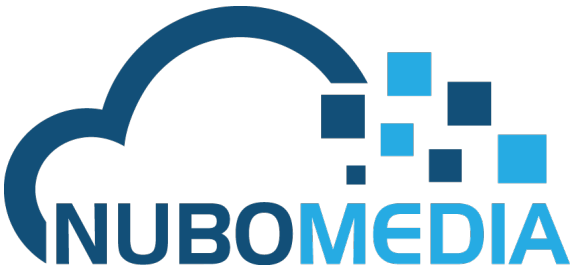
D2.4.1	
Version	1.0
Author	URJC
Dissemination	PU
Date	27/01/2015
Status	Final



D2.4.1: NUBOMEDIA Architecture v1

Project acronym:	NUBOMEDIA
Project title:	NUBOMEDIA: an elastic Platform as a Service (PaaS) cloud for interactive social multimedia
Project duration:	2014-02-01 to 2016-09-30
Project type:	STREP
Project reference:	610576
Project web page:	http://www.nubomedia.eu
Work package	WP2: User centric analysis of user scenarios and requirements
WP leader	VTOOLS
Deliverable nature:	Report
Lead editor:	Luis López Fernández
Planned delivery date	31/01/2015
Actual delivery date	27/01/2015
Keywords	Architecture

The research leading to these results has been funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement nº 610576

FP7 ICT-2013.1.6. Connected and Social Media

Contributors:

Luis López Fernández (URJC)
Giuseppe Carella (TUB)
Javier López Fernández (NAEVATEC)
Alin Calinciuc (USV)

Internal Reviewer(s):

Giuseppe Carella (TUB)

Version History

Version	Date	Authors	Comments
0.1	28/07/14	Luis López Fernández	First initial version
0.2	8/10/14	Luis López	Cloud architecture as functional architecture.
0.3	3/01/15	Luis Lopez	Mayor modifications to architecture in order to simplify it and to adapt to a more flat model where all the logic of distributed media pipeline management is left to applications.
1.0	19/01/15	Luis Lopez	Added CFD descriptions for TUB. Added new reference points descriptions.

Table of contents

List of Figures:

Acronyms and abbreviations:

API	Application Programming Interface
AS	Application Server (refers to function instance)
ASS	Application Server Services
CFM	Cloud Functions Manager
CPU	Central Processing Unit
DAS	Distributed Application Server (refers to function class)
DMP	Distributed Media Pipeline
DMS	Distributed Media Server (refers to function class)
DoD	Denial of Service
DSFE	Distributed Signaling Front End (refers to function class)
IaaS	Infrastructure as a Service
JSON	JavaScript Object Notation
ME	Media Element
MP	Media Pipeline
MS	Media Server (refers to function instance)
PA	Platform Application
PaaS	Platform as a Service
PAL	Platform Application Logic
PAMC	Platform Application Media Control
RFC	Request For Comments
RTC	Real-Time Communications
SaaS	Software as a Service
SFE	Signaling Front End (refers to function instance)
UE	User Equipment
WS	WebSocket
WWW	World Wide Web
XMPP	eXtensive Message and Presence Protocol

1 Executive summary

This document contains a formal specification of the NUBOMEDIA architecture. This specification must be used for driving the technological developments of WP3, WP4 and WP5.

2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3 Definitions

Connection: A transport layer virtual circuit established between two programs for the purpose of communication. A connection may be reliable or unreliable
Message: Refers to the communication unit of an application layer protocol and consists on a sequence of octets matching a specific syntax as defined by the application layer protocol.
Request: A message asking for the execution of a specific semantics, as defined by the application layer protocol.
Response: A message providing the result of the execution of a request
Transaction: A pair of messages consisting of a request and its answer(s).
Client: A process that establishes connections for sending requests.
Infrastructure: A program or a set of programs acting in coordination providing semantics to request and issuing responses as result.
User Equipment: Refers to a end-user device acting as client for the infrastructure
Interface: Formal description of the protocol or primitives used among components to communicate.
Component: A black box abstracting the internal logic of a system which, through interfaces, exposes and consumes services.
Object: A software artifact instance implementing one or several components. In Object Oriented Programming, the software artifacts take the form of classes and object are class instances. However, we don't restrict to Object Oriented Programming models for this definition.
Function: Logical component with an interface based on the exchange of messages. Functions take one or several messages as input, execute the appropriate logic and issue one or several messages as output. Functions can be state-less or state-full.
Architecture: Specification of the main functions of a system and interactions among them and with the external world.
Reference Point: Interfaces and messaging capabilities found among two or more functions with direct interaction. In this context, direct interaction means that functions sharing a reference point must be capable of exchanging messages without requiring any third function to mediate among them. Remark that messaging transport and routing capabilities are not considered as functions.
Monolithic: When qualifying a function or component, indicates that it consists of a single computing resource. A monolithic function comprises one process where its internal components communicate through shared memory.
Distributed: When qualifying a function or component, indicates that it executes across a network. Hence, a distributed function comprises a number of monolithic processes communicating through a network.

Cloud: When qualifying a function or component, indicates that it has a role in the management of the cloud resources of the underlying cloud infrastructure.

4 Introduction

A functional architecture is understood as a high level abstraction showing components and interactions among them. Depending on the desired level of abstraction, and depending on the objective of the abstraction, we can find different types of architectures. For example, software engineers (who have the objective of designing and writing software) usually understand components as software modules (logical entities) or packages providing one or several functions and communicating with the rest through APIs. On the other hand, for a systems engineer, an architecture usually refers to one or more physical entities (routers, CPUs, disks, etc.) that interact through wired or wireless connectivity capabilities.

In this document we present the architecture of NUBOMEDIA as a number of components and interactions among them. Given that we pursuit multiple objectives, we also need multiple architectures presenting the same reality but from different perspectives. In particular, we concentrate on two different views:

- **Functional architecture**: This architecture presents NUBOMEDIA from the functional perspective, meaning that components are understood as functions in the media and signaling domains and their interactions takes place through interfaces associated to reference points. The objective of this architecture is to expose a list of features that NUBOMEDIA provides to applications. Hence, we present the functions that directly relate to them (media-related functions), abstracting the rest of functions not related with that objective. This architecture can also be related to the NUBOMEDIA runtime, in the sense that the different functions can be seen as processes (or group of processes) providing their capabilities and orchestrating their behavior through the exchange of messages. This architecture also presents NUBOMEDIA from a cloud perspective showing also cloud components as functions. Cloud functions may include mechanisms for managing virtual computing, networking, and storage resources, and scale them accordingly to user traffic.
- **Software architecture**: This architecture presents NUBOMEDIA from a software engineering perspective concentrating on how it can be implemented in terms of software artifacts and software APIs.

