Greenfield agreements

Complete

	Thread	Originator	Topics	Details	Agreement (from application Point of view)	Scope for Nov. train
1	1	AccelerComm (@ Barry Graham)	Separation of control and Data	Separation of control and data in the API architecture.	**Agreement:** *The AAL interface shall use separate user plane APIs and control plane APIs, with appropriate identifiers.* *FFS: details of control and user plane API identification for AAL*	Cat 1 CR @ Lopa Kundu to migrate all Cat 1 CR to GAnP document.
2	2	AccelerComm (@ Barry Graham)	Support Large payload size	Possibly worth capturing in the requirement is the vast range of payload sizes that will be encountered in both the DU and the CU	**Agreement**: *AAL API shall be flexible to support a range of payload sizes suitable for all required use cases.*	Cat 1 CR
3	3	NVIDIA (@ Lopa Kundu)	Abstract the API objects as C typedef objects handle instead of int ID	Not limit the API only to use *int* type as the handler(like BBDEV) to implement API in different hardware efficiently.	**Agreement:** *AAL API shall support generalized LPU abstraction instead of mandating a specific object type (For example, "C typedef object handle" versus "int type")*	Cat 2 CR
4	4	NVIDIA (@ Lopa Kundu)	Allow aggregation of multiple profiles in a single API invocation (Nvidia, Marvell)	This is to save cycles of IO communication. This scheme allows to combine multiple requests in one shot, a. k.a improve the the efficiency of submitting the job request which needs to cater in various profiles in a given LPU.	**Agreement:** *AAL API shall support aggregation of one or more than one AAL profile (s) and offload them to AAL implementation using a single enqueue API invocation.* *FFS: the details of dequeue or job status query mechanism of the aggregated AAL profile(s).* *As one example, for inline high-PHY acceleration, multiple AAL profiles (where an AAL profile refers to a PHY channel/signal for one or more than one cell(s) and one or more than one UE(s)) scheduled within a slot can be aggregated and offloaded to the AAL implementation by the application using a single enqueue API invocation.*	Cat 2 CR
5	5	Marvell (@ Jerin)	Allow flexible AF chaining (Marvell, Saankhya Labs, Nokia)		**Agreement:** 1. The AAL API shall provide support for the discovery, configuration, and invocation of Acceleration Profiles. Support for at least one Profile is mandatory for an acceleration implementation. 2. AAL API shall provide support for the capability to discover, configure, and invoke individual AFs when supported by the acceleration implementation. Support for individual AFs outside of Acceleration Profiles is optional. 3. The AAL API will not preclude the addition of support for the configuration and invocation (e. g., chaining) of multiple AFs. FFS: Details on AF API, discovery, configuration (including chaining of AFs), and invocation outside of the context of Acceleration Profiles.	Cat 2 CR

6	6	Arm(Honnappa)	ABI compatibility (binary portability) with in a CPU architecture across all types of accelerators (GPU, FPGA, ASIC) from all participating vendors should be provided with best possible performance. (Arm, Marvell)	• Opaque handles (void pointers) can be used for various control (slow /configuration) path objects. • For objects that are shared between slow path and fast path opaque handles can still be used. However, there might be a need to maintain common information (information such as enqueue /dequeue function pointers that are private to the implementation) at a fixed offset across all the implementations. • For objects on the fast path (ex: bit stream or pkt object) opaque handles is not helpful as fast access to the fields is required. A set of meta data common across all implementations might be needed and exposed to the application as a well defined structure.	**Latest Version:** *AAL API shall consider the aspects that matter for the flexibility of AAL implementation. AAL API shall not mandate any constraint that can potentially restrict such flexibility.* *For example, some of the aspects that matter for the flexibility of AAL implementation are* 1. *Handles for various objects* 2. *The presentation of the meta data to the application* *FFS: any other aspects of AAL implementation*	
7	7	Marvell (@Jerin)	AAL API shall support different transport mechanisms(Marvell, Nokia)	The split 6 transport can be based on shared memory, as PCI device , or as Ethernet. The common/profile API should not preclude to use any of the transport mechanism. To support various data source/sink like ethernet, CPU, PCI without reinventing the API for each LPU, Introduce the port concept to LPU, Where API the provider can define the different data/sink supported for the given LPU and consumer can pick a set of data sink/source for data movement. This will enable the reuse of the APIs for the remaining part of the LPU like RNF Configuration APIs.	**Agreement**: *AAL API shall support abstraction of the different transport mechanisms between the 'Application' and 'AAL Implementation.* *FFS: details of how to abstract the difference in transport mechanisms at the API level.*	Cat 1 CR

