.

**Table 3.1.1-1: Functional requirements for HWA and LPUs lifecycle management**

|  |  |
| --- | --- |
|  | Description |
| [REQ-HAM-LCM 1] | The HAM shall support the capability to expose inventory information and capabilities of the physical and logical partitioning of the hardware and software acceleration resources. |
| [REQ-HAM-LCM 2] | The HAM shall support the capability to discover the capabilities of the HW Accelerator(s). |
| [REQ-HAM-LCM 3] | The HAM shall support the capability to configure the HW Accelerator. See NOTE. |
| [REQ-HAM-LCM 4] | The HAM shall provide the capability to preserve HWA and AAL-LPU identifiers that are provided either by the IMS or the HAM and used in the AALI-C-Mgmt APIs:   * HAM, O-Cloud node, HWA, AAL-LPU planned and unplanned restarts or reboots, * HAM, O-Cloud node, HWA, AAL-LPU SW updates or configuration changes. |
| NOTE: The configuration of the HW Accelerator Manager may include HW Accelerator resource assignment to AAL- LPUs. | |

**Table 3.1.1-2: Functional requirements for faults and logs management**

|  |  |
| --- | --- |
|  | Description |
| [REQ-HAM-FM 1] | The HAM shall support the capability to expose fault information related to the HWA(s) it manages to IMS. |
| [REQ-HAM-FM 2] | The HAM shall support the capability to expose fault information related to the LPU(s) it manages to IMS. |
| [REQ-HAM-FM 3] | The HAM shall support the capability to expose logging information related to the HWA(s) it manages to IMS. |
| [REQ-HAM-FM 4] | The HAM shall support the capability to expose logging information related to the LPU(s) it manages to IMS. |

**Table 3.1.1-3: Functional requirements for performance management**

|  |  |
| --- | --- |
|  | Description |
| [REQ-HAM-PM 1] | The HAM shall support the capability of performance management of HWAs. |
| [REQ-HAM-PM 2] | The HAM shall support the capability to expose performance information related to the HWA it manages (see NOTE). |
| [REQ-HAM-PM 3] | In response to a query, the HAM shall support the capability to provide the information about active PM jobs which match filtering criteria. |
| NOTE: Performance information can be also related to collected performance information of the LPUs that are operating on the HWA. | |

**Table 3.1.1-4: Functional requirements for security considerations**

|  |  |
| --- | --- |
|  | Description |
| [REQ-HAM-SEC 1] | The HAM shall support being certified by the O-Cloud vendor. |

**Table 3.1.1-5: Functional requirements for software management**

|  |  |
| --- | --- |
|  | Description |
| [REQ-HAM-SW 1] | The HAM shall support the capability to update and/or upgrade the software for an HWA on the O-Cloud node. See NOTE. |
| NOTE: 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. | |

**Table 3.1.1-6: Functional requirements for HAM registration to IMS exposure of the AAL-C-Mgmt interface**

|  |  |
| --- | --- |
|  | Description |
| [REQ-HAM-REG 1] | The HAM shall be able to be configured with an IMS service endpoint. |
| [REQ-HAM-REG 2] | The HAM shall be able to be configured with a local HAM service endpoint that the IMS will use to communicate with the HAM. |
| [REQ-HAM-REG 3] | The HAM shall be able to register the HAM service endpoint with the IMS via the AALI- C-Mgmt interface. |

**Table 3.1.1-7: Functional requirements for HAM related to AAL-Profiles**

|  |  |
| --- | --- |
|  | Description |
| [REQ-HAM-PROF 1] | The HAM shall provide the ability to retrieve the default values for attributes associated with AAL-Profiles that are supported by the HW Accelerator. See NOTE. |
| [REQ-HAM-PROF 2] | The HAM shall provide the ability to report the default values for attributes associated with AAL-Profiles that are supported by the HW Accelerator. |
| [REQ-HAM-PROF 3] | The HAM shall provide the ability to report the capabilities associated with AAL-Profiles that are supported by the HW Accelerator. |

NOTE: The mechanisms for the specification and retrieval of the default values for the AAL-Profile are vendor specific behaviour.

## AALI-C App Requirements

**Table 3.2-1: AALI-C-App Requirements**

|  |  |  |
| --- | --- | --- |
| **Requirement Number** | **Description** | **Additional Info** |
| [REQ-AALI-C-APP-1] | During deployment, the DMS shall provide the AAL Application with a list of LPU identifiers that the DMS has assigned to the AAL Application. | DMS-AAL  Interface |
| [REQ-AALI-C-APP-2] | During deployment, the DMS may provide the AAL Application with additional attributes needed by the AAL Implementation for the AAL Implementation to initialize the AAL Implementation for the AAL Application. | DMS-AAL  Interface |
| [REQ-AALI-C-APP-3] | During deployment, the AAL Application shall pass the information provided by the DMS that is needed to initialize the AAL Implementation for the AAL Application. | DMS-AAL  Interface |
| NOTE: The above AAL requirements (REQ-AALI-C-APP-1 to 3) for DMS to AAL interface are out of scope for the current version of the AAL Common API specification. | | |

### AALI-C-APP Transport Abstraction Requirements and Operations

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). AALI-C-App shall support abstraction of these various transport mechanisms between an AAL Application and an AAL Implementation through a set of common transport APIs, constituting a transport abstraction framework.

**Table 3.2.1-1: AALI-C-App Transport Abstraction Framework Requirements**

|  |  |  |
| --- | --- | --- |
| **Requirement Number** | **Description** | **Operation** |
| [REQ-AALI-C-APP- TRANSPORT1] | The Transport Abstraction Framework shall provide a method for the AAL Application to request an AAL Implementation to create buffer pool(s) constituting of buffers(s) of one size or more than *one size.* | createBufferPool (Sec. 5.4.3.1) |
| [REQ-AALI-C-APP- TRANSPORT2] | The Transport Abstraction Framework shall provide a method for the AAL Application to request an AAL Implementation to allocate buffers. | allocBuffer (Sec. 5.4.3.2) |
| [REQ-AALI-APP- TRANSPORT3] | The Transport Abstraction Framework shall support Synchronous (blocking), and Asynchronous (non-blocking) interactions for operations related to sending and receiving buffer(s) by the AAL Application**.** | sendBuffer (Sec. 5.4.3.5)  receiveBuffer (Sec. 5.4.3.6) |
| [REQ-AALI-C-APP- TRANSPORT4] | The Transport Abstraction Framework shall support a method of enabling buffer ownership transfer from the AAL Application to the AAL Implementation. |  |
|  | * When the AAL Application retains ownership of buffer(s), the AAL Implementation shall not free the |  |

|  |  |  |
| --- | --- | --- |
|  | buffer(s) on completion of an operation until buffer ownership is transferred by the AAL Application.   * If the AAL Application does not retain ownership of the buffer(s), the AAL Implementation shall assume ownership and free the buffer(s) on completion of an operation |  |
| [REQ-AALI-C-APP- TRANSPORT5] | The Transport Abstraction Framework shall support method for the AAL Application to receive data in the buffer(s) allocated by the AAL Application or the AAL Implementation. | receiveBuffer (Sec. 5.4.3.6) |
| [REQ-AALI-C-APP- TRANSPORT6] | The Transport Abstraction Framework shall support a method for freeing pre-allocated buffer(s). | receiveBuffer (Sec. 5.4.3.6) |
| [REQ-AALI-C-APP- TRANSPORT7] | The Transport Abstraction Framework shall support a method for destroying pre-created buffer pool(s). | freeBuffer (Sec. 5.4.3.7) |
| NOTE: Synchronous (Blocking) API - blocks until the operation is complete and returns the status of the operation execution (success/failure). Asynchronous (Non-Blocking) API - returns immediately before the operation is complete. The AAL Application invoking API calls in this mode can pre-register for a callback. | | |

## AAL Information Model

### Overview

The AALI-C-Mgmt interface is the management interface that is used by the IMS for administrative operations/actions/events. The HAM terminates the AALI-C-Mgmt interface.

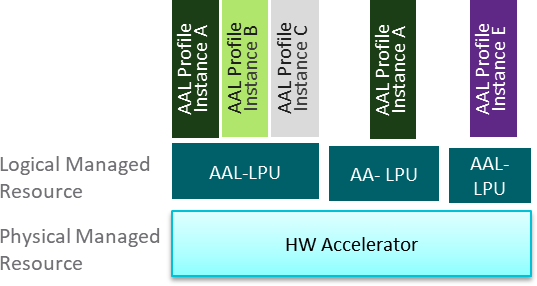
Operations and actions/events via AALI-C-Mgmt are applicable on the following entities:

* HW Accelerator
* AAL Logical Processing Unit (AAL-LPU)
* AAL Profile

The AALI-C-App interface is exposed by the AAL Implementation and is used by the AAL Application to discover, configure, and manage the AAL-Profile-Instance(s) and AAL-Profile-Queue(s) in the AAL-LPU assigned to the AAL Application.

NOTE: An AAL Application may choose to use one or more AAL Profiles as part of its implementation. Operations and actions/events via AALI-C-App are applicable on the following entities:

* AAL Logical Processing Unit (AAL-LPU)
* AAL Profile Instance
* AAL Profile Queue



**Figure 3.3-1: Physical and Logical resources**

NOTE: The definition of these entities are defined in GAnP specification [[7]](#_bookmark5) section 1.3: The cardinality and relationship between these entities are derived from the GAnP specification [[7]](#_bookmark5) section 2.3. The various scenarios supported are also available in the GAnP [[7]](#_bookmark5) section 5.

The Information model defined in [6] will be used to define these managed resources.

NOTE 1: Usage of ‘M', 'O', and 'CM' imply Mandatory, Optional and Conditional Mandatory in the spec

### AALI-C-Mgmt Information Model

#### Imported and associated information entities

##### Imported information entities and local labels

Information entities imported into the model are used for inheritance purposes.

**Table 3.3.2.1.1-1 AALI-C-Mgmt Imported Information Entities and Local Labels**

|  |  |
| --- | --- |
| **Label reference** | **Local label** |
| TS 28.622 [10] Information Object Class, TS 28.620  [11] Top\_ | Top\_ |

#### Associated information entities and local labels

Information entities referenced in the model through an association that is not based on inheritance are listed in the associated entities table.

**Table 3.3.2.1.2-1 AALI-C-Mgmt Associated Information Entities and Local Labels**

|  |  |
| --- | --- |
| Label reference | Local label |
|  |  |

#### Class Diagram

##### Relationships

@startuml

skin rose

skinparam ClassStereotypeFontStyle normal skinparam style strictuml

hide members

Namespace ORAN.AALI.Mgmt {

Class "HwAccel (HW Accelerator)" as hwa <<Information Object Class>> {

}

Class "AalLpu (AAL-LPU)" as lpu <<Information Object Class>> {

}

Class “AalProfile (AAL Profile)” as profile <<dataType>> {

}

Class “Fault” as fault <<dataType>> {

}

Class “AalLpuConfiguration” as lpuConfig <<dataType>> {

}

Class “HW Accelerator Manager” as ham <<Information Object Class>> {

}

Class “AalProfileTemplate” as aalProfileTemplate <<dataType>> {

}

hwa “1” \*-down- “<U+002B>aalProfileTemplates\n1..\*” aalProfileTemplate hwa “1” \*-down- “<U+002B>aalLpus\n0..\*” lpu

hwa “1” -down-> “0..\*” fault: faults

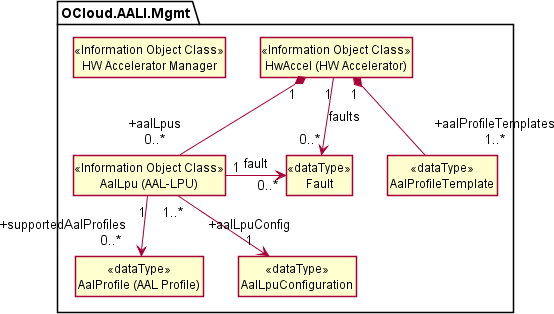
lpu “1” -right-> “0..\*” fault: fault

lpu "1..\*" -down-> "<U+002B>aalLpuConfig\n1" lpuConfig

lpu “1” -down-> "<U+002B>supportedAalProfiles\n\t\t\t0..\*" profile

}

@enduml



**Figure 3.3.2.2.1-1: AALI-C-Mgmt Class Relationships**

Editor’s Note: Open Action: Requires coordinated adjustment in WG6/WG10 specification.

Editor’s Note: The identification of the HAM and the HAM relationship with the IMS and O-Cloud Infrastructure is FFS.

##### Inheritance

@startuml

skin rose

skinparam ClassStereotypeFontStyle normal

skinparam style strictuml hide members

Namespace ORAN.AALI.App {

Class "AalLpu (AAL-LPU)" as lpu <<Information Object Class>> {

}

Class “AalProfileInstance (AAL Profile Instance)” as profilei <<Information Object

Class>> {

}

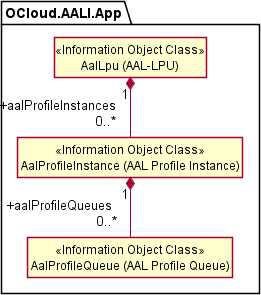
Class “AalProfileQueue (AAL Profile Queue)” as queue <<Information Object Class>> {

}

lpu “1” \*-- “<U+002B>aalProfileInstances\n\t\t\t0..\*” profilei profilei “1” \*-down- “<U+002B>aalProfileQueues\n\t\t\t0..\*” queue

}

@enduml



**Figure 3.3.2.2.2-1: AALI-C-Mgmt Class Inheritance**

#### Class Definitions

##### HwAccel <<InformationObjectClass>>

Definition

The ORAN.AALI.Mgmt.HwAccel class represents the HW Accelerator.

Attributes

**Table 3.3.2.3.1.2-1 Attributes Properties for HWAccel**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| hwAccelId | **Data Type:** string | x-support-qualifier: M |

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
|  | **Description:** UUID identifier for an instance of the HW Accelerator.  This identifier is unique within the O-Cloud.  Data Type pattern and examples are given in [[15]](#_bookmark9) | readOnly: False  x-isInvariant: False x-inventoryNotification True nullable: False  format: uuid |
| operationalState | **Data Type:** string  **Description:** Reflects the HW Accelerator’s operational state as defined in clause 7.1.1 of ITU X.731. | 1. support-qualifier: M   readOnly: True   * 1. sInvariant: False x-inventoryNotification False nullable: False   enum:   * + - DISABLED     - ENABLED   default: DISABLED |
| operationalConditi ons | **Data Type:** array  **Description:** List of conditions that support the operational state of the entity. | 1. support-qualifier: M   readOnly: True   * 1. sInvariant: False x-inventoryNotification False nullable: False   enum:   * + - UNKNOWN     - STARTED   default: UNKNOWN |
| serialNum | **Data Type:** string  **Description:** Serial number of this entity. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification True nullable: False |
| vendorName | **Data Type:** string  **Description:** Name of the Vendor/Provider of the entity. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification True nullable: False |
| model | **Data Type:** string  **Description:** Information about the model of the entity. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification True nullable: False |
| hwVersion | **Data Type:** string  **Description:** Version or generation of the entity. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification True nullable: False  example: ‘1.9.0' |
| maxNumAalLpus | **Data Type:** integer  **Description:** This attribute may be included if a HW Accelerator has an upper limit on AAL-LPUs it can support. Actual number of AAL-LPUs available at runtime may be lower based on resource  usage, AAL-LPU SW used, and | x-support-qualifier: O  readOnly: True  x-isInvariant: False x-inventoryNotification True nullable: True  format: integer8  minimum: 0 |

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
|  | AAL-Profile SW and types instantiated on the AAL-LPU(s). |  |
| aalProfileTemplate s | **Data Type:** array  **Description:** The list of AAL- Profile templates supported by the Accelerator. The AAL-Profile templates can be used to configure LPUs within the Accelerator. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  x-isOrdered: False  title: AAL-Profile  Templates  minItems: 1  uniqueItems: True items:  AalProfileT  emplate |
| faults | **Data Type:** array  **Description:** The list of current fault conditions. | x-support-qualifier: M  readOnly: True  x-isInvariant: False x-inventoryNotification False nullable: False  x-isOrdered: False  title: faults  maxItems: \*  minItems: 0  uniqueItems: True  items: Fault |
| lpuType | **Data Type:** string  **Description:** The method to apply the configuration settings for the aalLpus associated with a HW Accelerator. | 1. support-qualifier: M    1. sInvariant: False x-inventoryNotification True nullable: False   enum:   * + - Single     - Mixed   default: Mixed |
| numAalLpusConfi gured | **Data Type:** integer8  **Description:** The number of aalLpus currently configured. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  format: integer8  minimum: 0 |
| aalLpus | **Data Type:** array  **Description:** The list of AAL- LPUs configured for this HW Accelerator.  The number of aalLpu instances is determined by the value of the numAllLpusConfigured attribute. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  x-isOrdered: False  title: AAL LPUs  minItems: 0  uniqueItems: True  items: AalLpu |
| imageVersion | **Data Type:** string  **Description:** The version for the image applied to the HW Accelerator. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  example: ‘1.9.0' |
| imageLocation | **Data Type:** string  **Description: URI**[**[16]**](#_bookmark10)Path to the software/firmware image. | x-support-qualifier: M  x-isInvariant: False  x-inventoryNotification True nullable: False  format: uri |
| extensions | **Data Type:** array  **Description:** The list of | x-support-qualifier: M  x-isInvariant: False |

|  |  |  |
| --- | --- | --- |
|  | unspecified (not standardized) properties for the entity which are tailored by the vendor. | x-inventoryNotification True nullable: False  x-isOrdered: False  title: extensions  minItems: 0  uniqueItems: True items:  AttributeVa  luePair |

Attribute constraints None

State diagram

Introduction

The HWAccel maintains an operational state of the entity as defined in clause 7.1.1 of ITU X.731 [14]. How the HW Accel determines its operational state is implementation specific.

##### AalLpu <<InformationObjectClass>>

Definition

The ORAN.AALI.Mgmt.AalLpu class represents the AAL-LPU.

Attributes

**Table 3.3.2.3.2.2-1 Attributes Properties for AalLpu**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| aalLpuId | **Data Type:** string  **Description:** UUID identifier for an instance of the AAL-LPU.  This identifier is unique within the O-Cloud.  Data Type pattern and examples are given in [[15]](#_bookmark9) | x-support-qualifier: M readOnly: False  x-isInvariant: False x-inventoryNotification True nullable: False  format: uuid |
| operationalState | **Data Type:** string  **Description:** Reflects the AAL- LPU’s operational state as defined in clause 7.1.1 of ITU X.731. | 1. support-qualifier: M   readOnly: True   * 1. sInvariant: False x-inventoryNotification False nullable: False   enum:   * + - DISABLED     - ENABLED   default: DISABLED |
| operationalConditi ons | **Data Type:** array  **Description:** List of conditions that support the operational state of the entity. | 1. support-qualifier: M   readOnly: True   * 1. sInvariant: False x-inventoryNotification False nullable: False   enum:   * + - UNKNOWN     - STARTED     - STARTINGUP     - SHUTTINGDOWN |

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
|  |  | - STOPPED  default: UNKNOWN |
| faults | **Data Type:** array  **Description:** The list of current fault conditions. | x-support-qualifier: M  readOnly: True  x-isInvariant: False  x-inventoryNotification False nullable: False  x-isOrdered: False  title: faults  maxItems: \*  minItems: 0  uniqueItems: True  items: Fault |
| supportedAalProfil es | **Data Type:** array  **Description:** The list of supported AAL Profiles. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  x-isOrdered: False  title: AAL  Profiles  minItems: 0  uniqueItems: True  items: AalProfile |
| imageVersion | **Data Type:** string  **Description:** The version or generation for the image applied to the entity.  If an AAL-LPU image is the same as the Accelerator HW's image, then the HW Accelerator's imageVersion attribute and this attribute would contain the same value. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  example: ‘1.9.0' |
| imageLocation | **Data Type:** string  **Description: URI** [**[16]**](#_bookmark10)Path to the software/firmware image.  If an AAL-LPU image is the same as the Accelerator HW's image, then the HW Accelerator's imageLocation attribute and this attribute would contain the same value. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  format: uri |
| aalLpuConfig | **Data Type:** AalLpuConfig  **Description:** The common configuration attributes that are common to one or more AAL- LPUs including this AAL-LPU. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False |

Attribute constraints None

State diagram

Introduction

The AAL-LPU maintains an operational state of the entity as defined in clause 7.1.1 of ITU X.731 [14]. How the AAL- LPU determines its operational state is implementation specific.

##### AalProfile <<dataType>>

Definition

The ORAN.AALI.Mgmt.AalProfile class represents the AAL-Profile.

Attributes

**Table 3.3.2.3.3.2-1 Attributes Properties for AalProfile**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| name | **Data Type:** string  **Description:** Name identifier of the AAL Profile. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification True nullable: False  format: string |
| version | **Data Type:** string  **Description:** The version or AAL Profile. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  example: ‘1.9.0' |
| imageVersion | **Data Type:** string  **Description:** The version or generation for the image of the entity.  If an AAL-Profile image is the same as the AAL-LPU's image then the AAL-LPU's imageVersion attribute and this attribute would contain the same value. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  example: ‘1.9.0' |
| imageLocation | **Data Type:** string  **Description:** Path to the software/firmware image.  If an AAL-Profile image is the same as the AAL-LPU's image then the AAL-LPU's imageLocation attribute and this attribute would contain the same value. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  format: uri |
| attributes | **Data Type:** array  **Description:** The list of AAL- Profile attributes represented as key/value pairs.  The values for entries in this list that are duplicates of entries in the AALProfileTemplate's attributes | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  x-isOrdered: False  title: Attributes  minItems: 0  uniqueItems: True items:  AttributeVa |

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
|  | attribute override the value provided by the AALProfileTemplate attributes attribute's entry. | luePair |
| extensions | **Data Type:** array  **Description:** The list of AAL- Profile unspecified (not standardized) attributes represented as key/value pairs. The structure and format of the value attribute is specified as part of the definition of the key.  The values for entries in this list that are duplicates of entries in the AALProfileTemplate's extensions attribute override the value provided by the AALProfileTemplate extensions attribute's entry. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  x-isOrdered: False  title: Extension  Attributes  minItems: 0  uniqueItems: True items:  AttributeVa luePair |

Attribute constraints None

State diagram None

##### Fault <<dataType>>

Definition

The ORAN.AALI.Mgmt.Fault class represents a current Fault condition associated with an entity.