5 NUBOMEDIA functional architecture

5.1 Introduction

For having an intuitive image of what a functional architecture is, observe Figure 1 a). There the high level functional components of a WWW infrastructure are shown in what is usually called the three-tier WWW architecture. If we come to the Real-Time Communications (RTC) arena, the traditional functional components of media infrastructures are the ones depicted on Figure 1 b), which play roles symmetric to the ones of WWWs:

- Signaling: Exposes to end-user clients the capability of requesting access to the service and of negotiating the conditions in which the service needs to be provided. This usually takes place through some kind of protocol such as SIP, XMPP or JSON-over-WS.

- Application: provides the specific logic (semantics) of the application. This means that this component is in charge of executing actions when signaling messages are received so that these actions guarantee that the service is provided. This may involve instantiating media functions, querying media repositories, authenticating users, etc.
- Media server: provides media capabilities such as media transport, media transcoding, media recording, media dispatching, etc.

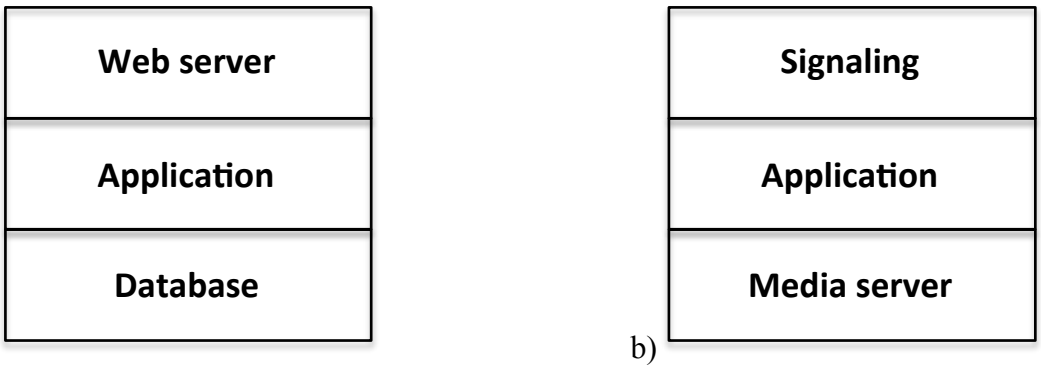


Figure 1

NUBOMEDIA inherits many aspects from the three-tier architectural pattern but some additional specificities need to be considered:

- NUBOMEDIA components are distributed objects. For enhancing this, we will qualify as "Distributed" all components and functions consisting of the aggregation of monolithic modules. In other words, a distributed function (starting with "D") will be materialized into many orchestrated instances of the function. Coming to an Object Oriented metaphor, the Distributed function represents the class and the instance the object.
- NUBOMEDIA is a PaaS (Platform as a Service) meaning that it needs to expose its media capabilities to external applications which could consume them through a PaaS API or SaaS services.

As a result, the top level functional architecture of NUBOMEDIA could be represented by the scheme depicted on Figure 2.

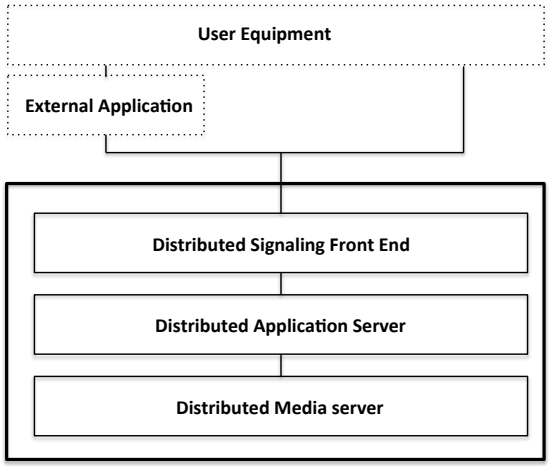


Figure 2: High-level functional architecture of the NUBOMEDIA infrastructure. This architecture responds to common architectural models of RTC systems splitting the main functions between signaling, application logic, and media features.

- User Equipment (UE): This component is not part of the NUBOMEDIA architecture and it's only added for completeness. It represents the end-user terminal. The UE may contain the application logic and consume directly

NUBOMEDIA PaaS API or may interact with an external application providing the application logic and consuming itself the NUBOMEDIA PaaS API.

- External Application: This component is not part of the NUBOMEDIA architecture and it's only added for completeness. It represents an application not hosted by NUBOMEDIA. Given that the dialogue with the UE is not mediated by NUBOMEDIA, the application may use any signaling mechanism without NUBOMEDIA being aware of it. However, the control of the media capabilities need to take place through NUBOMEDIA PaaS API.
- Distributed Signaling Front End: This distributed component is in charge of managing the signaling between UEs (and external applications) and NUBOMEDIA. This component exposes NUBOMEDIA PaaS API.
- Distributed Application Server: This distributed component is in charge of hosting NUBOMEDIA application logic.
- Distributed Media Server: This distributed component hosts the media functions required by applications.

5.2 NUBOMEDIA functional architecture

More formally, and extending the vision depicted in Figure 2, the NUBOMEDIA functional architecture is depicted in Figure 3. The description of each of the functions and reference points is provided in the sections below.