8	8	Ericsson (@ Christopher Richards)	Device shall provide applications a set of Common APIs for the profile(s) supported (Ericsson, Marvell)	Primary candidates for the common APIs are memory and profile management, including: initialization, get capabilities/version, device state, device configuration; start, stop, reset of resources and device, resource and device counters including performance monitoring metrics, events, etc. The common APIs shall be supported by all HW accelerator implementations.	**Agreement:** *AAL API shall define a set of Common APIs between 'Application(s)' and 'AAL implementation(s)'* *FFS: Details of common API(s).* *FFS: Candidate common API(s).* *Below are a few examples of potential candidate common API(s) for further discussion.* *NOTE: The list below is neither comprehensive, nor indicative of any agreed set of common API(s).* 1. *Initialization,* 2. *Get capabilities/version,* 3. *LPU state,* 4. *LPU configuration; start, stop, reset of LPU and other resources,* 5. *Resource and LPU counters including performance measurements /indicators, performance monitoring metrics, events, faults etc.* 6. *Profile management API (e.g. profile discovery, profile enumeration etc.)*	Cat 2 CR
9	9	Marvell (@Jerin)	AAL API naming convention must be friendly with other mainstream data plane projects such as DPDK (Marvell)	If a vendor or community decides to have an implementation of AAL API in other opensource projects such as DPDK, then it needs to follow specific naming conventions for API. Typically it will be: <Implemention_Prefix>< *Subsystem Name*><Sub subsystem name>..<verb/action> example: aal_rpf_xxxxx_set() Where *aal* is the implementation prefix, it will be *rte* for DPDK or *odp* for ODP implementation, The subsystem should be unique to the AAL project. A generic name such as, LPU, AF should be no go for upstreamablity as it will conflict with upstream projects functions. Suggested subsystem prefix *rpu* -> Radio processing unit for existing LPU *rpf* -> Radio profile *rnf* -> Radio network function.	**Agreement:** *AAL API shall have a unique name space.* *FFS: The details of unique name space .*	Cat 1 CR

10	10	NVIDIA (@ Lopa Kundu)	Asynchronous dequeue /status query (NVIDIA, Marvell)	Application should have the flexibility to asynchronously dequeue/query the processing status of enqueued workloads, i.e. enqueue and dequeue (or status query) invocations does not necessarily have to back-to-back. Application can invoke multiple enqueues before invoking the first dequeue (or status query) and application does not have to follow the same sequence of enqueue in the dequeue /status query invocation.	**Agreement:** *AAL API shall support asynchronous 'status query', i.e. query of job request status of the AAL profile(s) enqueued to the AAL implementation by the application.* Note 1: In case of uplink processing, 'status query' may be bundled with a dequeue request for processed data. Note 2: In general, 'status query' and 'dequeue' request corresponding to multiple enqueue requests can be bundled. FFS: Details of bundling the request and semantics of asynchronous operations.	Cat 2 CR
11	11	NVIDIA (@ Lopa Kundu)	Cell level and slot level configurations for L1 high PHY acceleration (NVIDIA, Marvell)	Having cell level and slot level parameters configured using separate API calls has the advantage of overhead reduction, since static parameters (cell-specific) change much less frequently than dynamic (PHY channel specific, on a slot basis) parameters, and doesn't need to be passed in every enqueue invocation.	**Agreement:** *For inline, L1 high PHY acceleration, AAL API shall support configuration of "cell specific" (i.e. static or semi-static) and "slot specific" (i.e. dynamic) parameters to AAL implementation using separate API functions* *Note: Configuration can include both control and user planes*	Cat 1 CR
12	13	Saankhya Labs (@ Sandeep Pendharkar)	Facilitate portability of AAL profile implementations from one DU hardware platform to another (Saankhya Labs, Vodafone)	Source code of AAL profile implemented for one O-Cloud platform consisting of some h.w. accelerator - say "X" should be portable to another O-Cloud platform consisting of some h.w. accelerator "Y" supporting the same AAL profile One of the ways to possibly achieve this is by baselining the signal processing kernels. This could be viewed in conjunction with requirement #15 mentioned above.	@ Sandeep Pendharkar to work with @ Honnappa Nagarahalli on the possibility of merging with thread 6.	
13	14	Saankhya Lab (@ Sandeep Pendharkar)	Dynamic management of the underlying HW acceleration resources being used by an AAL profile. (Saankhya Labs, Vodafone)	As an example, consider a Cloud platform consisting of an inline h.w. accelerator. Further assume, that this inline accelerator has "n" processing elements. The application or SMO should be able to inform AAL by means of an API or as apart of arguments to APIs like init or configure about the number of processing elements (out of "n") that should be used by the AAL profile for one particular instantiation of the CNF	*Agreement:* *Open pending decision on thread #19*	