Attributes

**Table 3.3.2.3.4.2-1 Attributes Properties for Fault**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| faultId | **Data Type:** string  **Description:** The identifier for the fault condition. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification False  nullable: False |
| detectedTime | **Data Type:** string  **Description:** The date and time the fault condition was last detected. | x-support-qualifier: M  readOnly: True  x-isInvariant: False x-inventoryNotification False nullable: False  format: date-time |

Attribute constraints None

State diagram None

##### AalLpuConfig <<dataType>>

3.3.2.3.3.1 Definition

The ORAN.AALI.Mgmt.AalLpuConfig class represents the AAL-LPU configuration settings to be applied to an instance of an AAL-LPU.

The AAL-LPU's HW Accelerator determines how the AAL-LPU configuration settings are applied using the HW Accelerators lpuType and numAalLpusConfigured attributes. If the lpuType attribute's value is "mixed", then each AAL-LPU for the HW Accelerator has a different instance of the AalLpuConfig entity. If the lpuType attribute's value is "single" then all AAL-LPUs for the HW Accelerator share the same instance of the AalLpuConfig entity.

Attributes

**Table 3.3.2.3.5.2-1 Attributes Properties for AalLpuConfig**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| memory | **Data Type:** integer  **Description:** Memory allocated to this entity. | x-support-qualifier: O  x-isInvariant: False x-inventoryNotification True nullable: True  format: integer64  x-units: bytes |
| multiprocessorNum | **Data Type:** integer  **Description:** Number of multi- processors allocated to this entity. | x-support-qualifier: O  x-isInvariant: False x-inventoryNotification True nullable: True  format: integer8  minimum: 1 |
| computeSlicesNum | **Data Type:** integer  **Description:** Number of compute- slices allocated to this entity. | x-support-qualifier: O  x-isInvariant: False x-inventoryNotification True nullable: True  format: integer8  minimum: 1 |
| extensions | **Data Type:** array  **Description:** The list of unspecified (not standardized) properties defined by vendors that used for the configuration of the AAL-LPU.of the AAL-LPU. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification True nullable: False  x-isOrdered: False  title: Extensions  minItems: 0  uniqueItems: True items:  AttributeVa  luePair |

Attribute constraints None

State diagram None

##### AalProfileTemplate <<dataType>>

Definition

The ORAN.AALI.Mgmt.AalProfileTemplate class represents the version of the AAL-Profile template for capabilities and default configurations reported by the HW Accelerator.

Attributes

**Table 3.3.2.3.6.2-1 Attributes Properties for AalProfileTemplate**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| name | **Data Type:** string  **Description:** Name identifier of the AAL Profile template. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification True nullable: False  format: string |
| version | **Data Type:** string  **Description:** The version or AAL Profile Template | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  example: ‘1.9.0' |
| imageVersion | **Data Type:** string  **Description:** The version or generation for the image of the entity.  If an AAL-Profile image is the same as the AAL-LPU's image then the AAL- LPU's imageVersion attribute and this attribute would contain the same value. | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  example: ‘1.9.0' |
| attributes | **Data Type:** array  **Description:** The list of AAL-Profile attributes represented as key/value pairs.  Values for entries in this attribute would be considered the initial or default configuration setting for entries with the associated key. | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  x-isOrdered: False  title: Attributes  minItems: 0  uniqueItems: True items:  AttributeValuePair |
| extensions | **Data Type:** array  **Description:** The list of AAL-Profile unspecified (not standardized) capabilities and attributes represented as key/value pairs. The structure and format of the value attribute is specified as part of the definition of the key.  Values for attribute entries in this attribute would be considered the initial or default | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  x-isOrdered: False  title: Extension  Capabilities and Attributes  minItems: 0  uniqueItems: True items:  AttributeValuePair |

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
|  | configuration setting for entries with the associated key. |  |
| capabilities | **Data Type:** array  **Description:** The list of AAL-Profile specified capabilities represented as key/value pairs. The structure and format of the value attribute is specified as part of the definition of the key in the corresponding AAL Profile specification. | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  x-isOrdered: False title  Capabilities  minItems: 0  uniqueItems: True items:  AttributeValuePair |

Attribute constraints None

State diagram None

##### AalHam <<InformationObjectClass>>

Definition

The ORAN.AALI.Mgmt.AalHam class represents the HAM acceleration management function.

Attributes

**Table 3.3.2.3.7.2-1 Attributes Properties for AalHam**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
|  |  |  |
| operationalState | **Data Type:** string  **Description:** Reflects the HAM’s operational state as defined in clause 7.1.1 of ITU X.731. | 1. support-qualifier: M   readOnly: True   * 1. sInvariant: False x-inventoryNotification False x-stateChangeNotification: True nullable: False   enum:   * + - DISABLED     - ENABLED   default: DISABLED |
| imsRegistrationSta te | **Data Type:** string  **Description:** Reflects the registration state of HAM toward the IMS. | 1. support-qualifier: M   readOnly: True   * 1. sInvariant: False x-inventoryNotification False x-stateChangeNotification: True nullable: False   enum:   * + - NOTREGISTERED     - REGISTERED   default: |

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
|  |  | NOTREGISTER ED |
| localServiceEndpoi nt | **Data Type:** string  **Description:** The service endpoint that the HAM exposes to the IMS that permits the IMS to send requests to the HAM and receive notifications from the HAM. | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  format: uri |
| registrationService Endpoint | **Data Type:** string  **Description:** The service endpoint that the HAM uses register with the IMS in order for the IMS to connect to and manage the HAM. | x-support-qualifier: M  x-isInvariant: True x-inventoryNotification True nullable: False  format: uri |

Attributes Constraints

**Table 3.3.2.3.7.3 -1 Attributes Constraints for AalHam**

|  |  |
| --- | --- |
| **Name** | **Definition** |
| imsRegistrationState | Condition:  Modification of the localServiceEndpoint or registrationServiceEndpoint will cause the HAM to invoke the de- registration transition. |

State diagram

Introduction

The HAM maintains an operational state of the HAM as defined in clause 7.1.1 of ITU X.731 [14]. How the HAM determines its operational state is implementation specific.

The HAM maintains an IMS Registration state that reflects the HAMs current registration with an IMS.

imsRegistrationState diagram

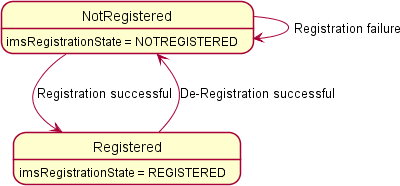
@startuml skin rose

skinparam ClassStereotypeFontStyle normal skinparam style strictuml

hide members

NotRegistered: imsRegistrationState = NOTREGISTERED NotRegistered--> NotRegistered: Registration failure NotRegistered--> Registered: Registration successful Registered : imsRegistrationState = REGISTERED Registered --> NotRegistered: De-Registration successful

@enduml



**Figure 3.3.3.2.1-1:State Diagram for AalHam Registration with IMS**

.

### AALI-C-App Information Model

#### Imported and associated information entities

##### Imported information entities and local labels

Information entities imported into the model are used for inheritance purposes.

**Table 3.3.3.1.1-1 AALI-C-App Imported Information Entities and Local Labels**

|  |  |
| --- | --- |
| **Label reference** | **Local label** |
| TS 28.622 [10] Information Object Class, TS 28.620  [11] Top\_ | Top\_ |

##### Associated information entities and local labels

None

#### Class Diagram

##### Relationships

@startuml

skin rose

skinparam ClassStereotypeFontStyle normal skinparam style strictuml

hide members

Namespace ORAN.AALI.App {

Class "AalLpu (AAL-LPU)" as lpu <<Information Object Class>> {

}

Class “AalProfileInstance (AAL Profile Instance)” as profilei <<Information Object Class>> {

}

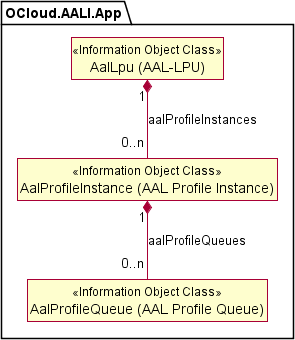
Class “AalProfileQueue (AAL Profile Queue)” as queue <<Information Object Class>> {

}

lpu “1” \*-- “0..n” profilei: aalProfileInstances profilei “1” \*-down- “0..n” queue: aalProfileQueues

}

@enduml



**Figure 3.3.3.2.1-1: AALI-C-App Class Relationships**

##### Inheritance

@startuml skin rose

skinparam ClassStereotypeFontStyle normal skinparam style strictuml

hide members

abstract class ”Top\_” as top <<Information Object Class>>

class “AalLpu (AAL-LPU)” as lpu <<Information Object Class>>

class “AalProfileInstance (AAL Profile Instance)” as profilei <<Information Object

Class>>

class “AalProfileQueue (AAL Profile Queue)” as queue <<Information Object Class>>

note right of top defined in

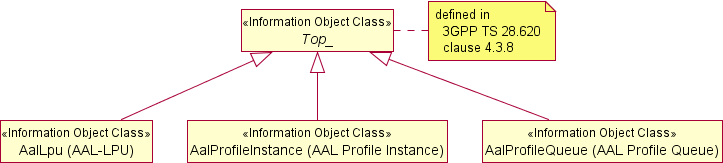
3GPP TS 28.620

clause 4.3.8 end note

top <|-d- lpu

top <|-d- profilei top <|-d- queue

@enduml



**Figure 3.3.3.2.2-1: AALI-C-App Class Inheritance**

#### Class Definitions

##### AalLpu <<InformationObjectClass>>

Definition

The ORAN.AALI.App.AalLpu class represents the runtime instance of an AAL-LPU entity.

Attributes

**Table 3.3.3.3.1.2-1 Attributes Properties for AalLpu**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| aalLpuHandle | **Data Type:** string  **Description:** Identifier that represents the handle (e.g., pointer) that the AAL Application uses to access the AAL-LPU exposed by the AAL Implementation  This identifier is unique between the AAL Implementation and the AAL Application.  The identifier is assigned by the AAL Implementation. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification False nullable: False  format: binary |
| aalProfileInstances | **Data Type:** array  **Description:** The list of AAL- Profile instances created by the AAL Application for this AAL- LPU. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification False nullable: False  x-isOrdered: False  minItems: 0  uniqueItems: True items:  AalProfileI  nstance |

3.3.3.3.2.3 Attribute constraints None

3.3.3.3.1.4 State diagram

Section 5.3.1 of this present document describes the state diagram for this entity.

##### AalProfileInstance <<InformationObjectClass>>

Definition

The ORAN.AALI.App. AalProfileInstance class represents the runtime instance of an AAL-Profile entity.

Attributes

**Table 3.3.3.3.2.2-1 Attributes Properties for AalProfileInstance**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| aalProfileInstance Handle | **Data Type:** string  **Description:** Identifier that represents the handle (e.g., pointer) that the AAL Application uses to access the AAL-Profile instance exposed by the AAL Implementation  This identifier is unique between the AAL Implementation and the AAL Application.  The identifier is assigned by the AAL Implementation. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification False nullable: False  format: binary |
| aalProfileName | **Data Type:** string  **Description:** Name identifier of the AAL Profile. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification False nullable: False  format: string |
| aalProfileVersion | **Data Type:** string  **Description:** The version of the AAL Profile. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification False nullable: False  example: ‘1.9.0' |
| aalProfileQueues | **Data Type:** array  **Description:** The list of AAL- Profile Queues created by the AAL Application for this AAL-Profile instance. | x-support-qualifier: M  x-isInvariant: False x-inventoryNotification False nullable: False  x-isOrdered: False  minItems: 0  uniqueItems: True items:  AalProfileQ ueue |

Attribute constraints None

State diagram

Section 5.3.1 of this present document describes the state diagram for this entity.

##### AalProfileQueue <<InformationObjectClass>>

Definition

The ORAN.AALI.App.AalProfileQueue class represents the runtime instance of an AAL-Profile-Queue entity.

Attributes

**Table 3.3.3.3.3.2-1 Attributes Properties for AalProfileQueue**

|  |  |  |
| --- | --- | --- |
| **Attribute name** | **Data Type/Description** | **Properties** |
| aalProfileQueueHa ndle | **Data Type:** string  **Description:** Identifier that represents the handle (e.g., pointer) that the AAL Application uses to access the AAL-Profile-Queue instance exposed by the AAL Implementation  This identifier is unique between the AAL Implementation and the AAL Application.  The identifier is assigned by the AAL Implementation. | x-support-qualifier: M  readOnly: True  x-isInvariant: True x-inventoryNotification False nullable: False  format: binary |

Attribute constraints None

State diagram

Section 5.3.1 of this present document describes the state diagram for this entity.

### Naming convention and managed resource tags

#### API managed resources tags

For the following managed resources, the associated API managed resource tags will be used:

**For the managed resources**:

* HW Accelerator - **HW-Accel**
* AAL LPU - **AAL-LPU**
* AAL Profile Instance - **AAL-Profile-Instance**
* AAL Profile Queue - **AAL-Profile-Queue Profile Naming convention**:
* FEC Profile – **AAL-FEC**
* L1 DL High Phy Profile – **AAL-DOWNLINK-High-PHY**
* L1 UL High Phy Profile – **AAL-UPLINK-High-PHY**

#### API Naming Conventions

**Operations and parameters naming convention**:

* lowerCamelCase for operation names
* lower\_snake\_case for parameters **Group of attributes are notified with a > sign**. Example:

A grouped list of variables a, b and c under one parameter i.e. parameter\_name is shown as follows:

|  |  |
| --- | --- |
| parameter\_name | struct |
| >a |  |
| >b |  |
| >c |  |

#### Model Conventions

The conventions for the development of the Information Model are derived from the following sources:

3GPP TS 32.156 [1] provides the basis of the Class definitions for modelling elements that also includes the Stereotypes to be used when defining Classes. 3GPP TS 32.156 also includes the descriptions in clause 5.2.1.1 of the 3GPP standard Class attribute properties.

OAS 3.0 Specification [[13]](#_bookmark8) provides the basis for the Information Model data types. The Swagger documentation provides additional descriptions for the Class Attribute properties.

# Chapter 4 AALI-C-Mgmt

## AALI-C-Mgmt operations and events

The below table lists the operations and events as part of the AALI-C-Mgmt interface.

Note: The present document version does not contain details of all the operations listed in the table below. The operations marked TBD need further evaluation.

**Table 4.1-1: AALI-C-Mgmt operations and events**

|  |  |  |
| --- | --- | --- |
| **Category** | **Operations/Events** | **Objects** |
| **Capabilities discovery and Configuration Services** | | |
| **Get Configuration Services** | Get configuration   * getAalHwAccelStatus * getAalLpuStatus * getAalAccelInfo (inventory and capabilities) * getPmCounters | HW-Accel AAL-LPU |
| **Set Configuration Services** | Set configuration (add/modify/delete) configuration (initial and inline updates): | HW-Accel AAL-LPU |

|  |  |  |
| --- | --- | --- |
|  | * setAalAccelConfig * setPmCollectionInterval (granularity/reporting etc.) (TBD) |  |
|  | Set identification of the managed resource.   * setAalAccelIdentifier | HW-Accel |
| **Inventory Notification** | Inventory notification event   * aalInventoryNotification | HW-Accel AAL-LPU |
| **Infrastructure Monitoring Services** | | |
| **Fault Service Operations** | * getAalHwAccelFaults * getAalLpuFaults | HW-Accel  AAL-LPU |
|  | * manageLogs (Start/Stop/Get logs (Fault, debug)) | HW-Accel  AAL-LPU |
| **Fault Subscription** | * createAalFaultSubscription * deleteAalFaultSubscription * getAalFaultSubscription | HW-Accel  AAL-LPU |
| **Fault Notification** | * aalFaultNotification | HW-Accel  AAL-LPU |
| **State Change Notification** | * State Change notification (Autonomous) | HW-Accel  AAL-LPU |
| **PM Service Operations** | * PM Subscription | HW-Accel  AAL-LPU |
|  | * Reporting of PM (file, streaming etc) | HW-Accel  AAL-LPU |
| **Operational Commands** | | |
| **Operational commands** | * restartHwAccel (TBD) * resetHwAccel (TBD) | HW-Accel |
|  | * powerCycleHwAccel (TBD) | HW-Accel |
|  | * startAalLpu | AAL-LPU |
|  | * stopAalLpu | AAL-LPU |
|  | * initiateTest (TBD) | HW-Accel  AAL-LPU |
| **HAM Operations** | | |
| **IMS**  **Registration** | * aalHamRegistrationNotification | HAM |

## AALI-C-Mgmt use case workflows

### HW Accelerator installation/bootstrap use case workflow

This use case describes the procedure with a focus on the interaction of IMS with the HAM when installing and optionally configuring the HW Accelerator resource and configuring the AAL-LPUs resources. This procedure would be applicable during the initial deployment of an O-Cloud and afterward to reconfigure or add new resources.

This AAL UC flow can be executed during the O-Cloud Registration and Initialization Use Case, O-Cloud Inventory Update Use Case and the Hardware Infrastructure Scaling of O-Cloud Post Deployment Use Case [ Ref[[5]](#_bookmark3): 3.1.3 “O- Cloud Registration and Initialization Use Case”, Configure AAL-LPU “3.1.4 : O-Cloud Inventory Update Use Case (DMS Example)” and “3.1.5: Hardware Infrastructure Scaling of O-Cloud Post Deployment”].

**Table 4.2.1-1: HW Accelerator installation/bootstrap workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | Install the HW Accelerator and update the IMS that the HW Accelerator is available. |  |
| Actors and Roles | Cloud Installer, SMO, FOCOM, IMS, HAM  SW Repo: Software repository |  |
| Assumptions | This UC can be applicable during the O-Cloud Registration and Initialization or during Hardware Infrastructure Scaling of O-Cloud Post Deployment. |  |
| Pre-conditions | SMO/FOCOM is active and running normally O-Cloud is available  IMS is operational  HAM is operational and registered with the IMS SMO and IMS connectivity is established  Accelerator SW Repository is accessible to HAM and populated with needed files. |  |
| Begins when | SMO sends request to install the HW Accelerator |  |
| Step 1 (M) | The IMS updates its inventory with the O-Cloud Resource information for the HW Accelerator discovered by the HAM.  The IMS obtains, via the AALI-C-Mgmt interface, the managed resource information for the HW Accelerator through reception of an inventory notification event for additions of HW Accelerator(s). |  |
| Step 2-3 (O) | If the IMS needs to assign the identifier of the HW Accelerator, the IMS sets the HW Accelerator’s identifier by sending the setAalAccelIdentifier request to the HAM. |  |
| Steps 4-7(O) | If the HW Accelerator status information is exchanged, additionally optionally the IMS can get the AAL-LPU status(es). |  |

|  |  |  |
| --- | --- | --- |
| Steps 8-9 (O) | Optionally the IMS can get the HW Accelerator configuration |  |
| Step 10-11 (O) | Optionally the IMS can configure the HW Accelerator (via the HAM): If the IMS (or SMO) determines that different image(s) should be used it may send a setAalAccelConfig request to the HAM, if the HAM determines new binaries, SW or configuration artifacts are needed from a SW repo, they are fetched, verified, and installed.  SMO is informed about the status if the operation |  |
| Ends when | HW Accelerator inventory has been received by the IMS and IMS has optionally configured the HW Accelerator. |  |
| Exceptions | None identified |  |
| Post Conditions | The HW Accelerator has been deployed on the O-Cloud. |  |
| Traceability | To be added later |  |

#### Sequence Diagram

@startuml

skinparam sequenceArrowThickness 1

skinparam ParticipantPadding 2

skinparam BoxPadding 5 Autonumber

Box "Personnel" #lightblue

Actor Installer as "Cloud Installer" End box

Box "Service Management & Orchestration Framework" #gold Participant "FOCOM" as FCC

End box

Box "O-Cloud

"#lightseagreen

Participant "IMS" as IMS Participant AccelMgr as "HAM"

End box

Box "SW Repo" #grey

Participant "SW Repo" as Repo End Box

Note over Installer, IMS PRECONDITIONS:

SMO/FOCOM is active and running normally O-Cloud is available

IMS is operational

HAM is operational and registered with the IMS SMO and IMS connectivity is established

IMS and HAM connectivity is established HW Accel(s) installed

HAM connection to HW Accel(s) is established End Note

AccelMgr --> IMS : <<AALI-C-Mgmt>> aalInventoryNotification (inc SW/FW/Config - same params as getAalAccelInfo)

Note over FCC, IMS

IMS informs SMO of the HW Accelerator Inventory

SMO Updates inventory with the HW Accelerator inventory information

End Note

Opt

Note over IMS, AccelMgr

If IMS needs to set the identifier of the HW Accel using the HW Accelerator vendor\_name, model and serial\_number input parameters.

End Note

IMS -> AccelMgr : <<AALI-C-Mgmt>> setAalAccelIdentifier request

AccelMgr -> IMS : <<AALI-C-Mgmt>> setAalAccelIdentifier response (success) End Opt

Opt

IMS -> AccelMgr : <<AALI-C-Mgmt>> getAalHwAccelStatus request

AccelMgr -> IMS : <<AALI-C-Mgmt>> getAalHwAccelStatus response (success) opt

IMS -> AccelMgr : <<AALI-C-Mgmt>> getAalLpuStatus request

AccelMgr -> IMS : <<AALI-C-Mgmt>> getAalLpuStatus response (success) End opt

End opt

Opt

IMS -> AccelMgr : <<AALI-C-Mgmt>> getAalAccelInfo request

AccelMgr -> IMS : <<AALI-C-Mgmt>> getAalAccelInfo response (success) End opt

Note over FCC, IMS

SMO optionally instructs IMS to configure HW Accelerator boot image or other config/SW/config artifact

End Note

Opt

Note over IMS, AccelMgr

If boot image or other config/SW/config artifact is needed or new driver is needed (in order to match an intended configuration) then it is set using setAalAccelConfig

API

End Note

IMS -> AccelMgr : <<AALI-C-Mgmt>> setAalAccelConfig request Note over AccelMgr, Repo

Download & apply needed binaries, images, FW, etc.

