Grant Agreement No.: 731677
Call: H2020-ICT-2016-2017
Topic: ICT-13-2016
Type of action: RIA

D3.5: FLAME Technology Roadmap V1

Michael Boniface (IT Innovation Centre) | 20 November 2017

This report is the first technology roadmap for a ground-breaking media service delivery platform being developed by the FLAME project. The report describes the software products to be delivered at infrastructure, platform and media service layers and how combinations of products are used to exploit the benefits of highly distributed software-defined infrastructures. Each product is described in terms of features, baseline implementation technologies and release schedule. At the core of the roadmap is the FLAME platform that brings together components for orchestration, Service Function Routing, Service Function endpoint management and cross-layer management and control. A systems integration and testing plan describes the DevOps environment including multi-project structure, development workflows and continuous integration processes supported by build, provisioning, configuration and automated testing tools. A software integration infrastructure is designed that replicates a part of the production infrastructures in ways that allows flexible configuration of different cross-component test scenarios. Finally, the downstream staging and production infrastructures are summarised completing the end-to-end DevOps pipeline for efficient and high-quality delivery.

Work package	WP 3
Task	Task 3.4
Due date	30/09/2017
Submission date	20/11/2017
Deliverable lead	IT Innovation
Version	1.0
Authors	Michael Boniface (IT Innovation), Simon Crowle (IT Innovation), Dirk Trossen (InterDigital), Istvan Lajtos (BRISTOLOPEN) , Carlos Alberto Martin Edo (Atos), Pouria Sayyad Khodashenas (i2CAT), Marisa Catalan Cid (i2CAT), David Jones (BRISTOLOPEN)
Reviewers	Gino Carrozzo (NXT), Julia Chatain (ETH)
Keywords	Media Services, Software-defined infrastructures, technical roadmap, systems integration, DevOps

Disclaimer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731677.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		**R**
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to FLAME project and Commission Services	

Co-funded by the Horizon 2020 Framework Programme of the European Union

EXECUTIVE SUMMARY

This report is deliverable D3.5 FLAME Technology Roadmap V1 of the FLAME project. The report describes the roadmap for development, integration and production deployment of a ground-breaking media service delivery platform. The roadmap aims to deliver software products deployed as operational services on real-life software-defined infrastructures for trials and experimentation. The primary purpose of the trials is to validate the FLAME offering by delivering performance and cost benefits to media service providers and enhanced quality of experience to end users.

The roadmap considers activities across all layers of the stack covering infrastructure, platform and media services. Each layer has one or more software products that is deployed to offer services at infrastructure, platform or media service layers with each software product having its development roadmap and schedule. At the infrastructure layer, different solutions are considered depending on how elements such as routers, switches and compute are realised through software or hardware implementations. The infrastructure is based on common technologies for the management of virtualized infrastructures (i.e. OpenStack) and agreed specifications for the SDN Fabric (i.e. Open Flow). On top of the virtualized infrastructures the FLAME Platform product is deployed. The Platform product is the core of the project and brings together advanced components offering media service orchestration, service function endpoint management, service function routing and cross-layer management and control. Finally, a set of Media Service products are deployed within the Platform to offer a variety of media service capabilities enhanced through platform features. This report describes the high-level features of each product and the release schedule. For each Platform product more detail is provided including a feature analysis that maps features to requirements, component interfaces, and an overall critical path analysis for implementation. The high-level release schedule for FLAME software products and FLAME services is shown in Figure 1.