14	15	Saankhya Labs (@ Sandeep Pendharkar)	Resource Management based on the Hardware architecture of the DU platform (Saankhya Labs, Vodafone)		**Latest version:** *Hardware Acceleration Manager shall support discovery of platform resources (example: different types of processing elements) of the Hardware Accelerator and expose it to the SMO.* *Ref Document : ETSI GS NFV-IFA 019 NFV Acceleration Technologies; Acceleration Resource Management Interface Specification (Section 8.5)*	
15	16	Nokia (@ Jan Ignatius)	Govern the behaviour and performance of acceleration function offload, . (Nokia, Saankhya Lab, Marvell, Vodafone)	To better govern the behaviour and performance of acceleration function offload, the AAL API shall provide the ability to configure/reserve quality-of-service (QoS) per LPU and/or per AF. Once a QoS request is admitted, the appropriate resources shall be allocated to support the service accordingly.	**Agreement**: *AAL API shall provide the ability to configure/reserve quality-of-service (QoS) per AAL-LPU and/or per AAL Profile depending on the service needs and are controllable per operation (i.e., not just a global setting).* *For example - Configurable QoS parameters can be:* • *Best effort* • *Hard guarantee e.g., processing throughput & latency are bounded* • *Soft guarantee e.g., processing throughput & latency are statistically bounded.* *Notification of the success or failure to admit the request shall be supported. Once a QoS request is admitted, the hardware resources can be allocated accordingly.* Note: Exact list of QoS Parameters and identification of M/O/CM is FFS. Note: Configuration of QoS information of AF for FFS.	Cat 2 CR
16	17		SFN/Slot synchronization (Qualcomm)	The SFN/SL synchronization procedure is used to maintain a consistent SFN/SL value (with respect to a given OTA slot) between the application and HighPHY accelerator.	**Agreement:** *AAL API shall support SFN/Slot synchronization between the application and AAL implementation as required by the AAL profile(s).*	Cat 1 CR
17	19	NVIDIA (@ Lopa Kundu)	Interface Standardization around Hardware Accelerator Manager	There are three key interfaces around HW Accelerator Manager: 1. Southbound (**SB**): HW Accelerator Manager <-> HW Accelerator Device and/or HW Accelerator Manager <-> Library/Driver /User Space /Kernel Space 2. Northbound -1 (**NB-1**): HW Accelerator Manager <-> O-Cloud IMS/DMS 3. Northbound -2 (**NB-2**): O-Cloud IMS /DMS <-> SMO **Questions:** Q1. Should the interface "**SB**" be standardized? Q2. Where the "**NB**" interface should be standardized? At **NB-2** level and/or at **NB-1** level?	**Latest Version:** Table on wiki page.	

18	20	AT&T (@ Kaustubh Joshi (KJ))	Define the license of the AAL API header file	After the requirement phase(End of Aug 2021), AAL contributors will contribute to the AAL API definition header file. This email thread is initiated to collect feedback on deciding the AAL API header file license to kick start the API definition from Sep 1 for the Nov train.	**Agreement:** *O-RAN WG6 shall define a stage 2 AALi API for the Hi-Phy profile (Nov 2021 train MVP). Definition of a stage 3 API (header files) is FFS.* *The need for this agreement is deferred until stage 3 work commences. It is noted that O-RAN currently only allows publication of stage 3 aspects of its specifications using the SCCL license. Any changes to this policy to use more permissive licenses such as BSDv3 require board approval (and a strong rationale)*	Out of scope for Nov 2021 train
19	21	Qualcomm (@ Rudra Shivalingaiah)	Compatibility with O-RAN WG4 FH interface (Qualcomm)	HighPHY on accelerator communicates with O-RU over WG4 FH (7.2) interface. (Ref: Fig 2.9 WG6-AAL-GAnP document). Example usages of this interface: 1. To support O-RAN M-Plane's hierarchical model 2. O-RU capability exchange and configuration 3. Analog/Digital beamforming configuration and control	**Latest Version**: *AAL API shall be compatible with O-RAN WG4 FH interface and support communication between the application and O-RU via highPHY accelerator as required by AAL profile (s).*	Cat 1 CR

Company responses for Thread #19

company Name	Response to Q1:		Response to Q2:			
	Should "SB" interface be standardized?		**Where the "NB" interface should be standardized?**			
	(Y/N)	Comment	At **NB-1** level (Y/N)	At **NB-2** level (Y/N)	Comment	
Marvell	Y	# AAL abstracts the differences in HW acceleration devices # If SB is standardized, AAL implementer can focus on abstracting HW devices differences under _one_ library. That library can have given interface for AALI and other aspects required for IMS/DMS # AALI interface also would need some of the functionalities like discovery, get/set configuration. No pointing duplicating interface for same to avoid duplication of API interface and enable coherency of APIs and libraries. # To avoid responsibility for AAL implementer to write yet another plugin different deployment scenarios	Y	Y	Once SB is standardized, naturally, the interfaces above will be standardized.	