End note

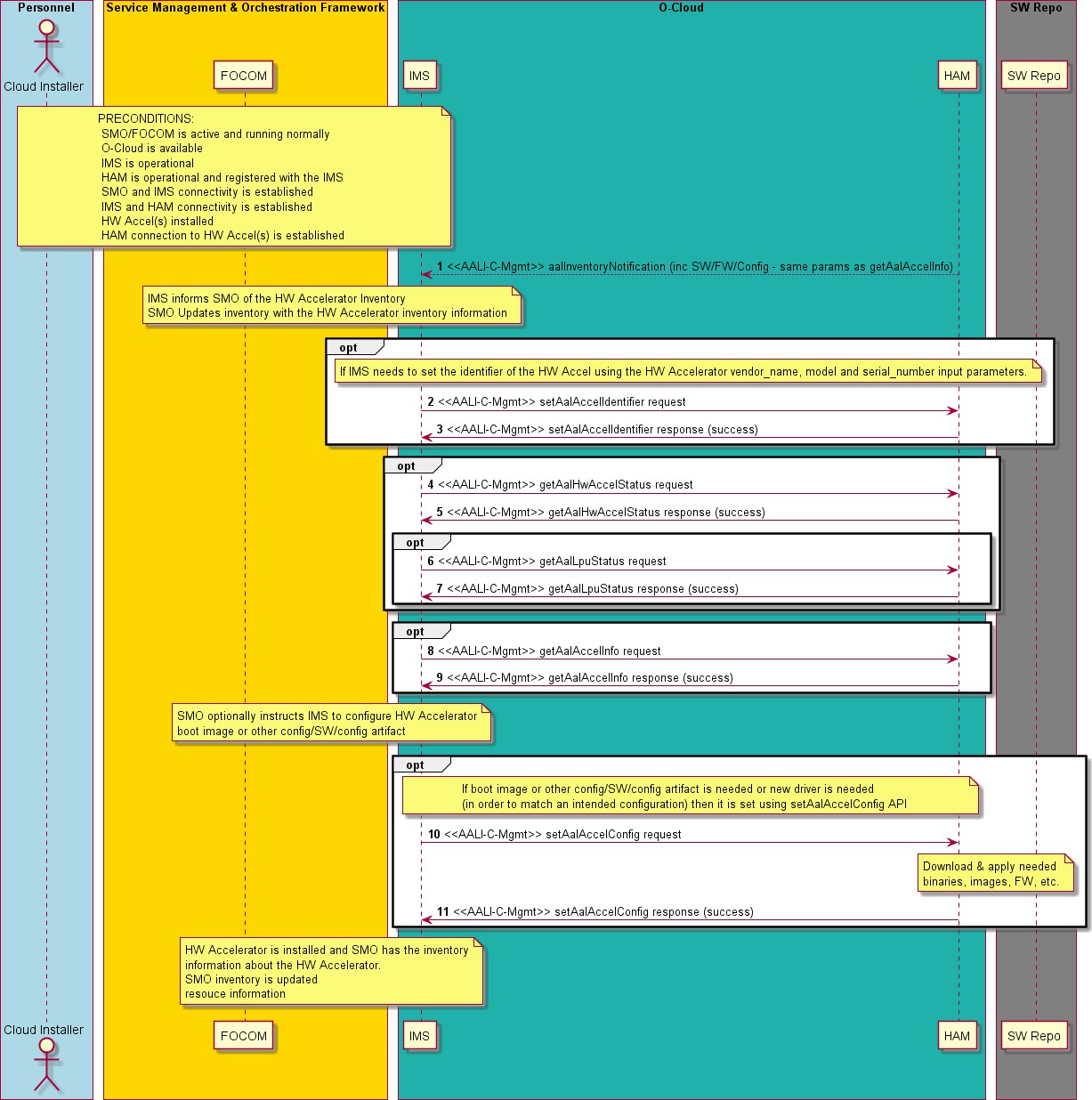
AccelMgr -> IMS : <<AALI-C-Mgmt>> setAalAccelConfig response (success) End Opt

Note over FCC, IMS

HW Accelerator is installed and SMO has the inventory information about the HW Accelerator.

SMO inventory is updated resouce information

End Note @enduml



### end-2end Life Cycle Management use case workflow (Informational)

NOTE: The e2e flow depicted in this section are known to be incomplete and are being provided as informational guidance to implementors on how the APIs documented in the remainder of the specification are to be used. These exchanges will be made normative in future versions of the specification.

This use case describes the end-to-end procedure with a focus on the interaction between the IMS and the HAM via the AALI-C-Mgmt interface for discovering the capabilities of the HW Accelerator, configuring the AAL-LPUs, instantiating an AAL Application and optionally updating/reconfiguring the managed resources based on the needs of the AAL Application. The flow also shows the interaction with the AAL Application via the AALI-C-App to create an AAL-Profile-Instance.

This AAL UC flow can be executed during the O-Cloud Registration and Initialization Use Case, O-Cloud Inventory Update Use Case and the Hardware Infrastructure Scaling of O-Cloud Post Deployment Use Case [ Ref[[5]](#_bookmark3): 3.1.3 “O- Cloud Registration and Initialization Use Case”, Configure AAL-LPU “3.1.4 : O-Cloud Inventory Update Use Case

(DMS Example)” and “3.1.5: Hardware Infrastructure Scaling of O-Cloud Post Deployment”]. This use case is also related to the “3.2.1 Instantiate Network Function on O-Cloud”.

**Table 4.2.2-1: end-2-end Life Cycle Management workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | Discover the capabilities of the HW Accelerator, update HW Accelerator and configure AAL-LPU(s) based on the AAL Application needs. |  |
| Actors and Roles | SMO, FOCOM, NFO, IMS, DMS, HAM |  |
| Assumptions | This UC can be applicable for initial or subsequent deployment of the O- Cloud, (re)instantiation of AAL Application and adding/deleting managed resources or AAL-Profile-Instances. |  |
| Pre-conditions | SMO/FOCOM/NFO is active and running normally O-Cloud is available  IMS/DMS is operational  HAM is operational and registered with the IMS  HW Accelerator is bootstrapped/installed ([4.2.1 HW Accelerator](#_bookmark40) [installation/bootstrap use case workflow](#_bookmark40)).  SMO and IMS/DMS communication is established IMS and HAM communication is established  SMO Catalogue, AAL Application Manifest is available SW Repo available  SMO inventory DB available |  |
| Begins when | SMO sends request to query the resource(s) |  |
| Step 1-5 (M) | SMO sends a query to retrieve the available O-Cloud resources Info.  The IMS queries the HAM for the HW Accelerator managed resourc(s) using the getAalAccelInfo via AALI-C-Mgmt interface. This includes hardware information and the HW accelerator capabilities (example: Vendor, Version, BIOS, AAL-LPU, and other capabilities supported by the HW Accelerator).  SMO inventory is updated based on the response. |  |
| Step 6-10 (M) | Based on the planned AAL Application manifest, the SMO/IMS configures the HW Accelerator (e.g.: AAL-LPU capabilities, Profile images etc) using the setAalAccelConfig via the AALI-C-Mgmt interface.  The IMS receives inventory notifications for managed resources that have been added, modified or removed.  DMS inventory is updated.  SMO inventory is updated based on the response. |  |
| Step 11-20 (M) | SMO gets the deployment artifacts for AAL Application (e.g., Manifest/ Helm Charts) that includes requested O-Cloud Resource and profiles. SMO |  |

|  |  |  |
| --- | --- | --- |
|  | then sends request to DMS for AAL Application instantiation with a manifest file.  Using the setAalAccelConfig via the AALI-C-Mgmt interface the HW Accelerator and AAL-LPU configuration may have to be updated as needed for each of the items in the manifest.  The IMS receives receive inventory notifications for managed resources that have been added, modified or removed.  The AAL Application is instantiated, and the DMS/SMO inventory updated. [[5].](#_bookmark3) |  |
| Step 21-22 (M) | The SMO configures the cell details, AAL Profile versions etc to the AAL Application via O1 interface |  |
| Step 23 (M) | Initialize AALI-C-App interface between the AAL Application and the AAL |  |
| Step 24-25 (M) | The AAL Application queries through the AALI-C-App interface the AAL implementation to get the capabilities of the AAL-LPU assigned using the getAalLpuInfo via the AALI-C-App interface.  In response, the AAL Application gets the aal-lpu-handle and the AAL Profile(s) supported. |  |
| Step 26-33 (M) | The AAL Application creates one or many AAL-Profile-Instance(s) via the AALI-C-App using setAalProfileInstance and starts the AAL-Profile- Instance. The AAL Application also sets the AAL-Profile-Queue configuration and starts the queue. |  |
| Step 34-39 (M) | If the AAL-Profile-Instance needs to be updated or is not needed anymore, the AAL Application via the AALI-C-App stops the AAL-Profile-Queue and AAL-Profile-Instance and deletes the AAL-Profile-Instance. |  |
| Ends when | The HW Accelerator and AAL-LPUs have been configured, the AAL Application is instantiated and the AAL Profile instances are managed. |  |
| Exceptions | None identified |  |
| Post Conditions | AAL Application is instantiated, and the AAL-Profile-Instances are managed. |  |
| Traceability | To be added later |  |

#### Sequence Diagram

@startuml

skinparam sequenceArrowThickness 1

skinparam ParticipantPadding 2

skinparam BoxPadding 5 Autonumber

Box "Service Management & Orchestration Framework" #gold Participant "FOCOM/NFO" as FCC

End box

Box "O-Cloud

"#lightseagreen

Participant "IMS" as IMS Participant DMS as "DMS" Participant HAM as "HAM"

Participant "AAL \n Implementation" as aal

End box

Box "Network Functions"#LightBlue

Participant "<<AAL-App>> AAL Application" as App End box

Box "Network Functions"#DarkBlue Participant "SMO" as O1

End box

Box "SW Repo" "#Blue

Participant "Repo" as Repo End box

Note over FCC, DMS PRECONDITIONS:

SMO/FOCOM/NFO is active and running normally O-Cloud is available

IMS/DMS is operational

HAM is operational and registered with the IMS HW Accelerator is bootstrapped/installed

SMO and IMS/DMS connectivity is established IMS and HAM connectivity is established

End Note

== Use case: AAL Flow ==

Note over FCC PRECONDITIONS:

SMO Catalog, HW Accelerator Manifest, AAL Application Manifest available FW/SW image Repo available, SMO inventory DB available

End Note

FCC -> IMS: <<O2>> QueryOcloudInfoRequest IMS -> HAM : <<AALI-C-Mgmt>> getAalAccelInfo

HAM --> IMS : getAalAccelInfo response Success( Vendor, Version, Bios etc ) IMS --> FCC : <<O2>> QueryOcloudInfoResponse

FCC -> FCC : SMO inventory updated

Note over FCC,IMS

Single Set command to configure the device with manifest, including AAL LPU and Profile Images.

The SMO knows the pre-defined set of configurations that support different use cases that it wants to deploy.

The configuration can be changed dynamically in the future. The HAM will be able to apply delta changes accordingly. end Note

group Set configuration of the HW Accelerator

Note over FCC,IMS

SMO triggers configuration of the HW Accelerator End Note

IMS -> HAM: <<AALI-C-Mgmt>> setAalAccelConfig \n(e.g. AAL-LPU's, Supported AAL-Profile etc..)

Note over HAM, Repo

Optionally, the SW/FW image is retrieved from the Software Repository End Note

group DMS inventory update

HAM <--> DMS : DMS Info exchange Note over DMS

Examples of information exchanged \n AAL-LPUs, AAL-Profile info etc End Note

End

HAM --> IMS: setAalAccelConfig response success

HAM --> IMS : <<AALI-C-Mgmt>> aalInventoryNotification w/ managed resource informaton

Note over FCC,IMS

Response to SMO with success/failure of setting AAL Accel Config end Note

FCC -> FCC : SMO inventory update end

group Instantiate AAL Applications and optionally update HW Accelerator information Note over FCC,IMS

SMO Instantiate AAL Applications

SMO gets the deployment artifacts for

containerized workload (e.g., Manifest/ Helm Charts).

This includes managed resource and AAL-Profiles requested end Note

REF over FCC : Use case 3.2.1 Orchestration Use Case Spec FCC -> DMS : <<O2>> Instantiate AAL Application

Note over DMS

The DMS manifest includes the LPU (Profile) required and I/O ports etc.. end NOTE

opt Update HW Accelerator information if needed Note over DMS

Optionally, the IMS sets the AAL Accel Config based on the information received from SMO

to add/modify managed resources as defined in the AAL Application instantiate request.

End Note

DMS --> FCC : <<DMS-SMO interface>>Update HW Accelerator information(to be define by O2dms)

FCC -> IMS: <<O2>> setAalAccelConfig \n(including AAL-LPU's, Supported AAL-Profile)

IMS -> HAM: <<AALI-C-Mgmt>> setAalAccelConfig \n(e.g. AAL-LPU's, Supported AAL-Profile) HAM --> IMS: <<AALI-C-Mgmt>> setAalAccelConfig response success

HAM --> IMS : <<AALI-C-Mgmt>> aalInventoryNotification w/ O-Cloud AAL-LPU information IMS --> FCC : <<O2>> Set AAL Accel Config response success

FCC --> DMS: Success End opt

DMS -> App : <<AAL-App>> AAL Application instantiated activate App

DMS --> FCC : <<AAL-App>> AAL Application instantiated response success End

group AAL Application application and AAL configured

O1 -> App : <<O1>> DU Configuration \n Cells, BW, Profile Version etc.. App --> O1 : response Success

Note over aal, App

Initilize the AAL interface betwwen AAL and the AAL Application end NOTE

App <-> aal : Init AALI-C-App

Note over aal, App

Get AAL LPU information and create AAL Profile Instance end NOTE

App -> aal : <<AALI-C-App>> getAalLpuInfo

aal -> App : <<AALI-C-App>> Response LPU/Profile handle

App -> aal : <<AALI-C-App>> createAalProfileInstance aal -> App : <<AALI-C-App>> Response Success

App -> aal : <<AALI-C-App>> setAalProfileInstance/startAalProfileInstance/startAalProfileInstance aal -> App : <<AALI-C-App>> Response Success

App -> aal : <<AALI-C-App>> setAalProfileQueueConfig aal -> App : <<AALI-C-App>> Response Success

App -> aal : <<AALI-C-App>> startAalProfileQueue aal -> App : <<AALI-C-App>> Response Success

Note over aal, App

Delete AAL Profile Instance end NOTE

App -> aal : <<AALI-C-App>> stopAalProfileQueue aal -> App : <<AALI-C-App>> Response Success

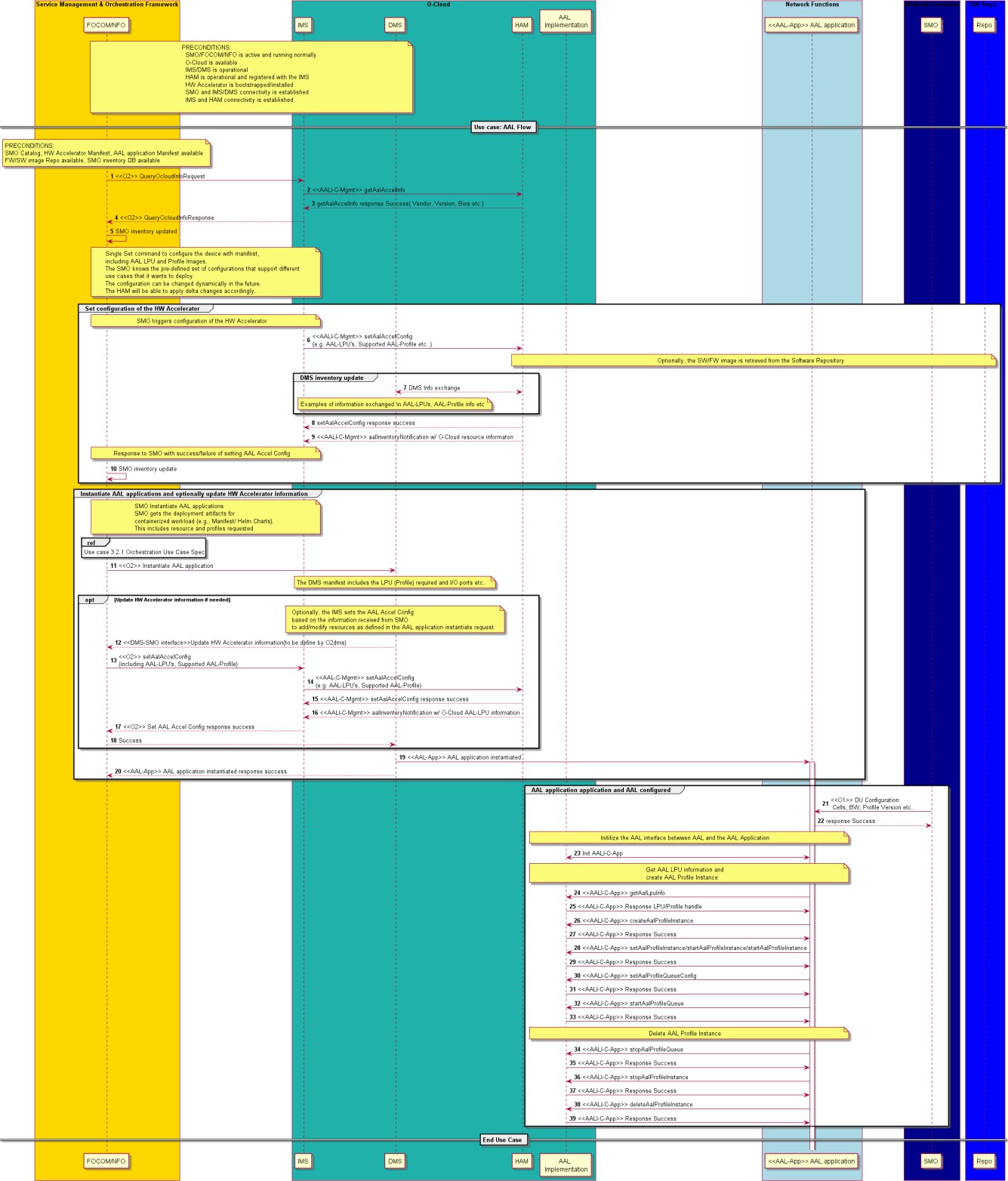
App -> aal : <<AALI-C-App>> stopAalProfileInstance aal -> App : <<AALI-C-App>> Response Success

App -> aal : <<AALI-C-App>> deleteAalProfileInstance aal -> App : <<AALI-C-App>> Response Success

end

== End Use Case ==

@enduml



### HW Accelerator Manager (HAM) registration use case workflow

The HAM registration procedure makes the HAM known to the IMS.

This use case describes how the HAM registers itself with the IMS using the AALI-C-Mgmt API. This registration at the minimum includes the HAM end point for the IMS to communicate with the HAM.

Following successful registration with the IMS, the HAM can participate in other resource management and AALI-C- Mgmt procedures.

Until successfully registered with an IMS, the HAM cannot accept management requests or provide notifications not associated with the IMS registration procedure.

NOTE: Installation and configuration of the HAM are not in the scope of this use case. How the HAM is configured with the IMS service endpoint and the local HAM service endpoint are implementation specific.

**Table 4.2.3-1: HW Accelerator Manager registration workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | Register the HW Accelerator Manager (HAM) with the IMS |  |
| Actors and Roles | SMO, FOCOM, IMS, HAM |  |
| Assumptions | This UC is applicable when the HAM is first configured with the IMS and HAM service endpoints or when the IMS or HAM service endpoint is modified. |  |
| Pre-conditions | SMO/FOCOM is active and running normally O-Cloud is available  IMS is operational and reachable.  HAM is configured and not registered with the IMS SMO and IMS connectivity is established |  |
| Begins when | When the HAM is ready to register with the IMS |  |
| Step 1 (M) | HAM initialized |  |
| Step 2 (M) | HAM registers itself with the IMS |  |
| Step 3 (M) | IMS sends a success response to the HAM |  |
| Ends when | The IMS updates its inventory for the availability of the HAM |  |
| Exceptions | None identified |  |
| Post Conditions | IMS can communicate with the HAM via its end point |  |
| Traceability | To be added later |  |

#### Sequence Diagram

@startuml

skinparam sequenceArrowThickness 1

skinparam ParticipantPadding 2

skinparam BoxPadding 5 Autonumber

Box "Service Management & Orchestration Framework" #gold Participant "FOCOM" as FCC

End box

Box "O-Cloud

"#lightseagreen

Participant "IMS" as IMS Participant AccelMgr as "HAM"

End box

Note over FCC,AccelMgr

HAM was is installed and configured with the IMS service endpoint End Note

AccelMgr --> AccelMgr : HAM is configured and not registered with the IMS

AccelMgr --> IMS : <<AALI-C-Mgmt>> aalHamRegistrationNotification (Registration Notification.callback URI)

IMS --> AccelMgr : <<AALI-C-Mgmt>> response (sucess)

Note over IMS

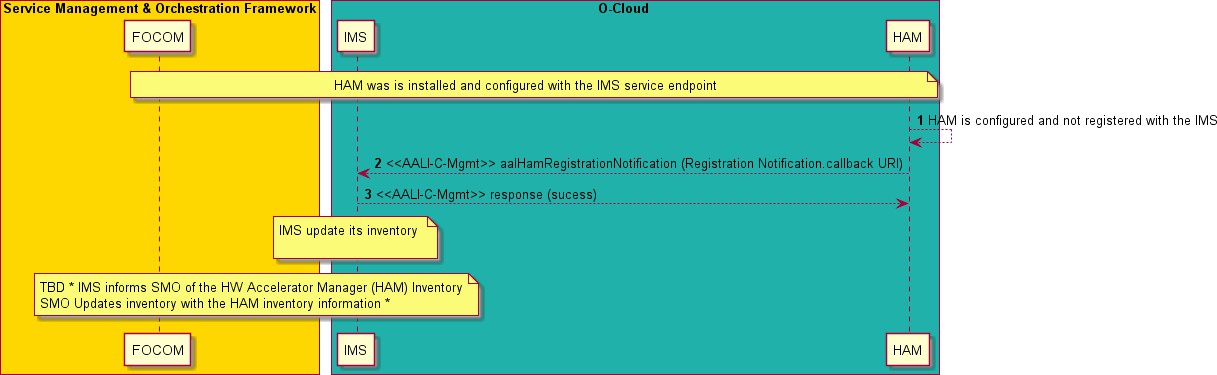
IMS update its inventory End Note

Note over FCC, IMS

TBD \* IMS informs SMO of the HW Accelerator Manager (HAM) Inventory SMO Updates inventory with the HAM inventory information \*

End Note

@enduml



## AALI-C-Mgmt API details

NOTE: The AALI-C-Mgmt interface defines the administrative operations/actions/events between the O-Cloud IMS and the HAM.