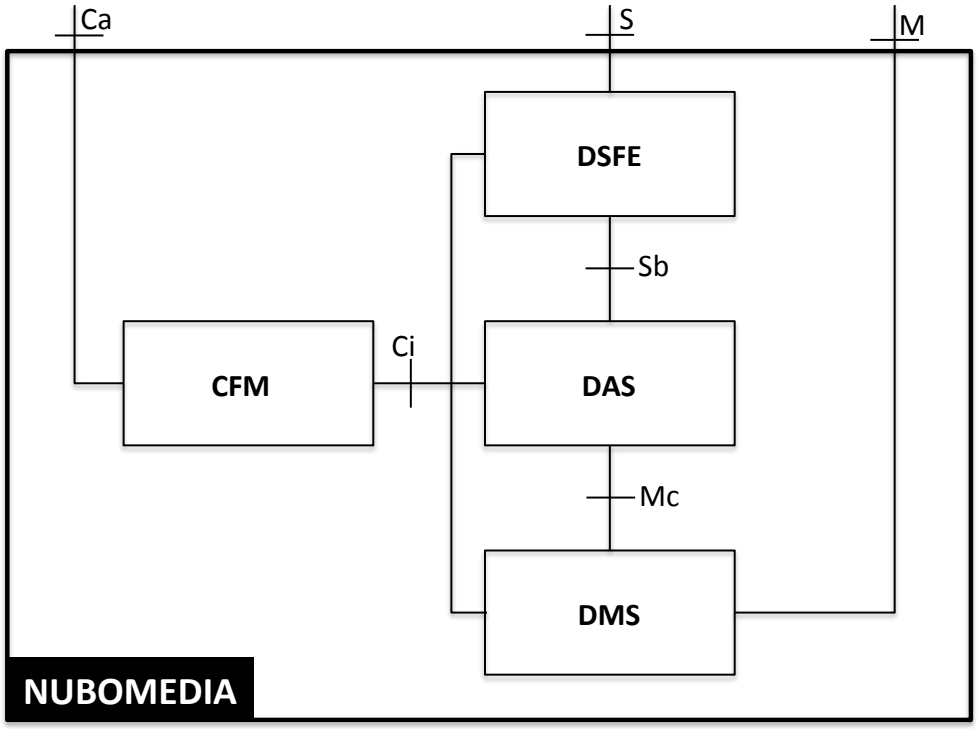


Figure 3. NUBOMEDIA high-level architecture identifying its main functions and reference points. Among the former (from left to right and from top to bottom): Cloud Functions Manager (CFM), Distributed Signaling Front-End (DSFE), Distributed Application Server (DAS) and Distributed Media Server (DMS). Among the later: Cloud Admin (Ca), Could interface (Ci), Signaling (S), Signaling broker (Sb), Media control (Mc), Media (M).

5.3 Functions

5.3.1 Distributed Signaling Front End (DSFE)

This function must take care of the exchange of signaling messages between NUBOMEDIA hosted applications and UEs. Signaling messages must comply with a

protocol supported by the architecture on reference point S. The DSFE also acts as a gateway between the UE (i.e. the external application) and the DAS function. At runtime, this distributed function is materialized as N monolithic SFE instance processes.

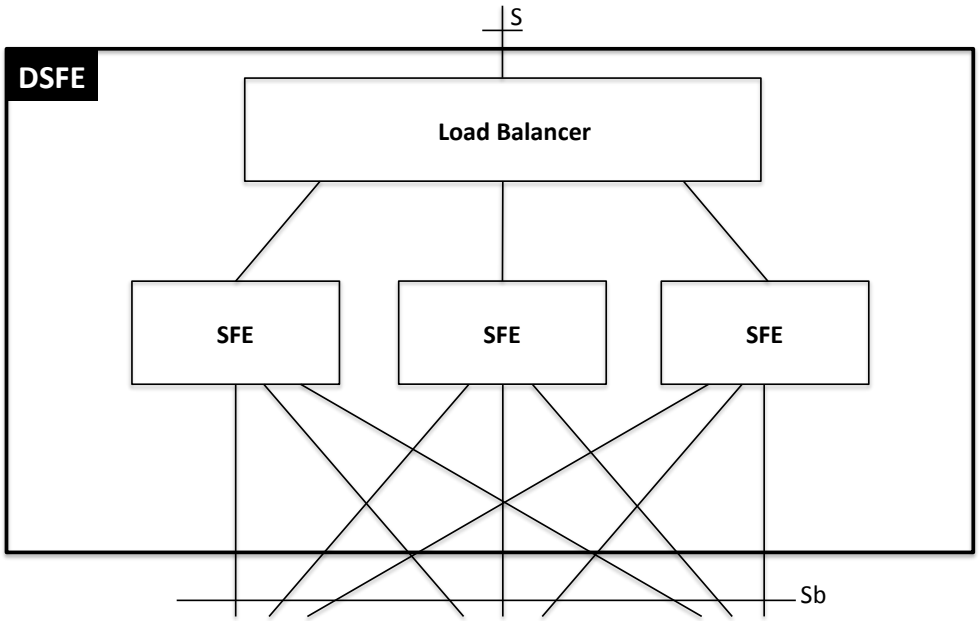


Figure 4. Example of materialization of the Distributed Signaling Front End (DSFE) function as orchestrated distributed monolithic functions (processes). At the northbound, the DSFE interacts with UA (i.e. external applications) through S interfaces. At the southbound the DSFE interacts with the DAS function through Sb interfaces.

DSFE functions are:

- At S reference point, it must receive signaling messages from the external world (e.g. JSON-RPC WebSocket endpoint, SIP UDP endpoint, REST endpoint, etc.) For this, DSFE uses endpoints, whose characteristics depend on the supported signaling protocol. Endpoints must be reachable from clients either directly (public network address) or indirectly (through proxy or load balancer).
- At S reference point, this function may provide load balancing capabilities for distributing client messages and connections among the different SFE instances hosting endpoints. In case of stateful SFEs the signaling protocol must provide some kind of session identification (i.e. sessionId) enabling unique mappings between sessions and SFE instances. This mapping is commonly known in the literature as the “sticky session” mapping. The load balancing capability at S reference point may be based on a DNS mechanism or on a proxy mechanism. In any case, this load balancing may be assisted by the CFM, through the Ci reference point, for the placement of new incoming session into the most appropriate SFE instance.
- At S reference point, this function may provide security mechanism for the contention and avoidance of DoS attacks.
- When a signaling message is received at a SFE instance it is translated to the specific protocol required by the DAS. This translation process may involve the (un)marshaling of messages and the checking of their compliance with signaling protocol specification and security rules. The generated messages are delivered to the DAS through the Sb reference point.
- At the Sb reference point a load balancing capability toward the DAS layer, selecting the most appropriate AS instance for executing a given application request, may be provided. AS instances may be stateful. In this case, the signaling protocol must provide some kind of session identification (i.e.

sessionId) enabling unique mappings between sessions and AS instances. This mapping is again based on sticky sessions. This load balancing capability may assisted by the CFM through the Ci reference point for the placement of new incoming sessions into the most appropriate AS instance.