Ericsson	N	A strong “No”. There are huge differences between HW Accel vendors in architecture and function as well as supported features and capabilities. Trying to define a lowest common denominator set of device/driver management functions that will satisfy all will lead to (a) slowing of the AAL and API work (b) more vendor specific extensions and (c) potentially lower performance and capabilities. The entire purpose of the HW Accel manager is to translate O-Cloud O2 mgmt API actions into vendor / device specific operations. HW Accel vendors will always package a manager function with their HW (just as is done for the drivers) – they perform extensive integration and verification to ensure the proper operation of the device with the mgmt function before releasing a product. Who would be responsible for this in a multi-vendor scenario? Who is responsible for dealing with issues (failures, errors) in the field?	Y	Y	It’s not clear what these two interfaces would be in practice if both were standardized. Some concrete examples would help. Recommend that the WG go through the use cases first to get a firmer understanding of what NB-1 and NB-2 really are before deciding which one of them (or both) need to be standardized.
Nvidia	N	Aligned with Ericsson’s comment above and a strong “No” as well.	N	Y	NB-1 should be an internal interface to O-Cloud-platform and will depend on how HW Accelerator Manager is on-boarded/integrated by the O-Cloud Platform- it can be based on de-facto K8s API (for example, Nvidia GPU operator) or it can be custom API (e.g. Nvidia SMI). As long as NB-1 can consume O-cloud O2 mgmt. API (from O2-IM or O2-DM), SMO should not care how NB-1 is implemented within O-cloud platform.
Wind River	Y	An abstraction layer at this level will simplify the functionality carried by the accelerator manager.	N	Y	Agree with Nvidia’s comment above. The SMO demarcation point should be the IMS/DMS, anything beyond is internal to the cloud.
Nokia	N	HW Manager will require understanding hardware specificities and logic and therefore difficult to standardize. If the group agrees we should update the diagram to represent that box as the hardware manager.. not just hardware acceleration manager.	Y	Y	NB-2 (O2) is definitely being standardized and there is already quite a bit of discussion on the IMS and DMS northbound (O2) and southbound i.e. NB-1 and NB-2. Whether these are two explicit interfaces or converged would continue to be part of the O2 discussions. Depending on where those discussions go, the HW Manager northbound (NB-1) will have to comply to the inventory info model specified by the O2 sub group.
Qualcomm	Y	The following are the advantages • There are some APIs common across the SB and AAL interface • Will decouple the SB interface and HW Accelerator from a specific orchestration framework HW Accelerator manager interface is complex and need further discussion. This cannot complete by the November train and voting may not resolve it. Here are our suggestions • Decouple this requirement from the November train deliverables. • Add this requirement for further study. • Discuss within WG6 as the specific use cases of discovery, configuration, and orchestration. • Discuss and agree on device capabilities advertised by HW AAL • Discuss current frameworks and their capabilities	Y	Y	
SaankhyaLabs	Y	Agree with comments from Qualcomm. We feel this topic is important enough to merit a separate detailed discussion. Hence it should be decoupled from the November train deliverable.	Y	Y	
Docomo	N	• Standardizing internal communication through the SBI of the Accelerator Manager is not mandatory and could be vendor-specific. • NBI-1/NB-2 of Accelerator Manager will be the "boundaries" between O-Cloud and SMO.	Y	Y	• NBI is mandatory/essential to support multi-vendor operations /management. • We also agree with Ericsson, the overall picture, use case and functional requirement, i.e. SMO, O-Cloud, DU, NW, and AAL are not well defined, and AAL GA&P does not have this concept. Without this idea and concept, it is difficult to elaborate on the specifics of NB-1 and NB-2.

VMware	Y	Main reasons (applicable for a MV scenario): • Will decouple the southbound interface and HW Accelerator and any compute resources from a specific orchestration framework. • Consistency across different vendors and alignment on what is exposed via the O2 We should really modify the figure to reflect that it is not just the accelerator that contributes to the compute resources that will be exposed via the O2. In that case a MV situation seems more relevant.	Y	Y	NB1/NB2 is Essential for MV orchestration. The AAL sub-group needs to align with the O2 and O1 sub group to provide input/requirements on info model w.r.t infrastructure, allocation of resources and orchestration of workloads on these resources.	
Orange	N	In our view, the SBI of the HW Accelerator Manager could stay vendor-specific.	Y	Y	• NB-2 is currently being standardized within O2 interface discussions. • As for the NB-1, we would like to see the HW Accelerator Manager software decoupled from other O-Cloud Platform software pieces (such as IMS/DMS). OTOH, the WG6 can at least provide some “reference design” for this decoupling, e.g., based on de-facto K8s operators’ framework…	

_·_,_·_,_