### Capabilities discovery and configuration

#### getAalHwAccelStatus

Description: This API is used to query the HAM about the status of the HW Accelerator.

Pre-conditions: HAM is registered with an IMS.

.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: N/A

**Table 4.3.1.1-1: getAalHwAccelStatus Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalHwAccelStatus | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.1.1-2: getAalHwAccelStatus Input Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id |  | Identifier | M | This ID is unique within a O-Cloud |

**Table 4.3.1.1-3: getAalHwAccelStatus Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_operatio nal\_state |  | *String* | M | Returns the operational state of the HW Accelerator |
| hw\_accel\_operatio nal\_conditions | 0..N | *Array* | M | Returns the operation conditions that support the operational state of the HW Accelerator. |
| status\_of\_operation | 1 | *Reference WG6 info model* | M | Returns the status of the operation |

#### getAalLpuStatus Description:

This API is used to query the HAM about the status of the HW Accelerator and the AAL-LPUs (per AAL-LPU).

).

Pre-conditions: HAM is registered with an IMS.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: N/A

**Table 4.3.1.2-1: getAalLpuStatus Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalLpuStatus | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.1.2-2: getAalLpuStatus Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id | 1.. N | Identifier | M | This ID is unique within an O-Cloud |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_lpu\_handle | 0..1 | Identifier | M | This ID is unique within a HW Accelerator |

**Table 4.3.1.2-3: getAalLpuStatus Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id | 1.. N | Identifier | M | This ID is unique within an O-Cloud |
| >aal\_lpu\_handle | 1 | Identifier | M | This ID is unique within a HW Accelerator |
| >aal\_lpu\_operation al\_state | 1 | String | M | Returns the operational state of the AAL-LPU. |
| >aall\_lpu\_operatio nal\_conditions | 0..N | Array | M | Returns the operation conditions that support the operational state of the AAL-LPU |
| status\_of\_operation | 1 | *Reference WG6 info model* | M | Returns the status of the operation |

#### getAalAccelInfo

Description: This API is used to query the HAM about a HW Accelerator and capabilities of its AAL-LPU(s).

Pre-conditions: HAM is registered with an IMS.

.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: N/A

**Table 4.3.1.3-1: getAalAccelInfo Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalAccelInfo | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.1.3-2: getAalAccelInfo Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id | 1.. N | Identifier | M | Unique Identification of the AAL-HW-Accel in the O-Cloud. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type**  **(Align with O2 Data model)** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG6 info model* | M | Returns the status of the get-AAL-Accel-info procedure |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type**  **(Align with O2 Data model)** | **Qualifier** | **Description** |
| hw\_accel\_list | 0.. N | List of struct  - inlined | M | List of Hardware Accelerator’s and their characteristics: |
| >hw\_accel\_id | 1 | Identifier | M | Unique Identification of the HW Accelerator in the O-Cloud. |
| >hw\_accel\_operati onal\_state | 1 | String | M | The operational state of the HW Accelerator |
| >hw\_accel\_operati onal\_conditions | 1 | Array | M | The operational conditions that support the operational state of the HW Accelerator |
| >vendor\_name | 1 | String | M | Name of the Vendor/Provider of the resource. |
| >date\_of\_manufact ure | 1 | String | M | Date of the manufacture of the resource. |
| >model | 1 | String | M | Information about the model of the resource. |
| >serial\_number | 1 | String | M | Serial number of the specific resource. |
| >hw\_version | 1 | String | M | Version or generation of the resource. |
| >hw\_accel\_image\_ version | 1 | String | M | The version for the image applied to the HW Accelerator. The parameter is associated with the HW Accelerator’s resource type. |
| >hw\_accel\_image\_ location | 1 | String | M | Path / URI to the software/firmware image. |
| >hw\_accel\_vendor  \_specific | 0.. N | List Key Value Pair  String/String | M | Vendor specific attributes associated with the HW Accelerator. |
| >max\_num\_aal\_lpu s | 0..1 | Integer | O | This parameter may be included if a HW Accelerator has an upper limit on AAL-LPUs it can support. Actual number of AAL-LPUs available at runtime may be lower based on resource usage, AAL-LPU SW used, and AAL-Profile SW and types instantiated on the AAL-LPU(s) |
| >aal\_profile\_templ ates | 0..N | List AALProfileT emplate | M | This parameter provides a list of templates for the AAL-Profiles that is supported for this HW Accelerator. |
| >num\_aal\_lpus\_co nfigured | 1 | Integer | M | Number of LPU(s) configured |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type**  **(Align with O2 Data model)** | **Qualifier** | **Description** |
| >lpu\_type | 1 | String | M | single/mixed  single: all LPUs are identical. mixed: non-identical LPUs |
| >aal\_lpu\_list | 0.. N | List of struct  - inlined | M | List of AAL-LPUs and their characteristics: |
| >> aal\_lpu\_handle | 1 | Identifier | M | Uniquely identifies the AAL-LPU within the accelerator |
| >>aal\_lpu\_administ rative\_state | 1 | String | M | The administrative state of the AAL-LPU |
| >>aal\_lpu\_operatio nal\_state | 1 | String | M | The operational state of the AAL-LPU |
| >>aal\_lpu\_operatio nal\_conditions | 1 | Array | M | The operational conditions that support the operational state of the AAL-LPU. |
| >>aal\_lpu\_image\_v ersion | 1 | String | M | The version or generation for the image applied to the AAL-LPU.  If an AAL-LPU image is the same as the Accelerator HW image then the hw\_accel\_image\_version and aal\_lpu\_image\_version parameters would contain the same value. |
| >>aal\_lpu\_image\_l ocation | 1 | String | M | Path / URI to the software/firmware image.  If an AAL-LPU image is the same as the Accelerator HW image then the hw\_accel\_image\_location and aal\_lpu\_image\_location parameters would contain the same value. |
| >>aal\_lpu\_profile\_l ist | 0..1 | List of Struct  - inlined | M | List of AAL-Profile(s) supported (Profile Name ID, Image) |
| >>>aal\_lpu\_profile  \_name | 1 | String | M | AAL Profile name as defined in section 3.3.3.1. |
| >>>aal\_lpu\_profile  \_version | 1 | String | M | AAL Profile version for the associated AAL Profile name. The AAL Profile version is the version of the AAL Profile’s approved specification. |
| >>>  aal\_lpu\_profile\_im age\_version | 1 | String | M | Version of the AAL Profile image (e.g. binaries, libraries, etc).  If an AAL Profile image is the same as the AAL- LPU image then the aal\_lpu\_image\_version and aal\_lpu\_profile\_image\_version parameters would contain the same value. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type**  **(Align with O2 Data model)** | **Qualifier** | **Description** |
| >>>  aal\_lpu\_profile\_im age\_location | 1 | String | M | Path / URI to the AAL Profile image (e.g. binaries, libraries, etc).  If an AAL Profile image is the same as the AAL- LPU image then the aal\_lpu\_image\_location and aal\_lpu\_profile\_image\_location parameters would contain the same value. |
| >>>aal\_lpu\_profile  \_attributes | 0..N | List Key Value Pair | M | This parameter provides a list of configuration attributes for the AAL-Profiles that used for this AAL-LPU. The structure and format of the Value attribute is specified as part of the definition of the Key. |
| >>>aal\_lpu\_profile  \_vendor\_specific | 0..N | List Key Value Pair | M | This parameter provides a list of vendor specific attributes for the AAL-Profiles that used for this AAL-LPU. The structure and format of the Value attribute is specified as part of the definition of the Key. |
| >>aal\_lpu\_config | 0..1 | Struct - Inlined | M | Can be one or more parameters that define the AAL-LPU persona – Some of the parameters are listed below |
| >>>aal\_lpu\_memor y | 0..1 | Integer | O | AAL-LPU memory |
| >>>aal\_lpu\_multip rocessors\_num | 0..1 | Integer | O | Number of Multi-Processors per AAL-LPU |
| >>>aal\_lpu\_compu te\_slices\_num | 0..1 | Integer | O | Number of compute slices per AAL-LPU |
| >>aal\_lpu\_vendor\_ specific | 0.. N | List Key Value Pair  String/String | M | Vendor specific AAL-LPU attribute.  Some examples: Number of DMA engines, number of encoders, number of decoders, number of JPEG engines, Number of OfA (Optical flow accelerator) engines |

NOTE: While this list provides the details of parameters between IMS and HAM, further analysis is needed to understand what is needed by the SMO.

#### setAalAccelConfig

Description: This is used to configure the HW Accelerator and AAL-LPU. The operation’s input parameters are the full configuration set of parameters for the HW Accelerator and contained AAL-LPUs. These input parameters replace the existing configuration of the HW Accelerator and AAL-LPUs. The HAM shall determine the changes to be applied to the HW Accelerator based on the input parameters contained in the operation and the current configuration parameters associated with the HW Accelerator. For example, if the operation contains an AAL-LPU instance that is part of the current configuration, the HAM adds the AAL-LPU instance to the current configuration. Likewise, if the current configuration has a configured AAL-LPU that is not contained within the input parameters of this operation, the HAM

deletes the AAL-LPU from the current configuration of the HW Accelerator. The HAM shall validate the state of the HW Accelerator and AAL-LPUs prior to applying the changes.

Pre-conditions:

* HW Accelerator Device is initialized, and HAM is operational.
* HW Accelerator Device and AAL-LPUs are in the correct state prior to performing configuration changes. Triggers: On demand request via AALI-C-Mgmt interface

Frequency: N/A

**Table 4.3.1.4-1: setAalAccelConfig Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| setAalAccelConfig | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.1.4-2: setAalAccelConfig Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_list | 1.. N | List of struct  - inlined | M | List of Hardware Accelerator’s and their characteristics: |
| >hw\_accel\_id | 1.. N | Identifier | M | Unique Identification of the AAL-HW-Accel in the O-Cloud.. |
| >hw\_accel\_image\_ version\_location | 1 | String | M | Path / URI to the HW Accelerator image (e.g., BIOS/FW/Host OS, libraries, drivers) to be used. |
| >num\_aal\_lpus\_co nfigured | 1 | Integer | M | Number of LPU(s) configured |
| >lpu\_type | 1 | String | M | single/mixed  single: all LPUs are identical. mixed: non-identical LPUs |
| >aal\_lpu\_list | 0.. N | List of struct  - inlined | M | List of AAL-LPUs and their characteristics: |
| >> aal\_lpu\_handle | 1 | Identifier | M | Uniquely identifies the AAL-LPU within the accelerator |
| >>aal\_lpu\_image\_v ersion\_location | 1 | String | M | Path / URI to the Image (e.g. FW/guest OS, libraries, drivers) to be used.  If an AAL-LPU image is the same as the Accelerator HW image then the  hw\_accel\_image\_version\_location and |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
|  |  |  |  | aal\_lpu\_image\_version\_location parameters would contain the same value |
| >hw\_accel\_vendor  \_specific | 0.. N | List Key Value Pair  String/String | M | Vendor specific attributes associated with the HW Accelerator. |
| >>  aal\_lpu\_profile\_list | 0.. N | List of Struct - inlined | M | List of AAL-Profile(s) supported (Profile Name ID, Image) |
| >>>  aal\_lpu\_profile\_na me | 1 | String | M | AAL Profile name as defined in section 3.3.3.1. |
| >>>  aal\_lpu\_profile\_ver sion | 1 | String | M | AAL Profile version for the associated AAL Profile name. The AAL Profile version is the version of the AAL Profile’s approved specification |
| >>>  aal\_lpu\_profile\_im age\_version | 1 | String | M | Version of the AAL Profile image (e.g. binaries, libraries, etc).  If an AAL Profile image version is the same as the AAL-LPU image version then the aal\_lpu\_image\_version and aal\_lpu\_profile\_image\_version parameters would contain the same value. |
| >>>  aal\_lpu\_profile\_im age\_location | 1 | String | M | Path / URI to the AAL Profile image (e.g. binaries, libraries, etc) to be used.  If an AAL Profile image is the same as the AAL- LPU image then the aal\_lpu\_image\_version\_location and aal\_lpu\_profile\_image\_location parameters would contain the same value. |
| >>>aal\_lpu\_profile  \_attributes | 0..N | List Key Value Pair | M | This parameter provides a list of configuration attributes for the AAL-Profiles that used for this AAL-LPU. The structure and format of the Value attribute is specified as part of the definition of the Key. |
| >>>aal\_lpu\_profile  \_vendor\_specific | 0..N | List Key Value Pair | M | This parameter provides a list of vendor specific attributes for the AAL-Profiles that used for this AAL-LPU. The structure and format of the Value attribute is specified as part of the definition of the Key. |
| >> aal\_lpu\_config | 0..1 | Struct - Inlined | M | Can be one or more parameters that define the AAL-LPU persona – Some of the parameters are listed below |
| >>aal\_lpu\_memory | 0..1 | Integer | O | AAL-LPU memory |
| >>aal\_lpu\_multipro cessors\_num | 0..1 | Integer | O | Number of Multi-Processors per AAL-LPU |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
|  |  |  |  |  |
| >>aal\_lpu\_compute  \_slices\_num | 0..1 | Integer | O | Number of compute slices per AAL-LPU |
| >>aal\_lpu\_vendor\_ specific | 0.. N | List Key Value Pair  String/String | M | Vendor specific AAL-LPU attribute. Some examples: Number of DMA engines, number of encoders, number of decoders, number of JPEG engines, Number of OfA (Optical flow accelerator) engines |

**Table 4.3.1.4-3: setAalAccelConfig Output Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG6 info model* | M | Returns the status of the procedure |

* The vendor specific attributes in the getAalAccelInfo and setAalAccelConfig needs to consider vendor specific attributes for the LPU and Accelerator.

#### startAalLpu

Description: This operation will start an AAL-LPU that is in the stopped state. The AAL-LPU is instantiated using the SW/FW image(s) and configuration that were provided in the setAalAccelConfig procedure.

The AAL-LPU state machine is moved to a started state. When the AAL-LPU is in the started state it is ready to be used by the AAL Application.

After successfully starting AAL-LPU using this procedure, the AAL-LPU can be stopped.

Pre-conditions: HW Accelerator Device is initialized, and HAM is operational. The AAL-LPU has been successfully configured and is in the stopped state.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: N/A

**Table 4.3.1.5-1: startAalLpu Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| startAalLpu | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.1.5-2: startAalLpu Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id | 1 | Identifier | M | HW Accel ID is unique within an O-Cloud |
| aal\_lpu\_handle | 1 | Identifier | M | AAL-LPU handle is unique within a HW Accelerator |

**Table 4.3.1.5-3: startAalLpu Output Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG6 info model* | M | Returns the status of the procedure |

#### stopAalLpu

Description: Moves AAL-LPU state machine to a stopped state.

AAL-LPU service is stopped. Configuration changes on the AAL-LPU can be performed. Re-configuration of the AAL-LPU compute / memory resource partitioning can be performed. Drivers can be updated / re-installed and configured. Maintenance / debugging operations can be performed.

After successfully stopping the AAL-LPU, the AAL-LPU can be (re-)configured, started or deleted.

Pre-conditions: HW Accelerator Device is initialized, and HAM is operational. The number of AAL-LPUs has been set on the HW Accelerator. The AAL-LPU has been successfully configured and is started. AAL Applications have previously been migrated off the AAL-LPU.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: N/A

**Table 4.3.1.6-1: stopAalLpu Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| stopAalLpu | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.1.6-2: stopAalLpu Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id | 1 | Identifier | M | This ID is unique within an O-Cloud |
| aal\_lpu\_handle | 1 | Identifier | M | This ID is unique within a HW Accelerator |

**Table 4.3.1.6-3: stopAalLpu Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure |

#### Inventory Notifications

Description: Inventory notifications are sent by the HAM to the IMS when the HAM discovers that managed resources have been added, modified, or removed from the HAM’s inventory of managed resources. Inventory notifications are not used to report administrative or operational state changes in the resources (e.g., restarts, configuration changes that do not include added or deleted managed resources). The Inventory Notification services in this section, provide the ability for the IMS to receive inventory notification events when managed resources have been added, modified or removed from the HAM.

The modification of a managed resource differs from addition and removal of an O-Cloud Resource. Examples of modifications of a managed resource that causes an Inventory Notification event to be emitted by the HAM would be the modification of the path to the image for an AAL-LPU or the modification of the compute resources assigned to the AAL-LPU..

##### aalInventoryNotification

Description: The HW Accelerator Manager emits notifications to the IMS. The [4.2.1 HW Accelerator](#_bookmark40) [installation/bootstrap use case workflow,](#_bookmark40) [4.2.2 end-2end Life Cycle Management use case workflow (Informational)](#_bookmark42) provide examples of when the inventory notification is sent by the HAM to the IMS that has previously subscribed to receive the notification event.

Pre-conditions: The HAM is operational. The IMS and the HAM have established connectivity. Triggers: On demand request via AALI-C-Mgmt interface.

Frequency: Infrequent

**Table 4.3.1.7-1: aalInventoryNotification Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| aalInventoryNotification | HW Accelerator Manager (HAM) => IMS |

**Table 4.3.1.7-2: aalInventoryNotification Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| resources | 0.. N | Structure | M | The current managed resource list. |
| > resource | 1 | Structure | M | The resource (HW Accelerator, AAL-LPU) and its associated attributes known by the HAM.  Section 3.3.1 contains the list of attributes for managed resource that are retrieval and are part of the Inventory Notification event.  When the HW Accelerator is newly discovered, the hw\_accel\_id attribute of the resource does not have a value reported since the value of this attribute is assigned by the IMS. |
| > detected\_time | 1 | DateTime | M | The time the inventory event was last detected. |
| > event | 1 | Enumeration | M | The event that caused the notification.  Values: Add, Modify, Remove |

#### setAalAccelIdentifier

Description: This is used to provide the HW Accelerator with its resource identifier (hw\_accel\_id) assigned by the IMS. The HAM uses the vendor, model, and serial\_number input parameters to identify the HW Accelerator and then assigns the resource identifier to the HW Accelerator.

Pre-conditions: HW Accelerator has been discovered but needs its identifier assigned by the IMS, and HAM is operational.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: N/A

**Table 4.3.1.8-1: setAalAccelIdentifier Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| setAalAccelIdentifier | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.1.8-2: setAalAccelIdentifier Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| vendor\_name | 1 | String | M | Input parameter that is used by the HAM to match the vendor of the HW Accelerator. |
| model | 1 | String | M | Input parameter that is used by the HAM to match the model of the HW Accelerator. |
| serial\_number | 1 | String | M | Input parameter that is used by the HAM to match the serial number of the HW Accelerator. |
| hw\_accel\_id | 1 | Identifier | M | Unique Identification of the AAL-HW-Accel in the O-Cloud.  If the HAM has an existing hw\_accel\_id value mapped to the device identified by vendor\_name, model and serial\_number, the HAM’s value will be replaced with the value provided in this parameter. |

**Table 4.3.1.8-3: setAalAccelIdentifier Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG6 info model* | M | Returns the status of the procedure |

#### aalHamRegistrationNotification

Description: The HAM Registration Notification API is used by the HAM to register itself with the SMO. This occurs:

* After the HAM is ready to support interactions with the IMS.
* After HAM is configured with the IMS and HAM's service endpoints.
* When the HAM or IMS service endpoint has been modified.

**Table 4.3.1.9-1: aalHamRegistrationNotification Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| aalHamRegistrationNotification | HW Accelerator Manager (HAM) => IMS |

**Table 4.3.1.9-2: aalHamRegistrationNotification Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aalHam\_ep | 1 | String | M | The service end point for the IMS to communicate with the HAM. |

### Infrastructure Monitoring Service

#### Fault Service Operations

The AALI-C-Mgmt API provides operations to query the current Fault conditions for managed resources and raise event notifications to subscribed entities when Fault conditions occur. These Fault conditions are operational conditions of the managed resource and can be retrieved from the “faults” attribute of the associated managed resource.

The HAM provides the capability to query the current Fault conditions for the HW Accelerator and AAL-LPU(s).

##### getAalHwAccelFaults

Description: This API is used between the IMS and HAM to query the currently active faults of the HW Accelerator. Pre-conditions: HW Accelerator Device is initialized, and HAM is operational.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: Infrequent

**Table 4.3.2.1.1-1: getAalHwAccelFaults Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalHwAccelFaults | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.2.1.1-2: getAalHwAccelFaults Input Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id |  | Identifier | M | This ID is unique within a O-Cloud |

**Table 4.3.2.1.1-3: getAalHwAccelFaults Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG6 info model* |  | Returns the status of the operation |
| faults | 0.. N | Structure | M | The currently active Fault list |
| > hw\_accel\_id | 1 | Identifier | M | This ID is unique within an O-Cloud |
| > detected\_time | 1 | DateTime | M | The time the Fault condition was last detected |
| > fault\_id | 1 | Identifier | M | The Fault identifier. |

##### getAalLpuFaults

Description: This API is used between the IMS and HAM to query the currently active faults of the AAL-LPUs (per AAL-LPU).