- At Sb reference point, this function must receive answers and push notifications from the DAS. These are translated into the appropriate signaling protocol, which may require message (un)marshaling, and are then sent back to the corresponding UE through the appropriate endpoint. The routing of messages to endpoints requires some kind of mapping between request, endpoints and responses. This mapping must be supported by some kind of ID provided by the signaling protocol acting as breadcrumb. The management of this mapping may require SFE processes to be stateful, in which case the routing needs also to be based on a sticky session mechanism, as described above.

5.3.2 Distributed Application Server (DAS)

This function must receive the signaling requests generated by the UE (i.e. the external application), and translated by the DSFE. It must provide the appropriate semantics for them and must generate the corresponding answers and notifications. The specific logic to be executed for each request is defined by the application. At runtime, this distributed function is materialized as N monolithic AS instances.

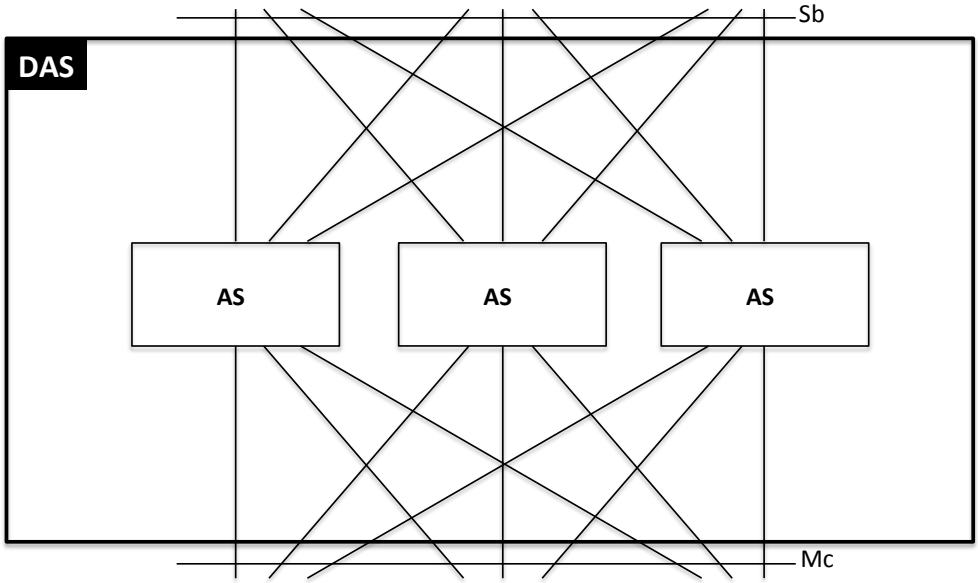


Figure 5. Example of materialization of the Distributed Application Server (DAS) function, which comprises a number of Application Server (AS) instances. At the northbound, AS instances interact with the DSFE through Sb interfaces. At the southbound, AS instances interact with the DMS function through Mc interfaces.

DAS functions are

- At Sb reference point, it receives messages from the DSFE. Those messages are routed to a specific AS instance holding the appropriate application logic suitable for processing them. We call Platform Application (PA) that specific logic. As specified above, this routing may require a sticky session mechanism. Then, the AS process provides semantics to request messages executing the appropriate logic (i.e. the PA logic). As result of this, answers shall be generated and sent back to the SDFE through the Sb reference point, following the mechanism specified above.
- As part of the AS, specific external services (e.g. databases, ESBs, authentication services, etc.) may be provided for the PA. We consider those

services to be application specific so that they do not constraint or influence NUBOMEDIA architecture.

- AS instances host PAs, and PAs must provide the appropriate Platform Application Logic (PAL). The PAL depends on the objectives of the application and provides its logic. PAs must also provide the appropriate Platform Application Media Control (PAMC). PAMC logic must use the Mc reference point for creating, managing and using media capabilities. PAL and PAMC roles are introduced in the section below.
- As part of the PAMC logic, at Mc reference point, messages for creating and managing the media capabilities required by the specific application will be sent to the DMS. After processing these messages, the DMS will generate be appropriate answers and push notifications, which are sent back to the DAS.
- At the Mc reference point, messages from the DAS to the DMS (i.e. message issued by the PAMC logic) must be routed and load-balanced appropriately. This must be implemented with the assistance of the CFM. Hence, every new media capability that need to be instantiated by the DAS will be placed by querying the appropriate Ci interface. In addition, given that MS instances are stateful, Mc protocols must provide some kind of media object identification (i.e. mediaObjectId) uniquely identifying each media capability. This mediaObjectId guarantees that a media control request issued to the Mc reference endpoint is routed to the appropriate MS instance.
- At the Mc reference point, answers and events pushed from DMS to the DAS (i.e. messages received by the PAMC) shall be routed to the appropriate AS monolithic processes (i.e. the specific application instances). If AS instance are stateful, this routing must require some kind of mapping between AS instances, Mc request, Mc responses and notifications. This mapping needs to be supported by some kind of ID provided by the Mc protocol acting as breadcrumb.

5.3.2.1 Application Server (AS) instances and the Platform Application (PA)

AS instances act as application holders. We name Platform Application (PA) each of these applications. PAs are created by platform developers for exposing PaaS APIs and SaaS services to the external world. In other words, a NUBOMEDIA instance holds different PAs and each PA is responsible of a specific capability exposed by that instance. For example, there may be a PA for exposing a SIP B2B videoconference service, a PA for exposing a JSON-based room videoconference API, a PA for exposing a video recording API, etc. Hence, PAs are the part of applications (understood as the final application consumed by the end-user) that needs to reside inside NUBOMEDIA. Developers will consume PAs exposed services and APIs applications to create such applications, which shall require additional logic external to NUBOMEDIA (directly at the UA or in external application servers).

Remark from the above that a PA is not a final application per-se. However, from the perspective of the NUBOMEDIA architecture, it plays the role of an application because it contains the logic (i.e. semantics) to be given to signaling messages. The fact that this logic may be designed to expose capabilities useful for external systems (i.e. UA or external applications to NUBOMEDIA) does not avoid us to call them Platform Applications inside this document.

AS instances may host one or many PAs and a given PA may be hosted at several AS instances at the same time.