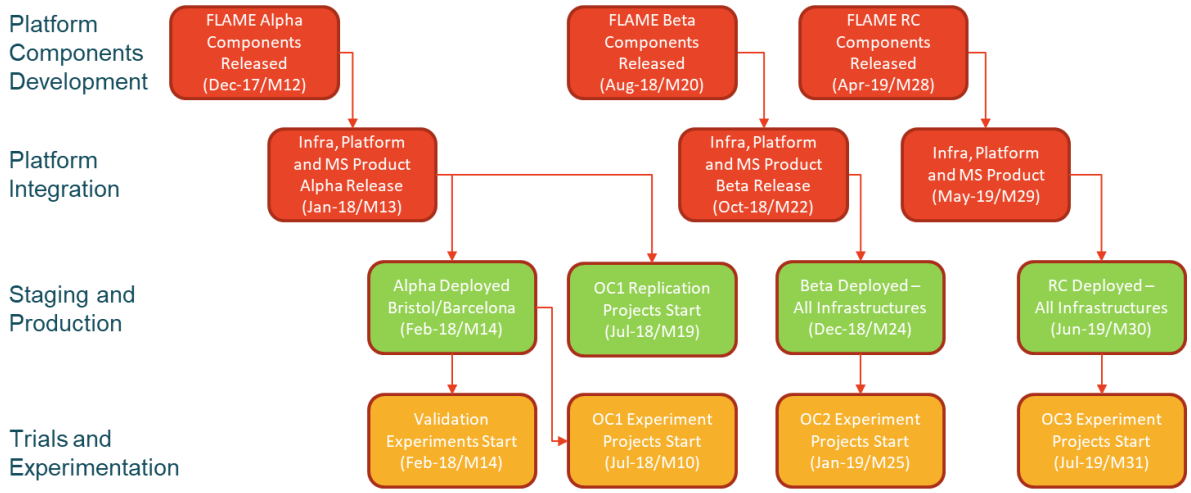

Figure 1: High-Level roadmap for FLAME product and service releases

A systems integration and testing plan describes the DevOps environment including multi-project structure, development workflows and continuous integration processes supported by build, provisioning, configuration and automated testing tools. A software integration infrastructure is designed that replicates part of the production infrastructures in ways that allow flexible configuration of different cross-component test scenarios. Finally, the downstream staging and production infrastructures are summarised completing the end-to-end DevOps pipeline for efficient and high-quality delivery. The overall roadmap is designed to ensure alignment of activities across all work

Co-funded by the Horizon 2020 Framework Programme of the European Union

packages in the project from component development, integration through to trials and experimentation.

Co-funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF FIGURES

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF TABLES

Co-funded by the Horizon 2020 Framework Programme of the European Union

ABBREVIATIONS

AR	Augmented Reality
AVC	Advanced Video Coding
CDN	Content Delivery Network
CI	Continuous Integration
CLMC	Cross-Layer Management and Control
CMS	Content Management System
DC	Data Centre
EaaS	Experimentation-as-a-Service
ETSI	European Telecommunications Standards Institute
FPGAs	Field Programmable Gate Arrays
FQDN	Fully Qualified Domain Name
FMI	Future Media Internet
GPUs	Graphical Processing Units
HTTP	Hyper Text Transfer Protocol.
IP	Internet Protocol
KPI	Key Performance Indicator
NAP	Network Access Point
NFV	Network Function Virtualisation
PCE	Path Computational Element
PIML	Personalisation, Interactivity, Mobility and Localisation
QoE	Quality of Experience
QoS	Quality of Service
SDN	Software Defined Network
SF	Service Function
SFC	Service Function Chain
SFEMC	SF Endpoint Management and Control

Co-funded by the Horizon 2020
Framework Programme of the European Union

SFR	Service Function Routing
SME	Small and Medium Enterprise
TLS	Transport Layer Security
TOSCA	Topology and Orchestration Specification for Cloud Applications
TRL	Technology Readiness Level
UE	User Equipment
UX	User Experience
VM	Virtual Machine
VIM	Virtual Infrastructure Manager

Co-funded by the Horizon 2020
Framework Programme of the European Union

1 INTRODUCTION

1.1 PURPOSE

This document aims to describe the technical roadmap for implementation, integration, testing and deployment of FLAME technologies supporting trials and experiments of media services on highly-distributed software-defined infrastructures. The goal is to provide software development teams responsible for FLAME software products and operations teams responsible for service deployment with the feature release schedules and DevOps processes that ensure timely delivery and results to an acceptable level of quality.