Pre-conditions: HW Accelerator Device is initialized, and HAM is operational. Triggers: On demand request via AALI-C-Mgmt interface

Frequency: Infrequent

**Table 4.3.2.1.2-1: getAalHwAccelFaults Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalLpuFaults | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.2.1.2-2: getAalHwAccelFaults Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| hw\_accel\_id | 1.. N | Identifier | M | This ID is unique within an O-Cloud |
| aal\_lpu\_handle |  | Identifier | M | This ID is unique within a HW Accelerator |

**Table 4.3.2.1.2-3: getAalHwAccelFaults Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG10 object model* | M | Returns the status of the operation |
| faults | 0.. N | Structure | M | The currently active Fault conditions list |
| > hw\_accel\_id | 1 | Identifier | M | This ID is unique within an O-Cloud |
| > aal\_lpu\_handle | 1 | Identifier | M | This ID is unique within a HW Accelerator needed for LPU Fault conditions. |
| > detected\_time | 1 | DateTime | M | The time the Fault condition was last detected |
| > fault\_id | 1 | Identifier | M | The Fault identifier. |

#### Fault Notifications

The HW Accelerator Manager (HAM) provides the capability to subscribe to receive Fault conditions for the HW Accelerator and AAL-LPU(s) that it is managing. This capability is managed within the O-Cloud through the subscription\_id identifier.

##### Subscription\_id

The subscription\_id is an identifier used to describe the fault notification relationship between the IMS and a given HAM; it applies indirectly to the notification of faults from all the HW Accelerators that the HAM is managing.

The subscription\_id includes a list of all the faults that the IMS needs to be notified in the event of their occurrence in any one of the HW Accelerators that is being managed by the HAM.

The validity and duration of the subscription\_id shall be defined by the following:

The existence and validity of the IMS to HAM relationship. For example, if the HAM needs to be updated or modified then a new subscription\_id needs to be issued by the IMS.

The existence and validity of any of the HW Accelerators for which the HAM is responsible for and is managing for the IMS. If any one of these HW Accelerators requires an update, upgrade, or any other modification is performed then the subscription\_id between the IMS and the given HAM needs to be renewed.

Any notification that the HAM issues to the IMS pertaining to a modification in the state or operation of the HW Accelerator by the HAM shall result in an update/renewal of the subscription\_id with the IMS.

In the case of restart or power-cycles in the HW Accelerator managed by the HAM the subscription\_id doesn't have to be renewed since restart or power-cycle does not alter the faults supported by the HW Accelerator or the fault formats.

If the IMS receives a fault notification from a HAM that includes a subscription\_id that does not match a subscription\_id maintained by the IMS for that given HAM, the IMS the IMS shall then delete that subscription by calling the deleteAalFaultSubscription API with the subscription\_id followed by calling the createAalFaultSubscription API with new subscription\_id(s) to (re-)create a new fault subscription.

##### createAalFaultSubscription

Description: This API is used between the IMS and HAM to create fault subscriptions for AAL-LPUs and the HW Accelerator managed resources.

Pre-conditions: HW Accelerator Device is initialized, and HAM is operational. Triggers: On demand request via AALI-C-Mgmt interface.

Frequency: Infrequent

**Table 4.3.2.2.2-1: createAalFaultSubscription Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| createAalFaultSubscription | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.2.2.2-2: createAalFaultSubscription Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| subscriptions | 1.. N | Structure | M | The subscriptions to be created. |
| > subscription\_id | 1 | Identifier | M | Unique identification of the subscription in the O- Cloud. |
| > filter\_criteria | 1 | String | M | Policy to subscribe that is evaluated by the HAM to determine what faults for which AAL managed resource types and instances should be notified to the IMS.  Each criteria includes the Fault condition(s) for a managed resource type. The criteria accepts Fault condition/ managed resource types that are associated with a set of instances of that type of resource.  Multiple criteria can be part of the subscription with criteria evaluation operators (i.e., and, or) that can be used to evaluate the individual criterion.  For example:  To subscribe to temperature faults for all HW Accelerator resources and the memory exhausted fault for an AAL-LPU with an identifier of 45; the subscription looks like:  ((TemperatureFault HW-Accel \*) and (MemoryExhausted AAL-LPU 45)) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
|  |  |  |  | NOTE: The filter\_criteria element is further described as part of the protocol specific implementation of this operation. |

**Table 4.3.2.2.2-3: createAalFaultSubscription Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| subscriptions | 1.. N | Structure | M | Status of the subscriptions. |
| > subscription\_id | 1 | Identifier | M | Returns the identifier of the subscription. |
| >  status\_of\_operation | 1 | *Reference WG6 info model* | M | Returns the status of the operation. |

##### getAalFaultSubscription

Description: This API is used between the IMS and HAM to query Fault Subscriptions for the HW Accelerator. Pre-conditions: HW Accelerator Device is initialized, and HAM is operational.

Triggers: On demand request via AALI-C-Mgmt interface Frequency: Infrequent

**Table 4.3.2.2.3-1: getAalFaultSubscription Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalFaultSubscription | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.2.2.3-2: getAalFaultSubscription Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| subscription\_id | 1.. N | Identifier | O | List of identifiers to retrieve.  If this attribute is not present, all subscriptions in the HAM are returned. |

**Table 4.3.2.2.3-3: getAalFaultSubscription Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| subscriptions | 0.. N | Structure | M | Returns the subscriptions |
| > subscription\_id | 1 | Identifier | M | Returns the identifiers of the requested subscriptions. |
| > filter\_criteria | 1 | String | M | Criteria evaluated by the HAM to determine what faults for which AAL managed resource types and instances should be notified to the IMS. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| >  status\_of\_operation | 1 | *Reference WG6 info model* |  | Returns the status of the operation for each subscription identifier. |

##### deleteAalFaultSubscription

Description: The HAM provides the ability for the IMS to delete a subscription for notification of faults. Pre-conditions: HW Accelerator Device is initialized, and HAM is operational.

Triggers: On demand request via AALI-C-Mgmt interface. Frequency: Infrequent

**Table 4.3.2.2.4-1: deleteAalFaultSubscription Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| deleteAalFaultSubscription | IMS => HW Accelerator Manager (HAM) |

**Table 4.3.2.2.4-2: deleteAalFaultSubscription Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| subscription\_id | 1.. N | Identifier | M | List of identifiers to delete |

**Table 4.3.2.2.4-3: deleteAalFaultSubscription Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| subscriptions | 1.. N | Structure | M | Returns the subscriptions that were deleted. |
| > subscription\_id | 1 | Identifier | M | Returns the identifiers of the subscriptions that were deleted. |
| >  status\_of\_operation | 1 | *Reference WG6 info model* |  | Returns the status of the operation for each subscription identifier. |

##### aalFaultNotification

Description: The HW Accelerator Manager emits notifications to the IMS based on a subscription. Pre-conditions: HW Accelerator Device is initialized, and HAM is operational.

Triggers: On demand request via AALI-C-Mgmt interface. Frequency: Infrequent

**Table 4.3.2.2.5-1: aalFaultNotification Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| aalFaultNotification | HW Accelerator Manager (HAM) => IMS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| subscription\_id | 1 | Identifier | M | Identifier of the subscription. |
| faults | 0.. N | Structure | M | The current fault list. |
| > hw\_accel\_id | 1 | Identifier | M | This ID is unique within an O-Cloud. |
| > aal\_lpu\_handle | 1 | Identifier | O | This ID is unique within a HW Accelerator needed for LPU faults. |
| > detected\_time | 1 | DateTime | M | The time the fault condition was last detected. |
| > event | 1 | Enumeration | M | The event that caused the notification.  Values: Raise, Clear |
| > fault\_id | 1 | Identifier | M | The identifier of the fault condition. |

# Chapter 5 AALI-C-APP

## AALI-C-App Identification & Representation

Within the AAL Application's process address space each, AAL-LPU can be represented by a generalized object handle (i.e., aal\_lpu\_handle) provided by the AAL implementation in order for the AAL Application to use the capabilities of the AAL-LPU.

Before the AALI-C-App interface can be used, the AAL implementation must be initialized by the AAL Application. The initialization of the AAL implementation allows for initialization, setup and discovery routines that are specific to an implementation. Where needed, the initialization of the AAL implementation also allows the AAL Application to inform the AAL implementation what O-Cloud Resources and associated attributes that the DMS has assigned to and can be used by the AAL Application. As an example, the DMS may inform the AAL Application about the AAL Application’s assigned O-Cloud Resource using environment variables that identify the O-Cloud Resource and associated attributes within the POD/container/VM. The identification of these O-Cloud Resources are to be shared with the AAL implementation by the AAL Application through the AALI-C-App initialization operation.

Depending on the HW design and implementation choice, an AAL-LPU implementation may want to accelerate multiple AAL Profiles, offer support for sharing HW Accelerator resources between multiple threads and/or partition AAL-LPU resources between multiple cells. For this reason, the AAL-Profile-Instance and AAL Profile Queue logical managed resources can optionally be used by the AAL Application to

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

As a logical managed resources, an AAL Profile Queue does not reflect a HW design specification or requirement but an AAL interface specification.

The AALI-C-App configuration is done on a per AAL-LPU basis. That is, for each AAL-LPU that is assigned to the AAL Application, the AAL Implementation exposes the AALI-C-App to the AAL Application to initialize and configure each AAL-Profile-Instance and AAL-Profile-Queue.

## AALI-C-App operations and events

The below table lists the operations and events as part of the AALI-C-App interface.

**Table 5.2-1: AALI-C-App operations and events**

|  |  |  |
| --- | --- | --- |
| **Category** | **Operations/**  **Events** | **Objects** |
| **Initialization Services** | initAal | N/A |
|  | cleanupAal | N/A |
| **Inventory Services** | getAalLpuInfo | AAL-LPU |
| **Configuration Services** | createAalProfileInstance | AAL-LPU |
|  | getAalProfileInstanceConfig | AAL-Profile-Instance |
|  | setAalProfileInstanceConfig | AAL-Profile-Instance |
|  | deleteAalProfileInstance | AAL-Profile-Instance |
|  | setAalProfileQueueConfig | AAL-Profile-Queue |
|  | getAalProfileQueueConfig | AAL-Profile-Queue |
| **Operations** | startAalProfileInstance | AAL-Profile-Instance |
|  | stopAalProfileInstance | AAL-Profile-Instance |
|  | startAalProfileQueue | AAL-Profile-Queue |
|  | stopAalProfileQueue | AAL-Profile-Queue |
| **Transport Operations** | createBufferPool | N/A |
|  | allocBuffer | N/A |
|  | getBufferAddr | N/A |
|  | getBufferSize | N/A |
|  | sendBuffer | N/A |
|  | receiveBuffer | N/A |
|  | freeBuffer | N/A |
|  | destroyBufferPool | N/A |

*Editor’s note: Use of AAL-Profile-Queue operations for user plane data transfer and buffer API operations needs to be clarified.*

## AALI-C-App Operations and Events Sequence and State Diagram

### AALI-C-App State

The below state diagram shows the high-level state and state transitions for all AALI-C-App Operations and Events

@startuml

skinparam StateBackgroundColor #FEFECE skinparam StateBorderColor #A80036 skinparam ArrowColor #A80036

skinparam ArrowHeadColor #A80036

skinparam ParticipantPadding 5

skinparam BoxPadding 10

hide members

hide empty description [\*] --> AAL : initAal

state AAL {

[\*] --> AalInitialized

AalInitialized --> AalInitialized : getAalLpuInfo AalInitialized -left-> [\*] : cleanupAal AalInitialized --> AalProfileInstance

state AalProfileInstance {

[\*] --> Stopped : createAalProfileInstance (s) Stopped --> Started : startAalProfileInstance Started --> Stopped : stopAalProfileInstance

Stopped --> Stopped : setAalProfileInstanceConfig\ngetAalProfileInstanceConfig Started --> Started : getAalProfileInstanceConfig

Stopped --> [\*] : deleteAalProfileInstance Started --> AalProfileQueue

state AalProfileQueue { [\*] --> QueueStopped

QueueStopped --> QueueStarted : startAalProfileQueue QueueStarted --> QueueStopped : stopAalProfileQueue

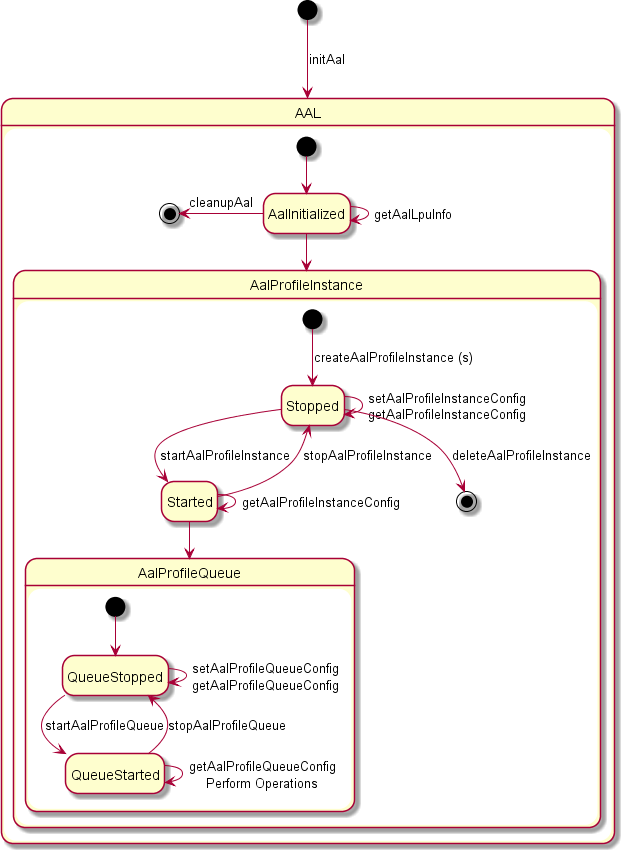
QueueStopped --> QueueStopped : setAalProfileQueueConfig\ngetAalProfileQueueConfig QueueStarted --> QueueStarted : getAalProfileQueueConfig\nPerform Operations

}

}

}

@enduml



### AALI-C-App API Init Sequence

The below sequence diagram shows the high-level initialization and configuration procedure for the AALI-C-App.

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << OAAL-App>> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box

note over ODU, AAL

Assumed the O-DU is already instantiated on the O-Cloud With desired HW Accelerator resources assigned from O2 and Configuration through O1 interface is complete

endnote

==Initialization Services== ODU -> AAL : initAal

AAL --> ODU : return

==Inventory Services==

ODU -> AAL : getAalLpuInfo

AAL --> ODU : Response getAalLpuInfo

==Configuration & Operation Services== loop for each AAL-LPU

loop for each AAL-Profile-Instance ODU -> AAL : createAalProfileInstance

AAL --> ODU : AAL-Profile-Instance handle / identifier ODU -> AAL : setAalProfileInstanceConfig

AAL --> ODU : return

ODU -> AAL : startAalProfileInstance AAL --> ODU : return

loop for each AAL-Profile-Queue

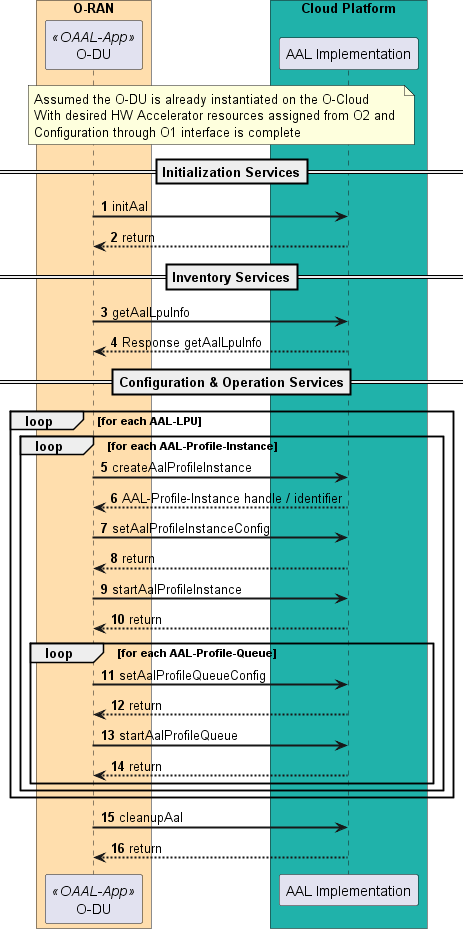
ODU -> AAL : setAalProfileQueueConfig AAL --> ODU : return

ODU -> AAL : startAalProfileQueue AAL --> ODU : return

end loop end loop end loop

ODU -> AAL : cleanupAal AAL --> ODU : return

@enduml



### AALI-C-Transport API Sequence

The AALI-C-App Transport API sequence is divided into two sets of call flows in the following subsections:

* Sec. 5.3.3.1 focusses on call flows related to allocation of buffers(s) to be used with other AALI-C-App Transport APIs.
* Sec. 5.3.3.2 and its subsections focus on call flows detailing the various procedures for sending and receiving buffer(s) using AALI-C-App Transport API.

#### Allocation of AAL Buffer(s)

The AALI-C-App Transport API supports allocation of buffers from either pre-created buffer pool(s) or from memory not mapped to pre-created buffer pool(s). Send buffer(s) are allocated by the AAL Application, while Receive buffer(s) are allocated either by the AAL Application or by the AAL Implementation.

.

NOTE 2: The terms "AAL" and "AAL Implementation" are used interchangeably in the tables below.

**Table 5.3.3.1-1 : Allocation of AAL Buffer(s) workflow steps**

|  |  |  |
| --- | --- | --- |
|  | Evolution / Specification | <<Uses>>  Related use |
| Goal | The AAL Application (e.g., O-DU) creates and allocates memory to be used when offloading operations to the AAL Implementation. |  |
| Actors and Roles | AAL Application: calls buffer operations with desired input parameters.  AAL: executes buffer operations. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the buffer operations. |  |
| Pre-conditions | AAL Implementation is initialized. |  |
| Begins when | The AAL Application is ready to create buffers. |  |
| Steps 1-2 (O) | The AAL Application can optionally create a buffer pool by calling the createBufferPool operation. On success, the AAL returns a buffer\_pool\_handle to the created buffer pool. |  |
| Steps 3-4 (M) | The AAL Application allocates a buffer to use in future AAL operations. The buffer can be allocated from a pre-created buffer pool from steps 1 and 2, or can be dynamically allocated. On success, the AAL returns a buffer\_handle. |  |
| Steps 5 – 8 (O) | The AAL Application can retrieve the pointer to the buffer to add/modify/delete the content of the buffers. The AAL Application can also check the size of the buffer. |  |
| Steps 9 – 10 (M) | Once the AAL Application is finished with the buffer it can free the buffer by calling the bufferFree operation. |  |
| Steps 11 – 12 (O) | If previously created in steps 1 and 2 and the AAL Application is finished with the buffer pool, the AAL Application will destroy the buffer pool. |  |

|  |  |  |
| --- | --- | --- |
| Ends when | All memory is freed and returned to the AAL Implementation. |  |
| Exceptions | Exceptions can occur in steps 1, 3, 5, 7, 9 and 10 if the AAL fails to allocate or free memory. The AAL will report an error. |  |
| Post Conditions | Memory is freed, AAL Application is free to continue processing. |  |
| Traceability | To be added later. |  |

* + - * 1. Sequence Diagram

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App>> end box

Box “Cloud Platform” #lightseagreen

participant “AAL Implementation” as AAL

End box

activate ODU

ref over ODU, AAL

AALI-C-App Sequence assumptions (Sec. 5.3.2) End ref

==Inventory Services== ref over ODU, AAL

AALI-C-App API Sequence call flow related to Inventory Service (Sec. 5.3.2) End ref

==Creating Buffer Pools and Allocating Buffers == opt if buffer pool creation

ODU -> AAL : createBufferPool activate AAL

AAL --> ODU : return buffer pool handle(s) Deactivate AAL

End opt

Note over ODU, AAL

Allocate a buffer using pre-created buffer pool with buffer\_pool\_handle or allocate dynamically with NULL buffer\_pool\_handle parameter

Endnote

ODU -> AAL : allocBuffer activate AAL

AAL --> ODU : return buffer handle deactivate AAL

autoactivate on

opt Get buffer address and buffer size using buffer handle if needed ODU -> AAL : getBufferAddr

AAL --> ODU: return buffer address ODU -> AAL: getBufferSize

AAL --> ODU: return buffer size End opt

ref over ODU, AAL

AALI-C-App API Sequence call flow related to AAL-LPU, AAL-Profile-Instance

and AAL-Profile-Queue initialization & configuration (Sec. 5.3.2) end ref

==Run Time Operations == ref over ODU, AAL

AALI-C-App API Sequence call flow related to Sending/Receiving Buffer(s) (Sec. 5.3.3.2) End ref

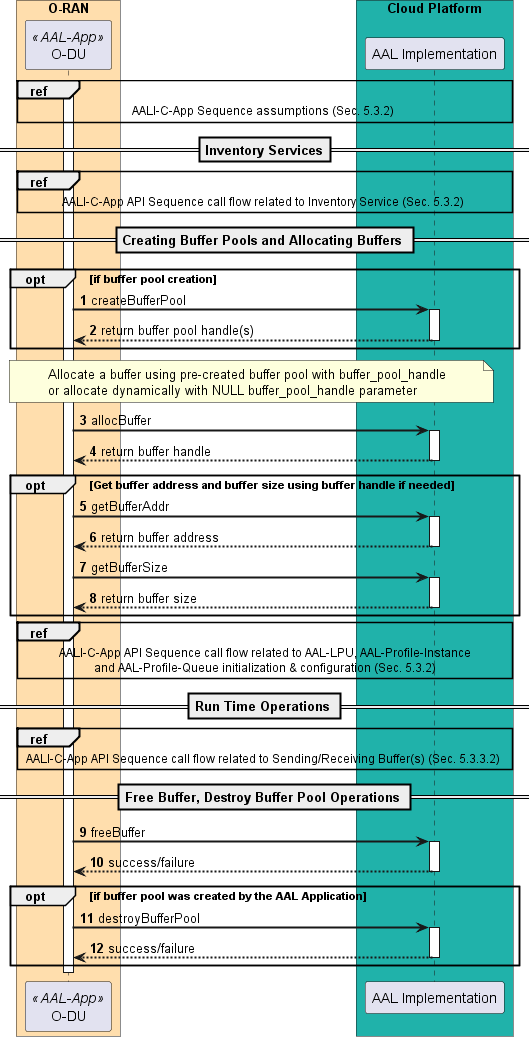
==Free Buffer, Destroy Buffer Pool Operations == ODU -> AAL : freeBuffer

AAL --> ODU : success/failure

opt if buffer pool was created by the AAL Application ODU -> AAL : destroyBufferPool

AAL --> ODU : success/failure End opt

deactivate ODU @enduml



#### Procedure for Buffer(s) Send and Receive

The AALI-C-App Transport API supports the following high level use cases

* Send Buffer Synchronous Mode with buffer free
* Send Buffer Synchronous Mode without buffer free
* Send Buffer Asynchronous Mode with buffer free
* Send Buffer Asynchronous Mode without buffer free
* Receive Buffer Synchronous Mode with non-NULL buffer\_handles(s) input
* Receive Buffer Synchronous Mode with NULL buffer\_handles(s) input
* Receive Buffer Asynchronous Mode with non-NULL buffer\_handle(s) input
* Receive Buffer Asynchronous Mode with NULL buffer\_handle(s) input

In the use cases above, 'with buffer free' and 'without buffer free' imply that the input parameter 'buffer\_free' = 'TRUE' (or 1) and 'FALSE' (or 0) respectively in sendBuffer operation (Sec. 5.4.3.5).