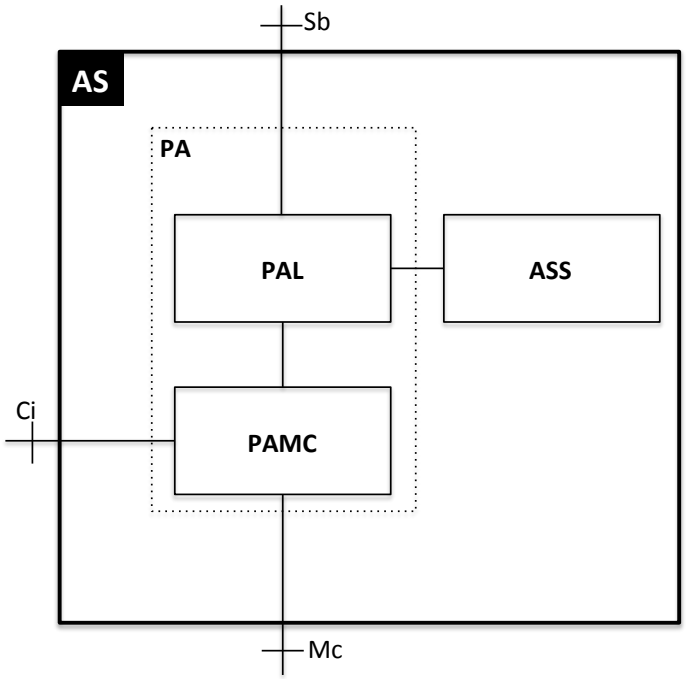


Figure 6. Each Application Server (AS) instance holds a Platform Application (PA). PAs always contains two types of logic: the Platform Application Logic (PAL), which is the specific logic of the platform application, and the Platform Application Media Control (PAMC) logic, which is the logic managing the media resources consumed by the application. In addition, AS instances by expose Application Server Services (ASS), which are external to NUBOMEDIA (e.g. databases, connectivity, etc.)

As shown in Figure 6, PAs must implement two types of logic: PAL and PAMC. Platform Application Logic (PAL) is the specific application logic, which implements the appropriate semantics associated to signaling messages. The PAL is in charge of security aspects (i.e. AAA) and of the management of the different application capabilities. The PAL may consume external Application Server Services (ASS) such as databases, corporate directories, web services, etc. The communication between the PAL and the ASS is out of the scope of NUBOMEDIA therefore requirements about the involved interfaces are not specified in this document.

The Platform Application Media Control (PAMC) provides the logic for controlling the low level media capabilities of NUBOMEDIA. This logic is responsible of transforming MS instances, which are monolithic, into a DMS function, which is elastic and distributed. This means that the PAMC is in charge of orchestrating, instantiating, releasing and managing the lower level media functions provided by the DMS. In other words, the DMS only holds monolithic media capabilities but does not have any intelligence.

Hence, the PAL manage high-level elastic cloud abstractions while the PAMC manages low level media capabilities located at specific MS instances. Following this, the PAMC receives control commands from the PAL and transform them into lower level control messages sent through the Mc reference point. This transformation may be complex so that one high level PAL command may generate many different low level ones and vice versa.

This high-level abstractions managed by the PAL strongly relies on the specific requirements of the application. For example, the abstractions for managing room videoconferences are very different from abstractions of real-time broadcasting

sessions. For this reason, specific abstractions and specific PAMC logic needs to the created for every specific service of PaaS API to be exposed at a NUBOMEDIA instance. However, in general, we call Distributed Media Pipeline (DMP) this abstraction, meaning that a DMP comprises all the MS instances used in a media session managed at a specific PAMC instance.

Besides, for the exposition of the DMP abstraction, the PAMC logic must manage media capability placement. When a new media object has to be created, the PAMC needs to determine whether it shall be hosted into a pre-existing MS instance (in that case selecting the most appropriate one) or into a whole new MS instance not already known by the PAMC (i.e not already belonging to its managed DMP). For achieving this, the PAMC logic may consume the cloud media management interfaces exposed at the Ci reference point. These interfaces must provide lifecycle management capabilities for the low level media features. These capabilities must include assistance for placement and disposal of MS instances.

The interface between PAL and PAMC may take different shapes depending on the nature of the application. In general, the PAL shall invoke primitives onto the PACM requesting two types of primitives:

- Factory primitives for the creation or disposal of DMP resources.
- Control primitives for the management of DMP resources.

On the opposite direction, the PAMC shall provide answers to PAL request as well as asynchronous events related to the different execution conditions of the DMPs.

For achieving this, the PAMC logic needs to maintain the appropriate state information for being able to provide that intelligence. Hence, PAMC instances might be stateful.

5.3.3 Distributed Media Server (DMS)

The DMS holds NUBOMEDIA media functions, which include media sending and receiving, media transcoding, media processing, media mixing and many others. The DMS function receives media control messages from the DAS through the Mc reference point. The DMS function provides media transport services with UAs and external applications through the M reference point. For this reason, Mc interfaces can be seen as the “signaling” of the DMS, however, for avoiding confusion, we will reserve the term “signaling” for interfaces and protocols on S and Sb reference points, while calling “media control” the signaling on all DMS related reference endpoints. At runtime, this distributed function is materialized as N monolithic MS instances.

DMS functions are

- At Mc reference point, it receives media control messages from the DAS (specifically from the PAMC) for instantiating and managing media capabilities. As specified above, the Mc reference point provides routing and load balancing capabilities toward the stateful MS instances. Each MS hosts media transport and processing capabilities organized as chains of monolithic and atomic abstractions called Media Elements (MEs). There may be media elements for sending/receiving media streams through different protocols, for applying a specific computer vision algorithm onto a stream, for recording streams, for mixing streams, for blending streams, etc. Media elements should be pluggable components meaning that additional media elements might be added at any time without requiring any other function of the architecture other than the MS to be aware. We will call Media Pipeline (MP) a chain of media elements providing a

specific media processing to an application. Hence a MP is a directed graph of ME where the output media stream of source MEs are fed as input media streams for sink MEs.