1.2 SCOPE

The project is structured into three iterative development phases aligned with the strategic activities of the work:

- *Jan-17 to Feb-18: Research and innovation foundations*: design, implement and deploy the Alpha release of the FLAME platform ready for trials in Bristol and Barcelona production infrastructures.
- *Mar-18 to Jun-19: Ecosystem building and disruptive experimentation:* operate trials and experiments to validate the platform, working on feature enhancements towards the Beta release.
- *Jul-19 to Dec-19: Sustainability*: transition towards exploitation and sustainability, hardening the platform for RC release and working closely with technology adoption partners

The high-level platform engineering cycle follows these project phases. The project is currently in the "research and innovation foundations" phase. D3.5 FLAME Technology Roadmap forming part of a series of public reports delivered in each phase (see Figure 2).

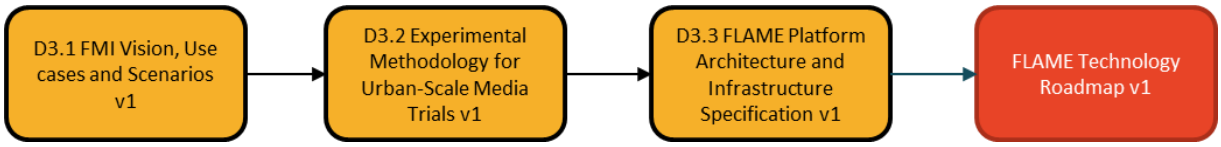

Figure 2: FLAME platform engineering reports

This report is the first version of the roadmap with further updates planned to be delivered at Jun-18, Dec-18 and Dec-29. Related reports include:

- D3.1 [FLAME-D3.1] describes a series of early scenarios and use cases for interactive media using the platform
- D3.2 [FLAME-D3.2] describing a methodology for conducting urban scale trials that explore the cross-layer and multi-stakeholder interactions within the systems-under-test.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- D3.3 [FLAME-D3.3] describes the architecture and infrastructure specifications for the FLAME platform, elaborating the use cases from D3.1, refining system requirements and identifying platform components and interfaces.

D3.3 is the primary reference point for the technical roadmap because it provides the overall structure of the platform and allows features and development tasks to be decomposed into areas of work.

The target audience for this deliverable are developers working on FLAME software products, infrastructure owners responsible for production deployment and wider stakeholders interested in the FLAME offering, features and expected release schedules.

1.3 DELIVERY PARTNERS

The project is delivered by members of the FLAME consortium who have specific responsibilities for implementation, operations, engagement and marketing of FLAME. The partners are referred to by acronyms throughout this report as shown in Table 1.

Participant organisation name	Short Name	Country	Roadmap Leadership Roles
IT Innovation Centre	ITINNOV	UK	Platform, CLMC
Atos Spain SA	ATOS	Spain	Media Services
InterDigital Europe Ltd	IDE	UK	SFR, SFEMC
Fundacio Privada i2CAT, Internet I Innovacio Digital a Catalunya	i2CAT	Spain	Barcelona Infrastructure Operator
University of Bristol	UNIVBRIS	UK	Bristol Infrastructure Strategy
Nextworks	NXW	Italy	Validation Experiment
Martel GmbH	Martel	Switzerland	Media Services
De Vlaamse Radio En Televisieomroeporganisatie NV	VRT	Belgium	Validation Experiment
The Walt Disney Company (Switzerland) GmBH	DRZ	Switzerland	Validation Experiment
Eidgenoessische Technische Hockschule Zuerich	ETH	Switzerland	Validation Experiment
Institut Municipal d'Informàtica de Barcelona	IMI	Spain	Barcelona Infrastructure Strategy
Bristol is Open Limited	BRISTOLOPEN	UK	Bristol Infrastructure Operator