These use cases are detailed in the sections below.

##### Send Buffer Synchronous Mode with buffer free

**Table 5.3.3.2.1-1 : Send Buffer Synchronous Mode with buffer free workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | The AAL Application (e.g., O-DU) sends buffer(s) to the AAL for processing in synchronous mode with buffer free. |  |
| Actors and Roles | AAL Application: formats buffer and its content, and calls AAL to send the buffer.  AAL Implementation executes sendBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the sendBuffer operation in a synchronous manner with buffer free enabled. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. Buffer(s) to be sent are allocated. |  |
| Begins when | The AAL Application is ready to send buffer. |  |
| Step 1 (M) | The AAL Application calls the sendBuffer API to transfer the buffer to the AAL which will execute the sendBuffer operation. The AAL Application thread invoking the API is blocked waiting for the sendBuffer operation to complete. |  |
| Steps 2 - 3 (M) | The AAL internally frees the buffer back to the buffer pool upon completion of the sendBuffer operation.  Once done, the AAL returns the status of the operation back to the AAL Application and unblock the AAL Application call thread process. |  |
| Ends when | sendBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the sendBuffer operation fails. Status indicating failure will be returned in Step 3. |  |

|  |  |  |
| --- | --- | --- |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10 autoactivate on

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App>> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Synchronous Send Buffer (with buffer free) == ODU -> AAL : sendBuffer

Deactivate ODU

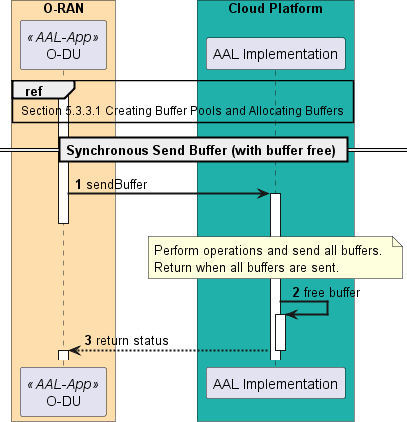
note over AAL

Perform operations and send all buffers. Return when all buffers are sent. endnote

AAL -> AAL : free buffer

AAL --> ODU : return status Activate ODU

deactivate ODU @enduml



##### Send Buffer Synchronous Mode without buffer free

**Table 5.3.3.2.2-1 : Send Buffer Synchronous Mode without buffer free workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | The AAL Application sends buffer(s) to the AAL for processing in synchronous mode without buffer free. |  |
| Actors and Roles | AAL Application: formats buffer and its content, and calls AAL to send the buffer.  AAL Implementation: executes sendBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the sendBuffer operation in a synchronous manner without buffer free enabled. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. Buffer(s) to be sent are allocated. |  |
| Begins when | The AAL Application is ready to send buffer. |  |
| Steps 1 – 2 (M) | The AAL Application calls the sendBuffer API to transfer the buffer to the AAL which will execute the sendBuffer operation. The AAL Application thread invoking the API is blocked waiting for the sendBuffer operation to complete. |  |

|  |  |  |
| --- | --- | --- |
|  | The AAL processes the operation and sends the buffers. Once done, the AAL returns the status of the operation back to the AAL Application and unblock the AAL Application call thread process. |  |
| Step 3 (M) | The AAL Application, when finished with the buffer, calls freeBuffer operation and the AAL will free the buffer back to the buffer pool. |  |
| Ends when | sendBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the sendBuffer operation fails. Status indicating failure will be returned in Step 2. |  |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10 autoactivate on

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App>> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Synchronous Send Buffer (without buffer free) ==

ODU -> AAL : sendBuffer deactivate ODU

note over AAL

Perform operations and send all buffers. Return when all buffers are sent.

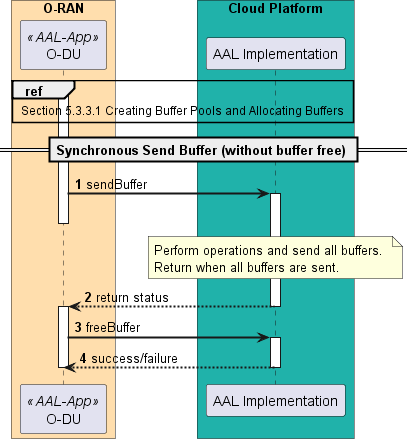
Endnote

AAL --> ODU : return status activate ODU

ODU -> AAL: freeBuffer

AAL --> ODU : success/failure

deactivate ODU @enduml



##### Send Buffer Asynchronous Mode with buffer free

**Table 5.3.3.2.3-1 : Send Buffer Synchronous Mode without buffer free workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | The AAL Application sends buffer(s) to the AAL for processing in asynchronous mode with buffer free. |  |
| Actors and Roles | AAL Application: formats buffer and its content, and calls AAL to send the buffer.  AAL Implementation: executes sendBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the sendBuffer operation in an asynchronous manner with buffer free enabled. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. Buffer(s) to be sent are allocated. |  |
| Begins when | The AAL Application is ready to send buffer. |  |
| Steps 1 – 3 (M) | The AAL Application calls the sendBuffer API to transfer the buffer to the AAL which will execute the sendBuffer operation.  The AAL Application thread invoking the API is returned immediately with the status of the API call, without being blocked until the completion of sendBuffer operation. |  |

|  |  |  |
| --- | --- | --- |
|  | The AAL continues to process the operation and sends the buffers. Upon completion of sendBuffer operation, the AAL internally frees the buffer back to the buffer pool. |  |
| Step 4 (O) | Once done, the AAL optionally returns the status of the operation back to the AAL Application, if the AAL Application preregisters for a callback. |  |
| Ends when | sendBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the sendBuffer API call fails. Status indicating failure will be returned in Step 2. |  |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App>> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Asynchronous Send Buffer (with buffer free) == ODU -> AAL : sendBuffer

Activate AAL note over AAL

Return immediately. endnote

AAL --> ODU : return status of API Note over AAL

Continue to process operations asynchronously

until all buffers sent. endnote

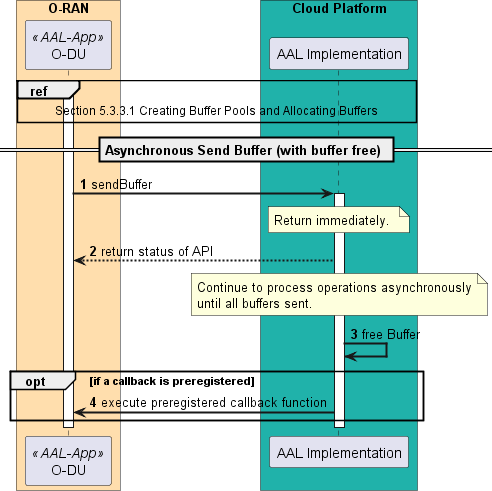
AAL -> AAL: free Buffer

opt if a callback is preregistered

AAL -> ODU: execute preregistered callback function End opt

deactivate AAL deactivate ODU

@enduml



##### Send Buffer Asynchronous Mode without buffer free

**Table 5.3.3.2.4-1 : Send Buffer Asynchronous Mode without buffer free workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | The AAL Application sends buffer(s) to the AAL for processing in asynchronous mode without buffer free. |  |
| Actors and Roles | AAL Application: formats buffer and its content, and calls AAL to send the buffer.  AAL Implementation: executes sendBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the sendBuffer operation in an asynchronous manner without buffer free enabled. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. Buffer(s) to be sent are allocated. |  |
| Begins when | The AAL Application is ready to send buffer. |  |
| Steps 1-2 (M) | The AAL Application calls the sendBuffer API to transfer the buffer to the AAL which will execute the sendBuffer operation. |  |

|  |  |  |
| --- | --- | --- |
|  | The AAL Application thread invoking the API is returned immediately with the status of the API call, without being blocked until the completion of sendBuffer operation.  The AAL continues to process the operation and sends the buffers. |  |
| Steps 3-4 (CM) | In case of without preregistered callback, the AAL Application, when finished with the buffer [1], calls freeBuffer operation. The AAL returns the status of the API call immediately and internally frees the buffer upon completion of sendBuffer operation. |  |
| Steps 5-7 (CM) | In case of with preregistered callback, the AAL returns the status of the sendBuffer operation back to the AAL Application, once done.  The AAL Application, when finished with the buffer [2], calls freeBuffer operation and the AAL will free the buffer back to the buffer pool. |  |
| Ends when | sendBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the sendBuffer API call fails. Status indicating failure will be returned in Step 2. |  |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

Note [1] : The AAL Application should not modify the content of the buffer(s) after step 2. Note [2] : The AAL Application can modify the content of the buffer(s) after step 5.

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App>> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Asynchronous Send Buffer (without buffer free) ==

ODU -> AAL : sendBuffer activate AAL

note over AAL return immediately. endnote

AAL --> ODU : return status of API Note over AAL

Continue to process operations asynchronously

until all buffers sent. endnote

alt without callback

ODU -> AAL: freeBuffer

AAL --> ODU : success/failure

Note over AAL

Internally buffer is freed when sendBuffer completes. endnote

else with preregistered callback

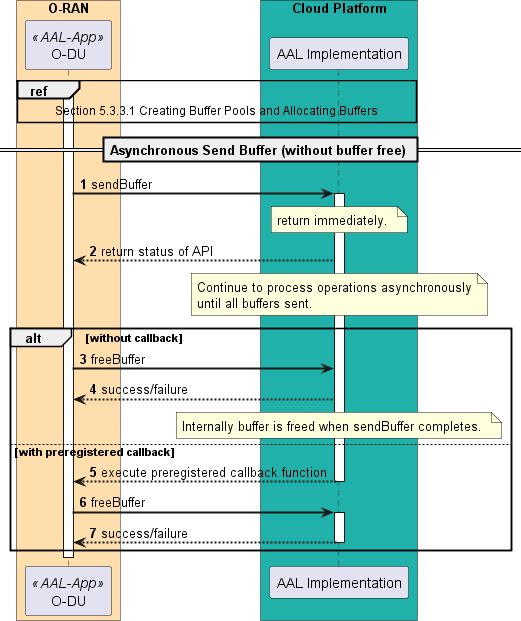
AAL --> ODU: execute preregistered callback function deactivate AAL

ODU -> AAL: freeBuffer activate AAL

AAL --> ODU : success/failure deactivate AAL

end

deactivate ODU @enduml



##### Receive Buffer Synchronous Mode with non-NULL buffer\_handle(s) input

**Table 5.3.3.2.5-1 : Receive Buffer Synchronous Mode with non-NULL buffer\_handle(s) input workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | The AAL Application (e.g., O-DU) receives buffer(s) from the AAL in synchronous mode with non-NULL buffer\_handle(s) input. |  |
| Actors and Roles | AAL Application: calls AAL to receive the buffer.  AAL Implementation: executes receiveBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the receiveBuffer operation in a synchronous manner with non-NULL buffer handle(s) input. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. Receive buffer(s) are allocated. |  |
| Begins when | The AAL Application is ready to receive buffer. |  |
| Steps 1 – 2 (M) | The AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receiveBuffer operation. The AAL Application thread invoking the API is blocked waiting for the receiveBuffer operation to complete.  Once done, the AAL returns the status of the operation back to the AAL Application and unblock the AAL Application call thread process. |  |
| Steps 3 - 4 (M) | The AAL Application, when finished with the buffer, calls freeBuffer operation and the AAL will free the buffer back to the buffer pool. |  |
| Ends when | receiveBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the receiveBuffer operation fails. Status indicating failure will be returned in Step 2. |  |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10 autoactivate on

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App>> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Synchronous Receive Buffer==

ODU -> AAL : receiveBuffer deactivate ODU

note over AAL

Perform operations and receive all buffers. Return when all buffers are sent.

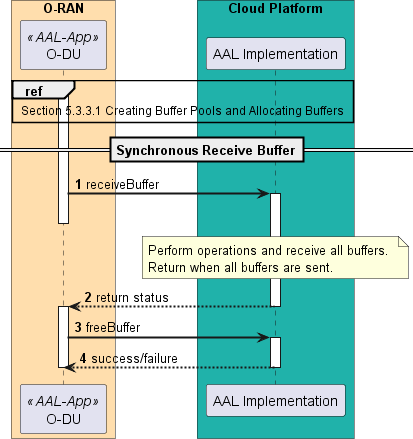
Endnote

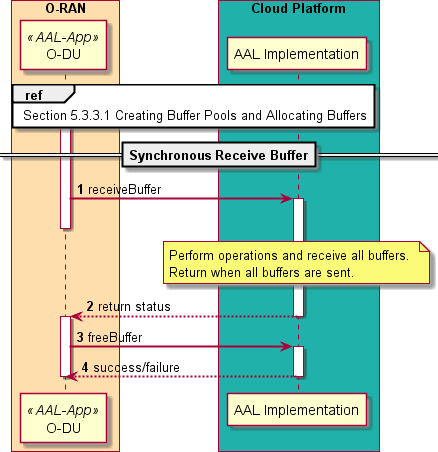
AAL --> ODU : return status activate ODU

ODU -> AAL: freeBuffer

AAL --> ODU : success/failure

deactivate ODU @enduml





##### Receive Buffer Synchronous Mode with NULL buffer\_handle(s) input

**Table 5.3.3.2.6-1: Receive Buffer Synchronous Mode with NULL buffer\_handle(s) input workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | The AAL Application (e.g., O-DU) receives buffer(s) from the AAL in synchronous mode with NULL buffer\_handle(s) input. |  |
| Actors and Roles | AAL Application: calls AAL to receive the buffer.  AAL Implementation: executes receiveBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the receiveBuffer operation in a synchronous manner with NULL buffer\_handle(s) input. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. |  |
| Begins when | The AAL Application is ready to receive buffer. |  |
| Steps 1 – 2 (M) | The AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receiveBuffer operation including the allocation of the buffers. The AAL Application thread invoking the API is blocked waiting for the receiveBuffer operation to complete.  Once done, the AAL returns the status of the operation back to the AAL Application and unblock the AAL Application call thread process. |  |
| Steps 3 - 4 (M) | The AAL Application, when finished with the buffer, calls freeBuffer operation and the AAL will free the buffer back to the buffer pool. |  |
| Ends when | receiveBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the receiveBuffer operation fails. Status indicating failure will be returned in Step 2. |  |

|  |  |  |
| --- | --- | --- |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10 autoactivate on

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App >> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Synchronous Receive Buffer==

ODU -> AAL : receiveBuffer deactivate ODU

AAL -> AAL : allocate buffer note over AAL

Perform operations and receive all buffers. Return when all buffers are sent.

endnote

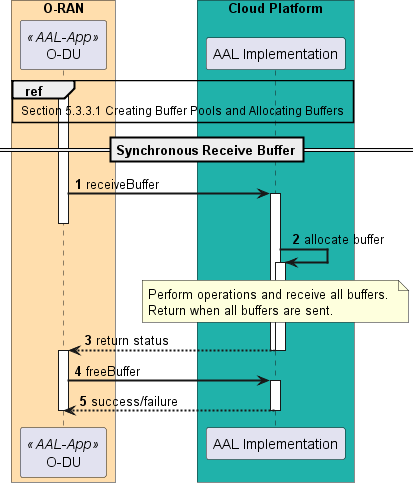
AAL --> ODU : return status deactivate AAL

activate ODU

ODU -> AAL: freeBuffer

AAL --> ODU : success/failure

deactivate ODU @enduml



##### Receive Buffer Asynchronous Mode with non-NULL buffer\_handle(s) input

**Table 5.3.3.2.7-1: Receive Buffer Asynchronous Mode with non-NULL buffer\_handle(s) input workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | The AAL Application (e.g., O-DU) receives buffer(s) from the AAL in asynchronous mode with non-NULL buffer\_handle(s) input. |  |
| Actors and Roles | AAL Application: calls AAL to receive the buffer.  AAL Implementation: executes receiveBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the receiveBuffer operation in an asynchronous manner with non-NULL buffer\_handle(s) input. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. Receive buffer(s) are allocated. |  |
| Begins when | The AAL Application is ready to receive buffer. |  |
| Step 1-4 (CM) | In case of without callback, the AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receiveBuffer |  |

|  |  |  |
| --- | --- | --- |
|  | operation. In the case no buffers are ready to be received the AAL Implementation returns immediately indicating no buffers were received. |  |
| The AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receiveBuffer operation. The AAL returns the status of the operation and the number of buffers received. |  |
| Steps 5-8 (CM) | In case of with preregistered callback, the AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receiveBuffer operation. In the case no buffers are ready to be received the AAL Implementation returns immediately indicating the status of the operation. The AAL Implementation waits asynchronously for buffers to be received. Upon receiving buffers, the pre-registered callback function is executed to asynchronously inform the AAL Application that new buffers have been received. |  |
| Steps 9-10 (M) | The AAL Application, when finished with the buffer, calls freeBuffer operation and the AAL will free the buffer back to the buffer pool. |  |
| Ends when | receiveBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the receiveBuffer operation fails. Status indicating failure will be returned in Step 1 or 5 or 8. |  |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10 autoactivate on

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App >> end box

Box “Cloud Platform” #lightseagreen participant “AAL Implementation” as AAL

End box activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Asynchronous Receive Buffer== alt without callback

ODU -> AAL : receiveBuffer deactivate ODU

note over AAL

If no buffers received return immediately.

In Asynchronous mode with no callback receive API behaves in a poll manner.

endnote

AAL --> ODU : (number of buffers = 0) activate ODU

ODU -> AAL : receiveBuffer deactivate ODU

AAL --> ODU : success (number of buffers) activate ODU

else with preregistered callback

ODU -> AAL : receiveBuffer Activate AAL

AAL --> ODU : return status

AAL -> AAL : wait for receive buffers

note over AAL

AAL executes receive operation. endnote

AAL --> ODU: execute preregistered callback function deactivate AAL

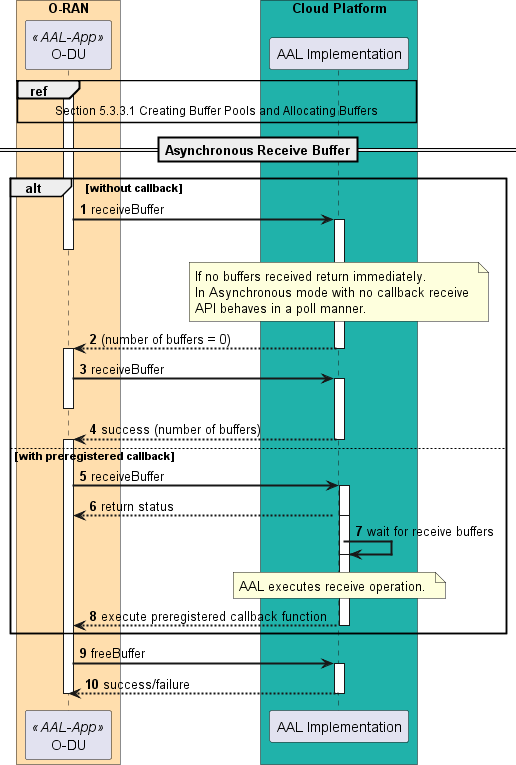
end

activate ODU

ODU -> AAL: freeBuffer

AAL --> ODU : success/failure

deactivate ODU @enduml



##### Receive Buffer Asynchronous Mode with NULL buffer\_handle input

**Table 5.3.3.2.8-1 : Receive Buffer Asynchronous Mode with NULL buffer\_handle input workflow steps**

|  |  |  |
| --- | --- | --- |
|  | **Evolution / Specification** | **<<Uses>>**  **Related use** |

|  |  |  |
| --- | --- | --- |
| Goal | The AAL Application (e.g., O-DU) receives buffer(s) from the AAL in asynchronous mode with NULL buffer\_handle input. |  |
| Actors and Roles | AAL Application: calls AAL to receive the buffer.  AAL Implementation: executes receiveBuffer operation. |  |
| Assumptions | This sequence is representative of how the AAL Application uses the receiveBuffer operation in an asynchronous manner with NULL buffer\_handle input. |  |
| Pre-conditions | At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started. |  |
| Begins when | The AAL Application is ready to receive buffer. |  |
| Steps 1-5 (CM) | In case of with callback, the AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receive. In the case no buffers are ready to be received the AAL Implementation returns immediately indicating no buffers were received.  The AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receiveBuffer operation including the allocation of the buffers. The AAL returns the status of the operation, newly allocated buffer\_hanlde(s) and the number of buffers received. |  |
| Steps 6-10 (CM) | In case of with preregistered callback, the AAL Application calls the receiveBuffer API to receive buffer content from the AAL which will execute the receiveBuffer operation. In the case no buffers are ready to be received the AAL Implementation returns immediately indicating the status of the operation.  The AAL Implementation waits asynchronously for buffers to be received. The AAL allocates the buffers. Upon receiving buffers, the pre-registered callback function is executed to asynchronously inform the AAL Application that new buffers have been received. |  |
| Steps 11-12 (M) | The AAL Application, when finished with the buffer, calls freeBuffer operation and the AAL will free the buffer back to the buffer pool. |  |
| Ends when | receiveBuffer operation has completed. |  |
| Exceptions | Exceptions can occur if the receiveBuffer operation fails. Status indicating failure will be returned in Step 1 or 5 or 8. |  |
| Post Conditions | The AAL Application can continue processing. |  |
| Traceability | To Be added later. |  |

@startuml Autonumber

Skinparam sequenceArrowThickness 2

skinparam ParticipantPadding 5

skinparam BoxPadding 10 autoactivate on

Box “O-RAN” #NavajoWhite

Participant “O-DU” as ODU << AAL-App>> end box

Box “Cloud Platform” #lightseagreen

participant “AAL Implementation” as AAL

End box

activate ODU

ref over ODU, AAL

Section 5.3.3.1 Creating Buffer Pools and Allocating Buffers end ref

==Asynchronous Receive Buffer==

alt without callback

ODU -> AAL : receiveBuffer deactivate ODU

note over AAL

If no buffers received return immediately.