- At the Mc reference point, MS instances shall publish answers to PAMC media control messages as well as asynchronous events related to the internal working of the media elements. These events may include semantic media information, media element lifecycle information, media transport information or any other information considered useful for application developers.
- At the M reference point, MS instances provide transport protocols toward UE or external applications. These transports must comply with any of the media protocols supported by the DMS function. The specific parameterization of media transport protocols are controlled by the PAMC through the appropriate Mc messages.
- At the Mi reference point, different MS instances may exchange media information (i.e. media bits) as well as media control information (i.e. events, formats, synchronization, etc.) As a result, the Mi reference point must enable the appropriate cooperation among MS instances for providing distributed media processing and management

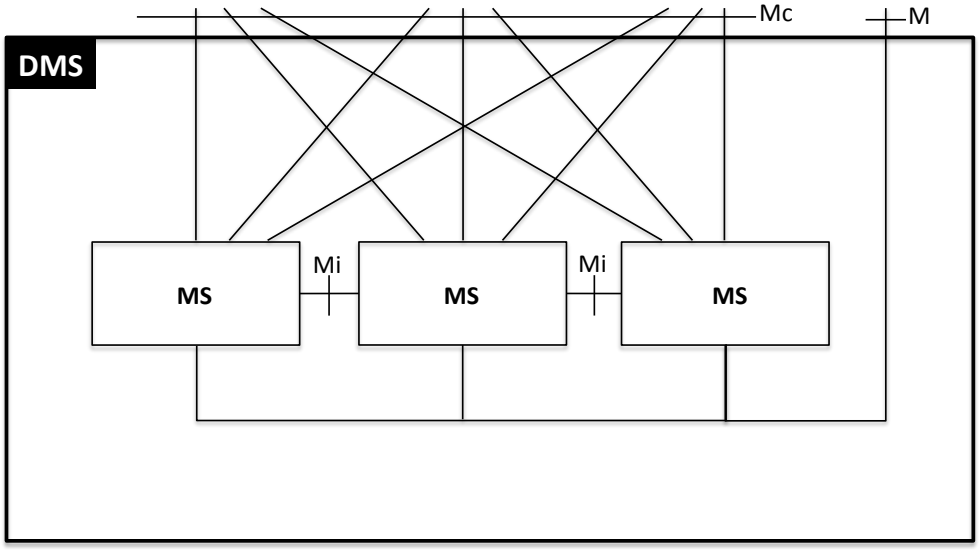


Figure 7. Example of materialization of the Distributed Media Server (DMS) function, which comprises a number of Media Server (MS) instances. At the northbound, MS instances interact with the DAS function through Mc interfaces. Also at the northbound, MS instances provide the media transport interfaces toward UE and external applications. MS instances may exchange control and media information through the Mi reference point.

5.3.3.1 Distributed Media Pipeline (DMP)

As introduced above, the DMP is not a function, but an abstraction created and exposed by the PAMC. However, understanding this abstraction is critical for having a meaningful comprehension of the NUBOMEDIA architecture as a whole. For this reason, we introduce a specific section devoted to digging into it.

The DMP abstraction creates the illusion of a "virtually infinite" (limited only by the underlying computing and networking resources) MS for the application, so that the PAL, and the PaaS APIs and SaaS services exposed by it, do not need to manage with the complexities of managing multiple MS instances. From this perspective, the PAMC acts as a DMP controller, managing the lifecycle of the DMP and controlling its internal MS instances and their relationship.

Hence, the DMP is materialized as N monolithic MS functions. Each MS hosts media transport and processing capabilities organized as chains MEs, as introduced above, creating MPs. So that the media control protocol exposed at Mc reference point makes possible to control MP behavior by creating and connecting media elements into intra MP media processing chains. In addition, the protocol also makes possible to create specific capabilities for creating inter MP media exchanges through the Mi reference point. This means that the DMP function comprises chains of network-interconnected and orchestrated MP executing in MS monolithic instances. In turn, each MP comprises chains of memory-interconnected media elements. As a result, the DMP function can be seen as distributed version of a MP, as illustrated in Figure 8.

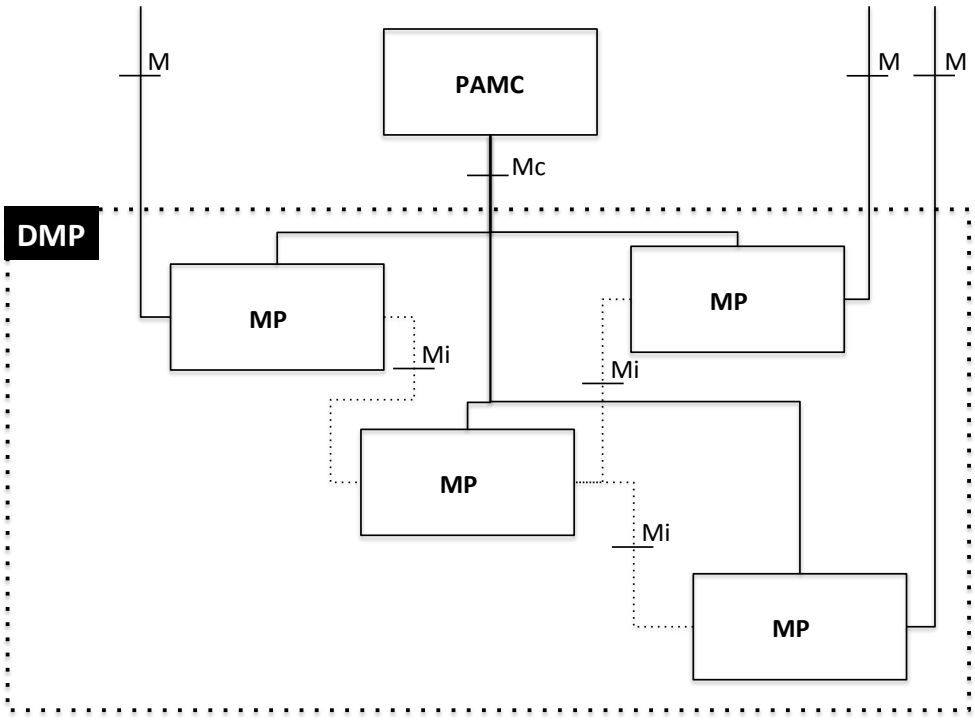


Figure 8. A Distributed Media Pipeline is an abstraction provided at the northbound interfaces of the Platform Application Media Control (PAMC) logic to platform developers and application developers. This abstraction can be seen as a generalization of the Media Pipeline (PM) concept. Hence, in the same way a PM is a chain of memory-interconnected Media Elements (ME) a DMP is a chain of network-interconnected MPs.