Table 1: FLAME consortium partners

Co-funded by the Horizon 2020 Framework Programme of the European Union

2 TECHNOLOGY ROADMAP

2.1 PROJECT MILESTONES

FLAME plans three major releases of the FLAME offering within the lifetime of the project. The timing of major releases is aligned with the timescales of trials. Each major release will include significant feature enhancements within the overall offering across infrastructure, platform and media services. A release at the project level indicates the launch of a "FLAME Service" for trials in contrast to the release of specific software products that the FLAME Service depends on.

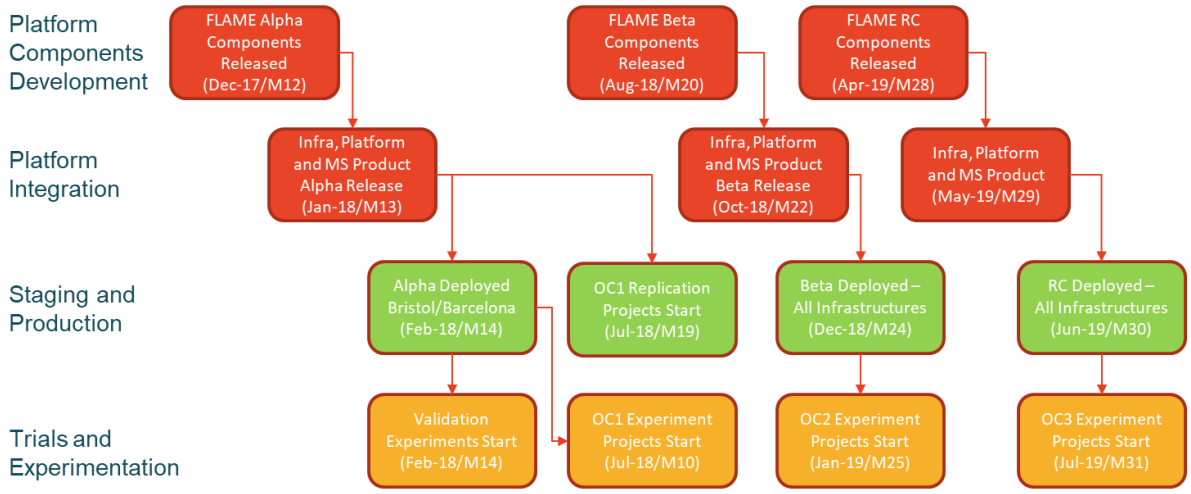

Figure 3: Platform releases in relation to project milestones

Major releases are planned for Feb-18, Dec-18 and Jul-19 with the working names of Alpha, Beta and Release Candidate (see Figure 3). The major releases correspond to milestones for FLAME feature implementation. The project expects to implement DevOps processes that offer greater agility in the implementation of release of features. As such Minor releases will be delivered in between major milestones to incorporate new features when they are available and hot bug fixes when they are critical to service operations.

2.2 OVERVIEW OF SOFTWARE PRODUCTS

FLAME will deliver three types of software products that reflect the layering in the architecture, as shown in Figure 4 and described in Table 2.

At the lowest level are infrastructures products used by infrastructure providers to offer compute, storage and networking resources to the FLAME platform. The infrastructure resources are typically a virtual slice of a larger physical infrastructure and the acquisition of such resources is typically wholesale on a long term basis between an infrastructure and platform operator. The FLAME platform product is the core of the project and consists of four main components as summarised below:

- Orchestration: provides infrastructure resources to media services by defining surrogate policies and key performance indicators for shorter term control.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Service Function Endpoint Management and Control: implements surrogate management policies as well as to set suitable shorter-term control policies for service function endpoints
- Service Function Routing: configures the switching fabric of the underlying infrastructure using OpenFlow
- Cross-Layer Management and Control: provides a rich pool of data as the basis for deriving insights into infrastructure, platform and media service performance.