In Asynchronous mode with no callback receive API behaves in a poll mode manner.

endnote

AAL --> ODU : (number of buffers = 0) activate ODU

ODU -> AAL : receiveBuffer deactivate ODU

AAL -> AAL : Allocate buffer(s)

AAL --> ODU : success (number of buffers) activate ODU

else with preregistered callback ODU -> AAL : receiveBuffer

AAL --> ODU : return status

AAL -> AAL : wait for receive buffers AAL -> AAL : Allocate buffer(s)

note over AAL

AAL executes receive operation. endnote

AAL --> ODU: execute preregistered callback function deactivate AAL

deactivate AAL

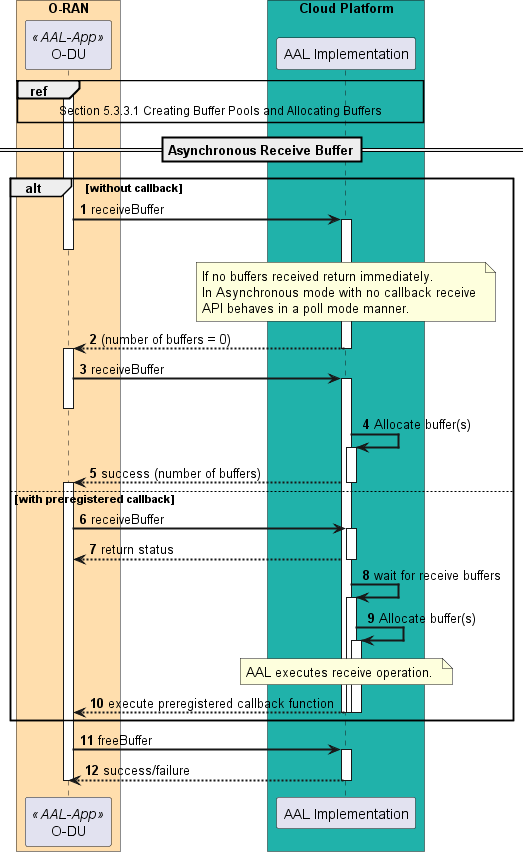
end

activate ODU

ODU -> AAL: freeBuffer

AAL --> ODU : success/failure

deactivate ODU @enduml



## AALI-C-App API details

### Capabilities discovery and configuration

#### initAal

Description: An AAL Application initializes the AAL Implementation. Pre-conditions: AAL Application is started.

Triggers: Executed once by the AAL Application Frequency: AAL Application Initialization time.

**Table 5.4.1.1-1: initAal operation**

|  |  |
| --- | --- |
| Operation | Direction |
| initAal | AAL Application => AAL |

**Table 5.4.1.1-2: initAal Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Parameter | Cardinality | Data Type | Qualifier | Description |
| lpu\_identifiers | 1..N | List Values | M | Optional list provided by DMS to the AAL Application that identifies the O-Cloud Resources and associated attributes that the DMS has assigned to and can be used by the AAL Application |
| vendor\_specific | 0..N | List Key/Value pairs |  | List of vendor specific attributes provided by the AAL Application. The AAL Application can obtain these parameters via the DMS or through another mechanism. |

**Table 5.4.1.1-3: initAal Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output Parameter | Cardinality | Data Type | Qualifier | Description |
| status\_of\_operatio n |  | Reference WG10 object model | M | Returns the status of the procedure |

#### cleanupAal

Description: This function must be called to allow the AAL implementation release any internal resources that the AAL implementation has allocated during initAal. After this call no other AALI-C-App API calls may be made.

Pre-conditions: initAal has been executed. Triggers: Executed once by the AAL Application Frequency: AAL Application termination time.

**Table 5.4.1.2-1: cleanupAal Operation**

|  |  |
| --- | --- |
| Operation | Direction |

|  |  |
| --- | --- |
| cleanupAal | AAL Application => AAL |

**Table 5.4.1.2-2: cleanupAal Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Parameter | Cardinality | Data Type | Qualifier | Description |
| NULL | 0 |  |  |  |

**Table 5.4.1.2-3: cleanupAal Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output Parameter | Cardinality | Data Type | Qualifier | Description |
| status\_of\_operation |  | Reference WG10 object model | M | Returns the status of the procedure |

#### getAalLpuInfo

Description: An AAL Application queries the handle and capabilities of the AAL-LPU(s) assigned to the AAL Application. AAL-LPUs may provide different capabilities such as different AAL Profiles can be supported by the AAL-LPUs that includes different types of AAL-Profile-Instances and AAL-Profile-Queue(s). The aal-lpu- handle is unique between the AAL Application and the AAL-LPUs assigned to the AAL Application. The aal- lpu-handle is generated by the AAL Implementation.

Pre-conditions: AAL Application has at least one AAL-LPU assigned and available for use. Triggers: On demand request from the AAL Application

Frequency: AAL Application Initialization time.

**Table 5.4.1.1-1: getAalLpuInfo operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalLpuInfo | AAL => AAL Implementation |

**Table 5.4.1.1-2: getAalLpuInfo Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_lpu\_handle | 0 to 1 |  | O | If supplied, the capabilities of only the AAL- LPU associated with the Handle is requested. If omitted, the capabilities of all the AAL-LPUs assigned to the AAL Application is requested. |

**Table 5.4.1.1-3: getAalLpuInfo Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal-lpu-cap-struct | 0 to many | struct | O | Struct containing the aal-lpu-handle and the capabilities supported by this AAL-LPU including AAL Profile type(s) that are available to the calling AAL Application. |
| status\_of\_operation |  | Reference WG10  object model | M | Returns the status of the procedure |

NOTE List of AAL Profile types to be defined, e.g. High-PHY. Types should also be versioned, e.g. High-PHY.V1

### AALI-C-App Configuration Services

Before using AAL-Profile-Instance services the AAL Application must first create the AAL-Profile-Instance, then configure the AAL-Profile-Instance and the AAL-Profile-Queue(s).

#### createAalProfileInstance

Description: The AAL Application creates an AAL-Profile-Instance on the AAL-LPU. The AAL-Profile-Instance is created based on the AAL Profile type that is requested by the AAL Application. The list of available AAL Profile type(s) is provided by the AALI in the getAalLpuInfo response. After successful creation, the AAL-Profile-Instance is in a ‘in use, stopped’ state. The AAL Application may now configure the AAL-Profile-Instance, AAL-Profile-Queue(s) and start the AAL-Profile-Instance.

This procedure may be repeated by the AAL Application in order to create multiple AAL-Profile-Instances, of the same or different AAL Profile types.

Pre-conditions: AAL Application has at least 1 AAL-LPU assigned and available for use. The AAL Profile type that is requested to be created is provided by the AAL-LPU (listed in the AAL-LPU-Capability struct of the getAalLpuInfo response operation).

Triggers: On demand request from the AAL Application Frequency: Infrequent.

**Table 5.4.2.1-1: createAalProfileInstance Operator**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| createAalProfileInstance | AAL Application => AAL Implementation |

**Table 5.4.2.1-2: createAalProfileInstance Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_lpu\_handle | 1 | N/A | M | The Handle of the AAL-LPU to create theAAL- Profile instance on. |
| aal\_profile\_instanc e\_type | 1 | tbd | M | The type (and version) of an AAL-Profile- Instance (returned in the getAalLpuInfo operation) to create on this AAL-LPU |

**Table 5.4.2.1-3: createAalProfileInstance Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| Status | 1 |  | M | Result of the createProfileInstance operation |
| aal\_profile\_instanc e\_handle | 1 |  | C | Handle of the AAL-Profile instance created in this create AAL-Profile-Instance Request.  This field is only present if the createProfileInstance operation is successful |
| aal\_profile\_instanc e\_config | 1 | Struct | C | Structure containing the default settings of the AAL-Profile-Instance . This field is only present if the createProfileInstance operation is successful. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| Status | 1 |  | M | Result of the createProfileInstance operation |
| status\_of\_operation |  | Reference WG10  object model | M | Returns the status of the procedure |

#### setAalProfileInstanceConfig

Description: The AAL Application configures the AAL-Profile-Instance on the AAL-LPU for a specific AAL Profile configuration for use by the AAL Application.

Pre-conditions: AAL Application has successfully created the AAL-LPU-Profile instance on the AAL-LPU. The AAL-Profile-Instance is not started.

Triggers: On demand request from the AAL Application Frequency: Infrequent.

**Table 5.4.2.2-1: setAalProfileInstanceConfig Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| setProfileInstanceConfig | AAL Application => AAL Implementation |

**Table 5.4.2.2-2: setAalProfileInstanceConfig Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_profile\_instanc e\_handle | 1 | N/A | M | The Handle of the AAL-Profile-Instance to be configured. |
| aal\_profile\_instanc e\_config | 1 | struct | M | Structure containing the AAL-Profile-Instance configuration to apply. |

**Table 5.4.2.2-3: setAalProfileInstanceConfig Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG10*  *object model* | M | Returns the status of the procedure |

#### deleteAalProfileInstance

Description: The AAL Application may delete the AAL-Profile-Instance when it no longer requires it. For example, if a cell is deleted from the O-DU by upper layers or the OAM system, the AAL-Profile-Instance resources used to offload that cell can be deleted. After deleting the AAL-Profile-Instance, that AAL-Profile-Instance is no longer usable by the AAL Application.

Pre-conditions: The AAL Application has successfully stopped the AAL-Profile-Instance. Triggers: On demand request from the AAL Application

Frequency: Infrequent.

**Table 5.4.2.3-1: deleteAalProfileInstance Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| deleteAalProfileInstance | AAL Application => AAL Implementation |

**Table 5.4.2.3-2: deleteAalProfileInstance Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_profile\_instanc e\_handle | 1 | N/A | M | The Handle of the AAL-Profile-Instance to delete. |

**Table 5.4.2.3-3: deleteAalProfileInstance Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG10*  *object model* | M | Returns the status of the procedure |

#### setAalProfileQueueConfig

NOTE: Use of AAL-Profile-Queue operations for user plane data transfer and buffer API operations needs to be clarified.

Description: The AAL Application configures the AAL-Profile-Queue on the AAL-Profile-Instance.

If no AAL-Profile-Queue Handle is included, this operation will create and configure a new AAL-Profile- Queue. The response will include the AAL-Profile-Queue Handle for the AAL-Profile-Queue. To add an AAL- Profile-Queue to an AAL-Profile-Instance, the AAL-Profile-Instance must be stopped by the AAL Application.

If an AAL-Profile-Queue Handle is supplied, the configuration of the AAL-Profile-Queue identified by the AAL-Profile-Queue Handle is modified with the supplied configuration. Any items already on the queue are deleted when this operation is called. The AAL-Profile-Queue must be in the stopped state.

Pre-conditions: AAL-Profile-Instance has already successfully been created in the AAL-LPU. The AAL Application has successfully configured the AAL-Profile-Instance. The AAL-Profile-Queue (if already existing) is in the stopped state. To add an AAL-Profile-Queue to an AAL-Profile-Instance, the AAL-Profile-Instance must be stopped by the AAL Application. To re-configure an existing AAL-Profile-Queue on an AAL-Profile-Instance, the AAL-Profile- Instance does not need to be stopped by the AAL Application.

Triggers: On demand request from the AAL Application Frequency: Infrequent.

**Table 5.4.2.4-1: setAalProfileQueueConfig Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| setAalProfileQueueConfig | AAL Application => AAL Implementation |

**Table 5.4.2.4-2: setAalProfileQueueConfig Input Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal-profile- instance-handle | **1** |  | **M** | The Handle of the AAL-Profile-Instance |
| aal\_profile\_queue- handle | 1 | N/A | O | If this field is included, the operation is for an existing AAL-Profile-Queue.  If this field is omitted, a new AAL-Profile- Queue is being configured. |
| aal\_profile\_queue- config | 1 | struct | M | Structure containing the AAL-Profile-Queue configuration to apply. E.g.  Direction (Input or Output);  Max number of entries permitted.  Queue priority (relative to the other queues of the same direction associated with this AAL- Profile-Instance) |

**Table 5.4.2.4-2: setAalProfileQueueConfig Output Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG10*  *object model* | M | Returns the status of the procedure |
| aal\_profile\_queue- handle | 0 to 1 |  | C | If no AAL-Profile-Queue Handle was included in the request, a new AAL-Profile-Queue Handle is returned if the operation was successful. |

#### getAalProfileQueueConfig

Description: The AAL Application may query the current AAL-Profile-Instance queue configuration.

Pre-conditions: AAL-Profile-Instance has already successfully been created in the AAL-LPU. The AAL Application has successfully created the AAL-Profile-Instance and the AAL-Profile-Queue.

Triggers: On demand request from the AAL Application Frequency: N/A

**Table 5.4.2.5-1: getAalProfileQueueConfig Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalProfileQueueConfig | AAL Application => AAL Implementation |

**Table 5.4.2.5-2: getAalProfileQueueConfig Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_profile\_queue- handle | 1 | N/A |  | The handle of the AAL-Profile-Queue to be queried. |

**Table 5.4.2.5-3: getAalProfileQueueConfig Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | *Reference WG10*  *object model* | M | Returns the status of the procedure |
| aal\_profile\_queue\_ config | 1 | structure | C | On success the struct that contains the AAL- Profile-Queue configuration. |

#### getAalProfileInstanceConfig

Description: The AAL Application may query the current configuration of the AAL-Profile-Instance.

Pre-conditions: AAL Application has successfully created the AAL-LPU-Profile instance on the AAL-LPU. Triggers: On demand request from the AAL Application.

Frequency: Infrequent.

**Table 5.4.2.2-1: getAalProfileInstanceConfig Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getAalProfileInstanceConfig | AAL Application => AAL Implementation |

**Table 5.4.2.2-2: getAalProfileInstanceConfig Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_profile\_instanc e\_handle | 1 | N/A | M | The handle of the AAL-Profile-Instance to be configured. |

**Table 5.4.2.2-3: getAalProfileInstanceConfig Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_profile\_instanc e\_config | 1 | struct | M | Structure containing the AAL-Profile-Instance configuration to apply. |
| status\_of\_operation |  | *Reference WG10*  *object model* | M | Returns the status of the procedure |

### AALI-C-App Transport Operations

#### createBufferPool

Description: The createBufferPool operation shall support creation of N buffer pools (N>=1) of a specific buffer type, with buffer read/write access to the AAL Application.

* + - * + The AAL Application shall have the flexibility to invoke the createBufferPool operation either one time, or multiple times, to request creation of buffer pool(s) with multiple buffer sizes.
        + The AAL Application shall invoke the createBufferPool operation multiple times for creation of buffer pools of different buffer types.
        + The AAL Application shall not assume buffers in created buffer pools to be in a specific memory state (e.g., zero ’ed buffer).
        + Buffer pool(s) and buffers are sharable by AAL-LPU(s).

FFS: support for inter-accelerator buffer pool sharing. Pre-conditions: AAL Implementation is initialized.

Triggers: On demand request from the AAL Application Frequency: No restriction

**Table 5.4.3.1-1: createBufferPool Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| createBufferPool | AAL Application => AAL Implementation |

**Table 5.4.3.1-2: createBufferPool Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| number\_of\_buffer\_pools | 1 | integer | M | Indicates the number of buffer pool(s) creation requested by the AAL Application |
| buffer\_type | N | integer | M | Indicates the type(s) of content that can be placed in buffers allocated from the buffer pool(s).  AAL-Profile instances can share buffer pools of the same buffer type(s)  The buffer type values are identified within the AAL-Profile specification.  An empty list indicates that the buffer(s) within the buffer pool(s) do not have an intended type of content that can be placed in the buffer pool(s). |
| buffer\_size | N | integer | M | Indicates the buffer size for each requested buffer pool  NOTE: |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
|  |  |  |  | * AAL Application shall indicate N buffer sizes, when requesting for creation of N buffer pool(s) (N>=1) in a single createBufferPool Operation |
| number\_of\_buffers | N | integer | M | Indicates the size of each buffer pool, i.e., number of buffers available in each pool upon creation  NOTE:   * AAL Application shall indicate N buffer pool sizes, when requesting for creation of N buffer pool(s), (N>=1) in   a single createBufferPool Operation |

**Table 5.4.3.1-3: createBufferPool Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure. Indicates the status of createBufferPool Operation execution (TRUE for success, FALSE for failure)  If FALSE: Returns failure code if the AAL Implementation does not support pre-created buffer pools. |
| buffer\_pool\_handle | N | opaque handle | M | For each buffer pool, a handle to the pool  NOTE: Each buffer\_pool\_handle is associated with a specific buffer\_size  A NULL shall be returned if the status\_of\_operation is FALSE (failure) |

#### allocBuffer

Description: The allocBuffer operation shall support allocation of a buffer either from pre-created buffer pool(s), or from memory that is not associated with a pre-created buffer pool.

Pre-conditions: If a non-NULL buffer\_pool\_handle is passed, the buffers are allocated from the buffer pool(s) created using the createBufferPool operation.

Triggers: On demand request from the AAL Application Frequency: No restriction

**Table 5.4.3.2-1: allocBuffer Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| allocBuffer | AAL Application => AAL Implementation |

**Table 5.4.3.2-2: allocBuffer Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| buffer\_pool\_handle | 1 | opaque handle | M | A handle to a pre-created buffer pool generated using createBufferPool operation or a NULL handle. When a NULL handle is passed, the AAL Implementation shall not use memory that is part of a pre-created buffer pool. |
| message\_size | 1 | integer | M | Indicates message size (in bytes) for which the AAL Application is requesting buffer allocation  NOTE:   * The AAL Application shall not use the allocated buffer for a message greater than the indicated message\_size. * Whether/how an AAL Implementation takes into consideration message\_size in allocating the buffer is transparent to the AAL Application.   When the value of the buffer\_pool\_handle is not NULL:   * The AAL Application shall request buffer allocation from a pre-created buffer pool with buffer\_ size >= message\_size > 0   + The AAL Application can indicate the message\_size equal to the buffer\_size associated with the buffer\_pool\_handle.   + For optimal memory usage, AAL Application should indicate a buffer\_pool\_handle with associated buffer\_size closest to the indicated memory size. |

**Table 5.4.3.2-3: allocBuffer Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure. Indicates the status of allocBuffer Operation execution (TRUE for success, FALSE for failure)  NOTE: If the status of the operation is FALSE:   * Returns a failure code if the AAL Implementation requires buffers to be allocated from a pre-created buffer pool and the operation was invoked with buffer\_pool\_handle = NULL. * Returns a failure code if the AAL Implementation does not support pre- created buffer pools and the operation was   invoked with a buffer\_pool\_handle that is not NULL. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
|  |  |  |  |  |
| buffer\_handle | 1 | opaque handle | M | Handle to the allocated buffer |

#### getBufferAddr

Description: The getBufferAddr operation shall return address from buffer\_handle Pre-conditions: The buffer\_handle is created using allocBuffer operation Triggers: On demand request from the AAL Application

Frequency: No restriction

**Table 5.4.3.3-1: getBufferAddr Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| getBufferAddr | AAL Application => AAL Implementation |

**Table 5.4.3.3-2: getBufferAddr Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| buffer\_handle | 1 | opaque handle | M | buffer object handle returned by allocBuffer operation |

**Table 5.4.3.3-3: getBufferAddr Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure. |
| buffer\_address | 1 | void \* | M | Pointer that points to the starting virtual address of the buffer for the AAL Application to use. On error, buffer address will be NULL value. |

#### getBufferSize

Description: The getBufferSize operation shall return the size of the buffer from buffer\_handle Pre-conditions: buffer\_handle is created using allocBuffer operation

Triggers: On demand request from the AAL Application Frequency: No restriction

**Table 5.4.3.4-1: getBufferSize Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |

|  |  |
| --- | --- |
| getBufferSize | AAL Application => AAL Implementation |

**Table 5.4.3.4-2: getBufferSize Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| buffer\_handle | 1 | opaque handle | M | buffer object handle returned by allocBuffer operation |

**Table 5.4.3.4-3: getBufferSize Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure.  Indicates the status of *getBufferSize* operation execution (TRUE for success, FALSE for failure) |
| buffer\_size | 1 | Integer | M | Size of the buffer |

#### sendBuffer

Description: The sendBuffer operation shall support enqueuing of N buffer(s) (N >=1) by the AAL Application. The AAL Application shall either retain buffer ownership in sendBuffer operation or indicate buffer ownership transfer. AAL Application shall indicate the synchronization mode ('synchronous' or 'asynchronous') in sendBuffer operation.