5.3.4 Cloud Functions Manager (CFM)

The CFM holds all NUBOMEDIA cloud capabilities, which include the low level virtual infrastructures (i.e. virtual computing, virtual networking, virtual storage, etc.) as well as a number of enablers controlling them. The CFM has the objective of isolating cloud and media application logic, so that advanced cloud features such as autoscaling can be provided by NUBOMEDIA with minimal interference among them.

The CFM internal structure is depicted in Figure 9. The main components of the CFM are the EMM, the CM and the Virtual Infrastructure itself.

5.3.4.1 Elastic Media Manager (EMM)

The EMM is the functional element managing the lifecycle of the NUBOMEDIA components. The EMM is responsible of the NUBOMEDIA austoscaling features. This means that the EMM must provide the appropriate logic for adapting the utilized

computing resources to the workload changes, so that, at each point in time the available resources match the current demand as closely as possible.

The EMM autoscaling mechanism provides two main operations, scaling in and scaling out, for horizontally scale a set of components. Scaling in means removing virtual machine instances when they are no longer required. Scaling out means adding additional machine instances when they are required.

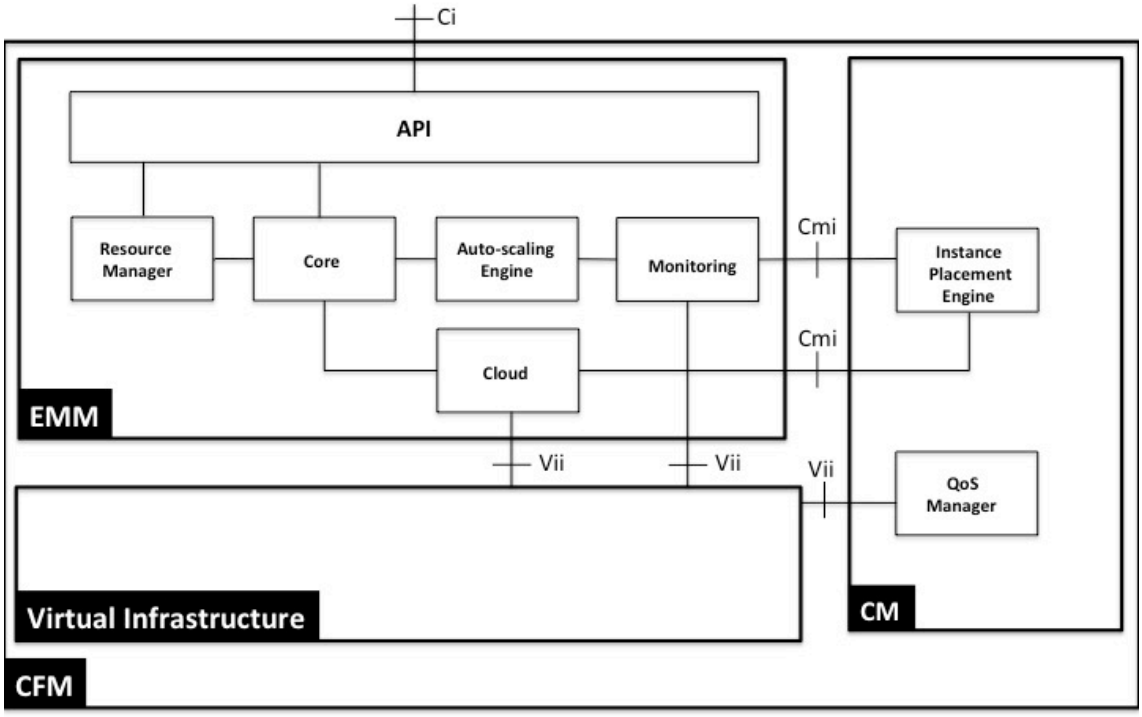


Figure 9. CFM architecture identifying its mains functions and reference points. Among the former (from left to right and from too to bottom): Elastic Media Manager (EMM), Virtual Infrastructure, Connectivity Manager (CM). Among the later: Cloud interface (Ci), Virtual infrastructure interface (Vii), Connectivity manager interface (Cmi)

The EMM autoscaling mechanism shall be based on scalability groups, so that the different NUBOMEDIA functions described above (i.e. DSFE, DAS and DMS) are managed by the EMM separately, allowing scalability based on different policies.

The EMM autoscaling procedures shall be based on alarms and scaling policies. Autoscaling alarms consists of rules based on monitoring information that are fired when specific monitoring conditions occurs. These alarms might be based on simple monitoring rules (e.g. average CPU utilization is over 60%) or on complex metrics involving advanced prediction algorithms combining different interrelated metrics (e.g. Kalman filters, etc.). These algorithms might contain QoS metrics provided by the DMS functions (e.g. media jitter, packet loss, etc.). When alarms are fired, one or several autoscaling policies should be executed.

Autoscaling policies are sets of instructions describing how the scaling action takes place. Autoscaling policies should define a type of scaling action (i.e. adding instances for scaling out, or removing instances for scaling in) as well as the intensity of the scaling action (i.e. the number of instances to add or remove).

A scale out operation shall never require any additional actions from any NUBOMEDIA function other than the EMM. However, a scale in operation might involve a two steps procedure requiring the intervention of the DAS function.

- At a first stage, the EMM selects a given instance for removal and sends a signal marking the instance as no longer available for receiving new sessions. The DAS function might use that signal for moving current users sessions on that instance to other instances, defined also as user mobility. Independently on that, the EMM must wait for a final signal before completely removing the instance. This signal might be provided by the DAS or by the DMS indicating that the involved instance does not contain user sessions any longer.
- At a second stage, the EMM receives that signal back and really terminates the involved instance.

For enabling autoscaling, the EMM requires the ability to start up and stop computing resources. This ability is based on two different capabilities:

- Deployment: it is the process of starting up (creating) a new virtual computing instance. The EMM consumes at it southbound a Virtual Infrastructure API (through the Vii reference point) for performing the deployment. Deployment requires the Vii to enable virtual image manipulation (i.e. storage and recovery) and virtual machine instance manipulation (i.e. start, stop, etc.)
- Provisioning: is the process of providing all the required configurations and commands to the computing resources created at the deployment phase. Provisioning is in charge of guaranteeing the appropriate service topology for the different function instances deployed in NUBOMEDIA.

The EMM also provides media session placement capabilities. At it northbound, through the Cm reference point, the EMM exposes an interface for runtime session placement on the media servers available inside the NUBOMEDIA infrastructure. The DAS function shall consume this interface as part of the PAMC logic. Thanks to this, the PAMC does not need to held any kind of placement logic and, hence, it is agnostic to the underlying Virtual Infrastructures and its associated monitoring mechanisms and APIs.