The software product types have dependencies as shown in Figure 5. Each product will be delivered through a dedicated development and continuous integration pipeline as described in Section 4. The products have been selected to ensure loose coupling and reuse of products in accordance with the architectural decisions. The overall integration process for the software products required to deliver the major releases of the FLAME service are described in Section 4.

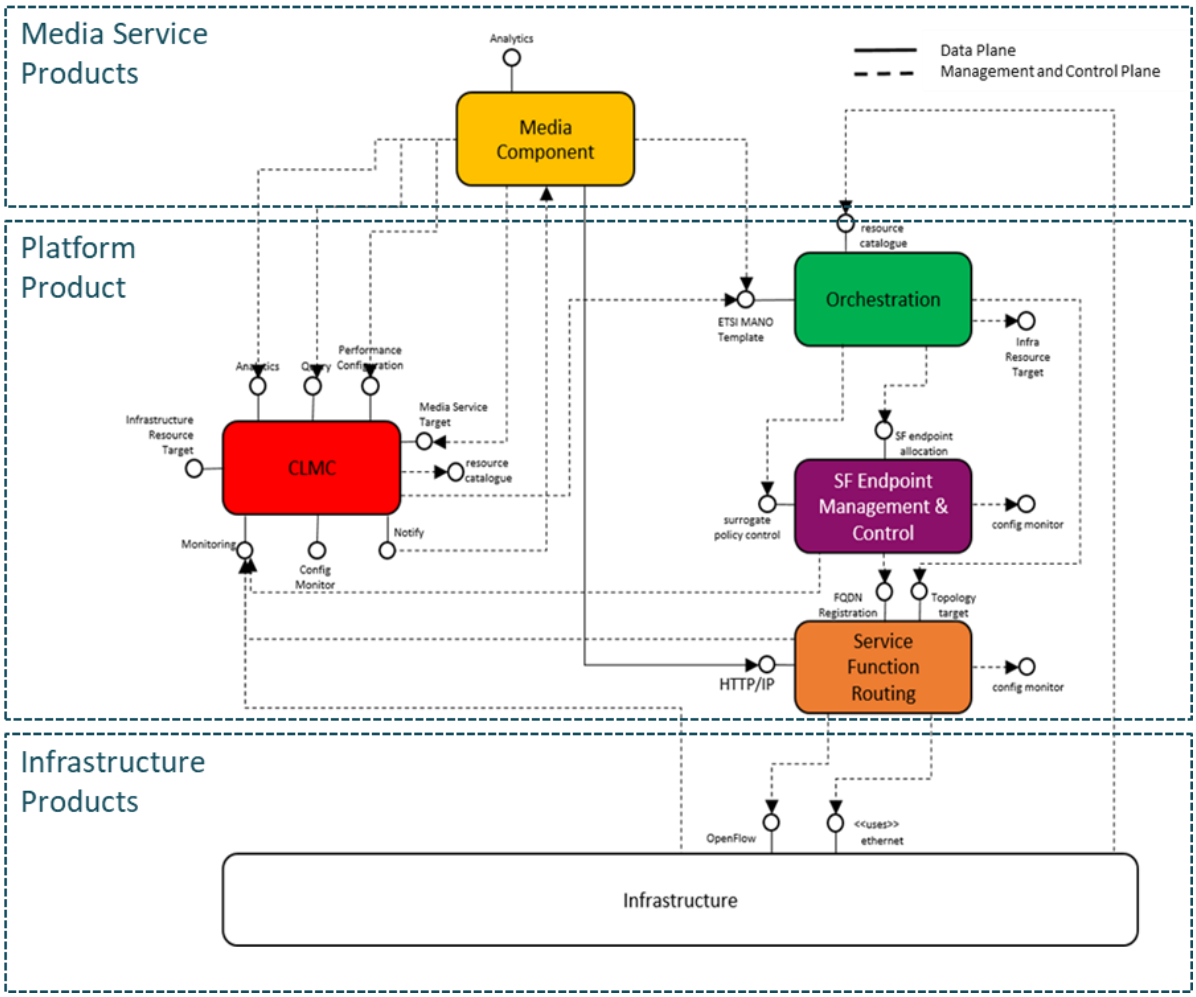

Figure 4: FLAME software products in relation to architecture

Software Product Type	Description
Infrastructure Product	A product offering access to and management of infrastructure resources based on specific hardware configurations. The infrastructure abstraction offered must be common across all Infrastructure Products although it is expected there will be some variation in function and performance for different Infrastructure Products. Multiple

Co-funded by the Horizon 2020 Framework Programme of the European Union

Software Product Type	Description
	Infrastructure Products are expected covering different integration, staging and production environments. It is expected that infrastructure products will build on and adapt widely used open source available solutions, e.g. OpenStack[1], OpenDaylight[2] and Floodlight[3] and where necessary contributions will be made to open source extensions of existing infrastructure products
Platform Product	A product offering flexible management and delivery of media services deployed on Infrastructure Products. One platform product is expected to be delivered. This product will be able to be configured for different Infrastructure Products. The platform product is the primary outcome of the FLAME project.
Media Service Product	A product offering content production, management and/or distribution features that directly benefit from the features of the Platform Product. Many Media Service Products are expected to be offered. The selection is based on the Media Service products that benefit most from Platform Product features and are in demand for delivery of new forms of user experience and social interaction.