AAL Implementation shall support both modes of operation.

* + - * + For synchronous mode, the AAL Implementation, upon execution completion of sendBuffer operation, shall return the status of the sendBuffer operation
        + For asynchronous mode, AAL Implementation shall return the status of sendBuffer operation execution, only if AAL Application preregisters for a callback; otherwise AAL Implementation is not expected to return the execution status of the sendBuffer operation

FFS: Mechanism of sending details of AAL-Profile specific parameters

Pre-conditions: Buffers are allocated through allocBuffer operation. At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started

Triggers: On demand request from the AAL Application Frequency: No restriction

**Table 5.4.3.5-1: sendBuffer Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| sendBuffer | AAL Application => AAL Implementation |

**Table 5.4.3.5-2: sendBuffer Input Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| number\_of\_buffers | 1 | integer | M | Indicates number of enqueued buffer(s) (N >=1) |
| per\_buffer\_config | N | structure | M | Structure containing attributes corresponding to enqueued buffer(s). |
| > buffer\_handle | 1 | opaque handle | M | Indicates the buffer(s) containing metadata and payload to be sent. |
| > buffer\_free | 1 | integer | M | When “TRUE” or 1, the AAL Application transfers ownership of the buffer to the AAL Implementation, which is expected to free the buffer upon completion of the sendBuffer operation, for each enqueued buffer.  When "FALSE" or 0, the AAL Application retains the ownership of the buffer and the AAL Implementation shall not automatically free the buffer upon completion of the sendBuffer operation. |
| aal\_profile\_instance\_handle | 1 | opaque handle | M | Indicates the AAL-Profile-Instance corresponding to the buffer payload(s) |
| aal\_queue\_handle | 1 | opaque handle | M | Indicates the AAL-Profile-Queue associated with the enqueued buffer(s) |
| synchronization\_mode | 1 | integer | M | Indicates synchronous/asynchronous mode of operation |

**Table 5.4.3.5-3: sendBuffer Output Parameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10 object model* | M | Returns the status of the procedure.  Indicates the status of sendBuffer operation execution (success or failure) in synchronous mode and in asynchronous mode with preregistered callback  FFS: Details of callback preregistration mechanism |
| number\_of\_buffers\_tran smitted | 1 | integer | M | Indicates the total number of successfully transmitted buffers |
| status\_of\_each\_buffer | N | bool | O | Optionally, returns status of each buffer (success or failure) |

#### receiveBuffer

Description: The receiveBuffer operation shall support dequeuing up to N buffer(s) (N >=1) by the AAL Application. The AAL Application shall indicate the synchronization mode ('synchronous' or 'asynchronous') in the receiveBuffer operation.

AAL Implementation shall support both modes of the operation.

* + - * + For synchronous mode, the AAL Implementation, upon execution completion of receiveBuffer operation, shall return the status of the receiveBuffer operation
        + For asynchronous mode, the AAL Implementation shall return the status of receiveBuffer operation execution, only if AAL Application preregisters for a callback; otherwise AAL Implementation is not expected to return the execution status of the receiveBuffer operation

Either the AAL Application, or the AAL Implementation shall allocate the receiving buffer(s) either from buffer pool(s) pre-created through createBufferPool operation or from memory not mapped to a pre-created buffer pool.

Pre-conditions: At least one AAL-Profile-Instance is configured and started. At least one AAL-Profile-Queue is configured and started

Triggers: On demand request from the AAL Application Frequency: No restriction

**Table 5.4.3.6-1: receiveBuffer Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| receiveBuffer | AAL Application => AAL Implementation |

**Table 5.4.3.6-2: receiveBuffer Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| number\_of\_buffers | 1 | integer | M | Indicates maximum number of buffers (N >=1) that can be dequeued by the AAL Application |
| buffer\_handle | 0..N | opaque handle | M | Indicates dequeue buffer(s) allocated by the AAL Application.  If NULL, the AAL Implementation allocates the dequeue buffer(s)   * The AAL Application shall not use the dequeue buffer(s) allocated by the AAL Implementation for any other subsequent send or receive operation |
| aal\_profile\_instance\_handle | 1 | opaque handle | M | Indicates the AAL-Profile-Instance corresponding to the dequeued buffer payloads |
| aal\_queue\_handle | 1 | opaque handle | M | Indicates the AAL-Profile-Queue associated with the dequeued buffers |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| synchronization\_mode | 1 | integer | M | Indicates synchronous/asynchronous mode of operation |

**Table 5.4.3.6-3: receiveBuffer Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure.  Indicates the status of receiveBuffer operation execution (success or failure) in synchronous mode and in asynchronous mode with preregistered callback  FFS: details of callback preregistration mechanism |
| number\_of\_received buffers | 1 | integer | M | Indicates number of buffers (L) to be received by the AAL Application, which is < = maximum number of buffers (N) |
| buffer\_handle | L | opaque handle | M | Indicates dequeue buffer(s) where from AAL Application retrieves the received data. |
| buffer\_content\_size | L | integer | O | Optionally, indicates the size (in bytes) of the contents of each received buffer |

#### freeBuffer

Description: The freeBuffer operation shall free the buffer back to the buffer pool to which the buffer was allocated from.

* + - * + If the buffer is not freed in the sendBuffer operation (i.e., buffer\_free parameter is set to 0 or 'FALSE'), the freeBuffer operation shall free the buffer upon completion of the sendBuffer operation execution by the AAL Implementation.

Pre-conditions: The buffer\_handle is created using allocBuffer operation

Post-condition: Upon successful completion of the freeBuffer operation, the buffer\_handle is no longer usable by the AAL Application

Triggers: On demand request from the AAL Application Frequency: No restriction

**Table 5.4.3.7-1: freeBuffer Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| freeBuffer | AAL Application => AAL Implementation |

**Table 5.4.3.7-2: freeBuffer Input Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| buffer\_handle | 1 | opaque handle | M | buffer object handle returned by allocBuffer operation |

**Table 5.4.3.7-3: freeBuffer Output Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure.  Indicates the status of freeBuffer Operation execution (TRUE for success, FALSE for failure) |

#### destroyBufferPool

Description: The destroyBufferPool operation shall support termination of buffer pool, which will make buffer\_handle(s) associated with the buffer pool unusable by the AAL Application.

Pre-condition: The buffer\_pool\_handle is created using createBufferPool operation Trigger: On demand request from the AAL Application

Frequency: Infrequent

**Table 5.4.3.8-1: destroyBufferPool Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| destroyBufferPool | AAL Application => AAL Implementation |

**Table 5.4.3.8-2: destroyBufferPool Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| buffer\_pool \_handle | 1 | opaque handle | M | Indicates handle to pre-created buffer pool |

**Table 5.4.3.8-3: destroyBufferPool Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation | 1 | *Reference WG10*  *object model* | M | Returns the status of the procedure.  Indicates the status of destroyBufferPool Operation execution (success or failure) |

### AALI-C-App Administrative Operations

#### startAalProfileInstance

Description: The AAL Application starts the AAL-Profile-Instance once it is ready to start operations.

Pre-conditions: AAL-LPU is started, AAL Application is successfully configured, AAL-Profile-Instance is configured. The AAL-Profile-Instance is in the stopped state.

On success, the AAL-Profile-Instance will be in the started state. Triggers: On demand request from the AAL Application.

Frequency: Run time, infrequent.

**Table 5.4.4.1-1: startAalProfileInstance Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| startAalProfileInstance | AAL Application => AAL Implementation |

**Table 5.4.4.1-2: startAalProfileInstance Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_profile\_instanc e\_handle | 1 | UUID | M | The identifier of the AAL-Profile-Instance to start. |

**Table 5.4.4.1-3: startAalProfileInstance Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | Reference WG10  object model | M | Returns the status of the procedure |

#### stopAalProfileInstance

Description: The AAL Application stops the AAL-Profile-Instance.

Pre-conditions: AAL-LPU is started, AAL Application is configured and started, the AAL-Profile-Instance is configured and started.

On success, the AAL-Profile-Instance will be in the stopped state. Triggers: On demand request from the AAL Application.

Frequency: Run time, infrequent.

**Table 5.4.4.2-1: stopAalProfileInstance Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| stopAalProfileInstance | AAL Application => AAL Implementation |

**Table 5.4.4.2-2: stopAalProfileInstance Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal\_profile\_instanc e\_handle | 1 | UUID | M | The identifier of the AAL-Profile-Instance to stop. |

**Table 5.4.4.2-3: stopAalProfileInstance Output Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | Reference WG10  object model | M | Returns the status of the procedure |

#### startAalProfileQueue

Description: The AAL Application starts the AAL-Profile-Queue once it is ready to start operations.

Pre-conditions: AAL-LPU is started, AAL Application is successfully configured and started, AAL-Profile-Instance is configured and started, the AAL-Queue is configured and in stopped state.

Triggers: On demand request from the AAL Application. Frequency: Run time.

**Table 5.4.4.3-1 startAalprofileQueue Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| startAalProfileQueue | AAL Application => AAL Implementation |

**Table 5.4.4.3-2: startAalprofileQueue Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal-profile-queue- handle | 1 | N/A |  | The handle or identification of the AAL- Profile\_Queue to start. |

**Table 5.4.4.3-2: startAalprofileQueue Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | Reference WG10  object model | M | Returns the status of the procedure |

#### stopAalProfileQueue

Description: The AAL Application stops the AAL-Profile\_Queue.

Pre-conditions: AAL-LPU is started, AAL Application has successfully configured the AAL-Profile-Instance, AAL- Profile-Instance is configured and started, the AAL-Queue is configured and in started state.

Triggers: On demand request from the AAL Application Frequency: Run time

**Table 5.4.4.4-1 stopAalProfileQueue Operation**

|  |  |
| --- | --- |
| **Operation** | **Direction** |
| stopAalProfileQueue | AAL Application => AAL Implementation |

**Table 5.4.4.4-2: stopAalProfileQueue Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| aal-profile-queue- handle | 1 | N/A |  | The handle or identification of the AAL- Profile\_Queue to stop. |

**Table 5.4.4.4-2: stopAalprofileQueue Input Parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Output Parameter** | **Cardinality** | **Data Type** | **Qualifier** | **Description** |
| status\_of\_operation |  | Reference WG10  object model | M | Returns the status of the procedure |

# Annex A (normative): Acceleration Capability Requirements

## Capability requirements used in the placement of an AAL Application to an AF

Table A.1-1 contains a list of acceleration capability requirements that are common across AAL implementations. These capability requirements are used by the AAL Application to identify their requirements for use of AFs and by the SMO and/or DMS when assigning the AAL Application with a selected AF.

**Table A.1-1: Acceleration Capability Requirements**

|  |  |
| --- | --- |
| **Capability**  **requirement name** | **Data Type/Description** |
| aalProfileRequirem ent | **Data Type:** AalProfileRequirement  **Description:** The capability requirement for expressing what AAL-Profile is applicable when assigning an AAL-Profile to an AAL-LPU. |
| hwAcceleratorReq uirement | **Data Type:** HwAccelRequirement  **Description:** The capability requirement for expressing what HW Accelerator is applicable when assigning an AAL-Profile to an AAL-LPU. |
| aalLpuRequiremen t | **Data Type:** AalLpuRequirement  **Description:** The capability requirements applicable when assigning an AAL-Profile to an AAL-LPU. |

**Table A.1-2: Attributes for AalProfileRequirement**

|  |  |
| --- | --- |
| **Attribute name** | **Data Type/Description** |
| name | **Data Type:** string  **Description:** The name of the AAL-Profile as defined in the corresponding AAL-Profile specification. |
| supportedVersions | **Data Type:** Array of string  **Description:** The versions of the AAL-Profile that the AAL Application can use. The version syntax is consistent with the version of the corresponding AAL-Profile specification. |
| preferredVersion | **Data Type:** string |

|  |  |
| --- | --- |
| **Attribute name** | **Data Type/Description** |
|  | **Description:** The preferred version of the AAL-Profile that the AAL Application can use. The version syntax is consistent with the version of the corresponding AAL-Profile specification. |

**Table A.1-3: Attributes for HwAccelRequirement**

|  |  |
| --- | --- |
| **Attribute name** | **Data Type/Description** |
| vendorName | **Data Type:** string  **Description:** The HW Accelerator’s vendor that is needed to support the AAL Application. |
| model | **Data Type:** string  **Description:** The model of the HW Accelerator that is needed to spport the AAL Application. |
| hwVersion | **Data Type:** string  **Description:** hardware version of the HW Accelerator that is needed to spport the AAL Application. |

**Table A.1-4: Attributes for AalLpuRequirement**

|  |  |
| --- | --- |
| **Attribute name** | **Data Type/Description** |
| aalIDriver | **Data Type:** string  **Description:** The name of the AAL Implementation driver that is needed to support the AAL Application. |

Annex (informative): Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Author | Description |
| 2022.2.28 | 00.00.01 | VMware, Ericsson, Wind River, Nvidia, Marvell, AT&T, Orange, Qualcomm | Initial skeleton |
| 2022.3.18 | 00.00.02 | VMware | Incorporated 2 CRs:  Management Objects Model CR (VMWare) Common API Requirements CR (VMWare) |
| 2022.3.25 | 00.00.03 | NVIDIA, Intel, VMware | Incorporated 4 CRs:  Transport abstraction (NVIDIA) AALI-C-App APIs (Intel)  AALI-C-Mgmt Workflow CR (VMWare)  AALI-C-Mgmt APIs CR (VMWare) |
| 2022.3.28 | 01.00 | Padma Sudarsan VMware | Spec for final approval |
| 2022.5.12 | 01.00.01 | VMware | Incorporate CR:  Align AAL Definitions and Acronyms (Ericsson) |
| 2022.6.10 | 01.00.02 | Padma Sudarsan VMware | Incorporate 7 CRs  AAL Transport Buffer Operations (Marvell) AAL Transport API Operations (Nvidia)  AAL Transport Abstraction Requirements (Nvidia) AAL-LPU start/Stop (Ericsson)  AAL Common API Spec Alignment (VMware) AAL Get Accel Info part 1 (VMware)  AAL Common API e2e lcm flow (VMware et al.) |
| 2022.7.15 | 01.00.03 | Padma Sudarsan VMware | Editor changes: |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Modified cover page, footer, copyright, and Annex Z per new template  Incorporate CRs:  AAL Common API AAL Application Operations (Ericsson)  Editor Changes:  Modify Naming Conventions for Operations/parameters Identify sections that are still in progress |
| 2022.7.20 | 01.00.04 | Padma Sudarsan VMware | Incorporate CRs:  AAL Get Accel Info part 2 (VMware et al)  AAL Common API e2e lcm flow enhancement (Intel, VMware.) AAL App Init Sequence alignment (VMware)  Buffer Pool Create Operation (Nvidia et al) Buffer Send Operation (Nvidia et al))  Buffer Receive buffer pool destroy operation (Nvidia et al)  Editor Changes:  Modify Naming Conventions for Operations/parameters etc. |
| 2022.7.21 | 01.00.05 | Padma Sudarsan VMware | Incorporate CRs:  AAL-C-Transport Call flow (Nvidia et al) |
| 2022.7.25 | 01.00.06 | Padma Sudarsan VMware | Incorporate CRs:  AAL-C-Transport Call flow (Nvidia et al) update Editorial Cleanup |
| 2022.11.07 | 02.00.01 | Padma Sudarsan VMware | Incorporate CRs:  AAL Transport API-freeBuffer clarification (Nvidia)  Transport Req. alignment with receiveBuffer and freeBuffer (Nvidia) Transport receives buffer Operation clarification (Nvidia)  e2e lcm flow updates (VMware)  Correct UML in AALI-C-App API Init Sequence (Ericsson) Add Detailed API Diagrams (Intel, Nvidia)  Update allocBuffer (Qualcomm) Update createBufferPool (Qualcomm) |

|  |  |  |  |
| --- | --- | --- | --- |
| 2022.11.07 | 02.00.02 | Padma Sudarsan  VMware | Editorial corrections based on comments from the member companies. |
| 2022.12.05 | 03.00.01 | Padma Sudarsan VMware | Incorporate Editorial corrections.  Editorial alignment of terminology (Qualcomm)  Fault management service with AALI-C-Mgmt (Qualcomm, VMware, Ericsson) |
| 2023.1.11 | 03.00.02 | Padma Sudarsan  VMware | Incorporate Editorial corrections |
| 2023.2.1 | 03.00.03 | Padma Sudarsan VMware | Incorporate CRs:  Common API Mgmt Requirement Clarification (Ericsson) AAL-AALI-C-MGMT uploadBootImage Removal (Ericsson) AAL-Common-API-03.00-Add-Inv-Sub-Not (Qualcomm) |
| 2023.3.11 | 03.00.04 | Padma Sudarsan VMware | Incorporate CRs:  CR 0027 AAL Common API Updates from F2F (Qualcomm) Incorporate Editorial corrections |
| 2023.3.17 | 03.00.05 | Padma Sudarsan  VMware | Incorporate Editorial corrections |
| 2023.4.18 | 04.00.01 | Padma Sudarsan  VMware | Incorporate Editorial corrections |
| 2023.6.20 | 04.00.02 | Padma Sudarsan VMware | Incorporated the following CRs:  QCM: Update description in setAalAccelConfig and vendor specific attribute in set and get  INT: Fix missing AALI-C-App Init API  INT: AAL Common: Fix and update AALI-C-App State diagram QCM: AAL Common: AAL IM Diagrams  QCM: AAL Common: AAL IM Management Definitions QCM: AAL Common: AAL IM App Definitions  VMW: AAL Common: Changing Arch reference |
| 2023.6.28 | 04.00.03 | Padma Sudarsan  VMware | Incorporated the following CRs:  QCM: AAL Common Fix AAL IM Namespace |
| 2023.7.21 | 04.00.04 | Padma Sudarsan  VMware | Incorporated editorial comment |
| 2023.9.14 | 05.00.01 | Padma Sudarsan  VMware | Incorporated the following CRs: |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | NOK: Clarification on AALI-C-Mgmt Inventory Notification Parameters  WNDR: AALI-C-Mgmt HAM registration Incorporate Editorial comments |
| 2023.11.7 | 05.00.02 | Padma Sudarsan VMware | Incorporate the following CRs.  QCM: Remove reference to Alarm Dictionary QCM: Add HAM IM State  QCM: AAL Common Add Capabilities QCM: Add HAM IM  QCM: Fix DMS AAL Scenarios Requirement  Incorporate Editorial comments. |
| 2024-1-22 | 06.00.01 | Padma Sudarsan VMware by Broadcom | Incorporate the following CRs:  DCM-2023.10.24-WG6-  commonAPI CR001\_scope\_overview\_v03  QCM-2023.12.01-WG6-CR-0052-Common-API-Add-aalLpuConfig attribute to LPU  QCM-2023.12.11-WG6-CR-0055-Common-API-Add-LPU-HW-  Accel-Conditions  QCM-2023.12.11-WG6-CR-0058-Common-API-Remove-PM-  Dictionary  QCM-2023.12.21-WG6-CR-0049-Common-API-Add-Capability- Advertisement  QCM-2023.12.21-WG6-CR-0056-Common-API-Identify-Common- Requirements  QCM-2023.11.30-WG6-CR-0051-Common-API-  Remove\_OCloud\_Resource\_Ref |
| 2024-1-22 | 06.00.02 | Padma Sudarsan VMware by Broadcom | Incorporate the following CRs:  NOK-2023.10.11-WG6-CR0083-AAL-Common-HAM\_IM v00.06 DCM-2023.01.19-WG6-  commonAPI CR002\_HAM\_func\_requirements\_v06  DCM-2023.10.24-WG6-commonAPI CR003\_c-mgmt requirements\_v04  NOK-2023.12.11-WG6-CR-AAL Common API  Mgmt\_requirement\_v03  DCM-2024.01.22-WG6-commonAPI CR007\_transport\_abstraction framework\_v1  ERI-2024.1.18-WG6-CR-0044-AAL Common API UUID definition  and reference\_v3 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ERI-2024.1.23-WG6-CR-0045-AAL Common API URI reference  and acronym\_v4 |
| 2024-3-17 | 06.00.03 | Padma Sudarsan VMware by Broadcom | DCM-2023.10.24-WG6-commonAPI CR003\_c-mgmt requirements\_v05  ERI - 20240110-WG6-CR-0041-AAL Common-API - Removal of  AALI-C-App requirements for interaction with DMS v04  ERI-2024.1.18-WG6-CR-0043-AAL Common API AAL resource  identifier requirements\_v7 |
| 2024-3-19 | 06.00.04 | Padma Sudarsan VMware by Broadcom | Editorial updates |
| 2024-4-18 | 07.00.01 | Padma Sudarsan VMware by Broadcom | Incorporate CR  ERI-2024.04.05-WG6-CR-0047-AAL Common API Re-add AALI-  C-App Operations section accidentally removed in version 01.03 v2  Editorial changes |
| 2024-7-5 | 07.00.02 | Padma Sudarsan VMware by Broadcom | Incorporate CR  NOK-2024.04.08-WG6-CR0112-AAL\_CommonAPI-  EditorialRemovalOfDuplicateText  QCM-2024.04.10-WG6-AAL-Common-CR-0063-Update-IM-  Namespace  ERI-2024.04.05-WG6-CR-0048-AAL Common API Update ALI-C-  App State Diagram v4 |

History

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Description** |
| 2022.03.28 | 01.00 | Published from version 00.00.03 |
| 2022.08.01 | 02.00 | Published from version 01.00.06 |
| 2022.11.21 | 03.00 | Published from version 02.00.02 |
| 2023.03.27 | 04.00 | Published from version 03.00.05 |
| 2023.07.31 | 05.00 | Published from version 04.00.04 |
| 2023.11.27 | 06.00 | Published from version 05.00.02 |
| 2024.4.10 | 07.00 | Published from version 06.00.04 |
| 2024.7.31 | 08.00 | Published from version 07.00.02 |