The EMM has an internal structure which is also depicted in Figure 9. Its main components are the following

- API: Abstracts all EMM functionalities and exposes them through a set of coherent networked APIs through at the Cm reference point.
- Resource Manager: It manages the resources which are available in NUBOMEDIA. Those resources are stored in a database, and provided to the users in a form of a catalogue via the APIs.
- Core: Holds the logic for deploying and provisioning NUBOMEDIA components. In particular, it provides a state machine for managing their lifecycle, and it interoperates with the other functions for realizing the instantiation of NUBOMEDIA.
- Autoscaling Engine: Holds the autoscaling logic including the definition of the alarms and policies to be applied to different scalability groups.
- Monitoring: It provides an abstraction of the different monitoring systems available, in order to retrieve monitoring information from all levels.
- Cloud: In order to have interoperability among different clouds, the Cloud element offers an abstracted interface of the CRUD operations available for managing virtual compute, storage and network resources.

5.3.4.2 Connectivity Manager (CM)

As shown in Figure 9, other main component of the CFM is the CM function. The CM is the part of the NUBOMEDIA Architecture that realizes the control and management of the SDN topology provided by the virtual network infrastructure. Its main features are:

- Optimal Instance Placement: during the deployment of an NUBOMEDIA Infrastructure an algorithm decides where to place each NUBOMEDIA instance inside the cloud infrastructure. For example, when a new MS or AS instance is to be created, the CM shall determine the precise placement for it to be deployed.
- SLA enforcement: due to quality requirements arising by the media type or an explicit configuration the cloud infrastructure will be modified to fulfill this requirements

Internally, the CM has the following modules:

- Instance Placement Engine: It calculates the best position for a virtual instance inside the Virtual Infrastructure. This function decides where to place the specific element based on different indicators such as resource utilization of the compute nodes and available network capacity.
- QoS Manager: It provides functions to enforce e specific QoS class for a specific host. According to previous calculations, hosts can be equipped with guarantied and maximum bitrate.

5.3.4.3 Virtual Infrastructure

This function referrers to the underlying virtual resources (i.e. virtual machine instances and virtual networks) as well as the capabilities for managing them. This Virtual Infrastructure function shall be provided to NUBOMEDIA, which controls it through a set of APIs exposed at the Vii reference point. In general the Virtual Infrastructure shall take the form of an IaaS (Infrastructure as a Service) cloud on top of which NUBOMEDIA is deployed.

NUBOMEDIA architecture assumes that he Virtual Infrastructure shall provide capabilities such as the following:

- Management of virtual images
- Management of virtual computing resources
- Management of virtual networking resources
- Management of virtual storage resources
- Monitoring of virtual resources

5.4 Reference points

5.4.1 Reference point S

The S reference point (S for Signaling) is where the external application logic (i.e. external UE or AS) finds the media enabled applications through an application layer signaling protocol. The S reference point needs to provide an IP transport service among the external application logic and the DSFE.

At reference point S, the following interfaces are expected to be found:

- SIP interface provided by the DSFE so that the external application logic can reach applications through SIP.
- RESTful interface provided by the SDFE so that the external application logic can reach applications through simple HTTP requrests.
- JSON over WebSockets provided by the SDFE.

5.4.2 Reference point Sb

The Sb reference point (Sb for Signaling broker) is where the DSFE and DAS find each other. At the Sb reference point, a message broker capability must be provided with the ability of routing messages from SFE instances to the appropriate AS instances and answers in the opposite direction.

At reference point S, interfaces such as the following could be used:

- A message broker providing a message queuing service suitable for implementing the required routing.
- A publish subscribe broker providing publish subscribe capabilities suitable for implementing the required routing.

5.4.3 Reference point Mc

The Mc reference point (Mc for Media control) is where the DMS function exposes control capabilities. At Mc reference point, the following interfaces must be found:

- A JSON over WebSockets protocol exposed by the DMS providing direct access to controlling DMS capabilities.

Besides, at Mc reference point, additional interfaces could be exposed:

- An H.248 interface.
- An MSML interface.

5.4.4 Reference point Ci

The Ci reference point (Ci for Cloud interface) is where NUBOMEDIA functions access cloud capabilities. At the Ci reference point the following interfaces should exist:

- At the CFM, the Ci reference point must provide an interface to the DAS exposing session placement capabilities on MS instances.
- At the CFM, the Ci reference point might provide an interface to the DSFE exposing session placement capabilities on AS instances.
- At the CFM, the Ci reference point must provide an interface to the DMS for the registration and failure detection of MS instances.

All these interfaces shall be based on a RESTful model with JSON data marshaling.

5.4.5 Reference point Ca

The Ca reference point (Ca for cloud administration) exposes capabilities for monitoring and managing autoscaling in NUBOMEDIA instances. Interfaces at this endpoint are to be defined in later releases of NUBOMEDIA.

5.4.6 Reference point M

The M reference point (M for Media) is where the DMS and the external application logic find each other for the exchange of media. The M reference point needs to provide

an IP transport capability suitable for UE and MP/MR instances to find each other (mechanisms for this are protocol dependent). The M reference point may expose additional services for improving media QoS (e.g. SDN based routing policies, etc.)

At M reference point, the following interfaces must be supported:

- H.26x/AMR over RTP
- WebRTC: DTLS/ICE/VP8+Opus over SRTP.

Besides, at M reference point, the following interfaces might be exposed

- MPEG-DASH
- Flash: H.264/RTMP
- HTTP Pseudostreaming for H.264 (MP4) and VP8 (WebM) based contents.

5.4.7 Reference point Mi

The Mi reference point (Mi for Media internal) is where the different DMS exchange media information for providing the DPM abstraction. The Mi reference point needs to provide an IP transport capability suitable the exchange of media with very low latency, very high throughput and efficient bandwidth utilization. The Mi reference point must also provide meta information exchange among the different MS instances guaranteeing that the internal media procedures (i.e. media negotiation) are possible.

Interfaces at the Mi reference point are still under conception.

References

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels. https://www.ietf.org/rfc/rfc2119.txt

[RFC2616] Hypertext Transfer Protocol -- HTTP/1.1. http://www.w3.org/Protocols/rfc2616/rfc2616.html