Table 2: FLAME software products

Figure 5 and Figure 6 provide an overall summary of the approach. Software products are developed within a dedicated continuous integration pipeline. In this example, pipelines for Infrastructure Product A, Platform Product and Media Service Product are shown. Firstly, Infrastructure Product A is built and tested on Infrastructure A, which for functional integration will be commodity hardware with software-based switching. If Infrastructure Product A passes integration tests it is made available for Platform Product integration testing on Infrastructure A. If the Platform Product passes integration the product is made available for Media Service Product X integration testing on Infrastructure A. If all integration tests pass then the Platform Product and the Media Service Product are distributed to the staging infrastructure for acceptance testing and finally deployed on the production infrastructure for real-life trials.

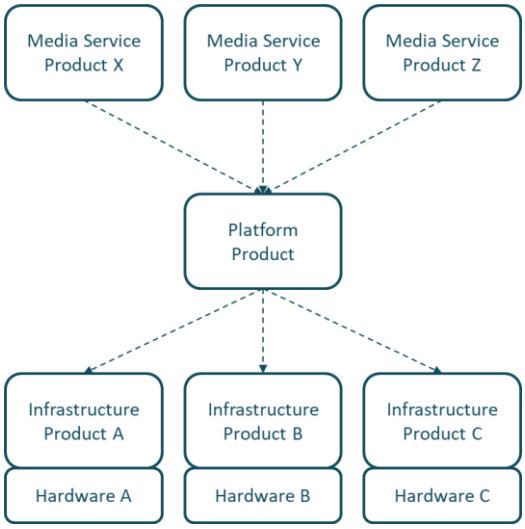

Figure 5: High level product dependencies

[1] https://www.openstack.org/

[2] https://www.opendaylight.org/

[3] http://www.projectfloodlight.org/floodlight/

Co-funded by the Horizon 2020 Framework Programme of the European Union

Any changes to software products in the pipeline may trigger continuous integration tests for products downstream in the pipeline that depend on the product. The level of automation in continuous integration process across products and the scheduling of integration tests at different phases in the pipelines depends on the level of human control desired and the cost of integration testing itself. The final stage of deployment on the production infrastructure can also form part of continuous deployment processes, however, this depends on the policy of infrastructure operators.

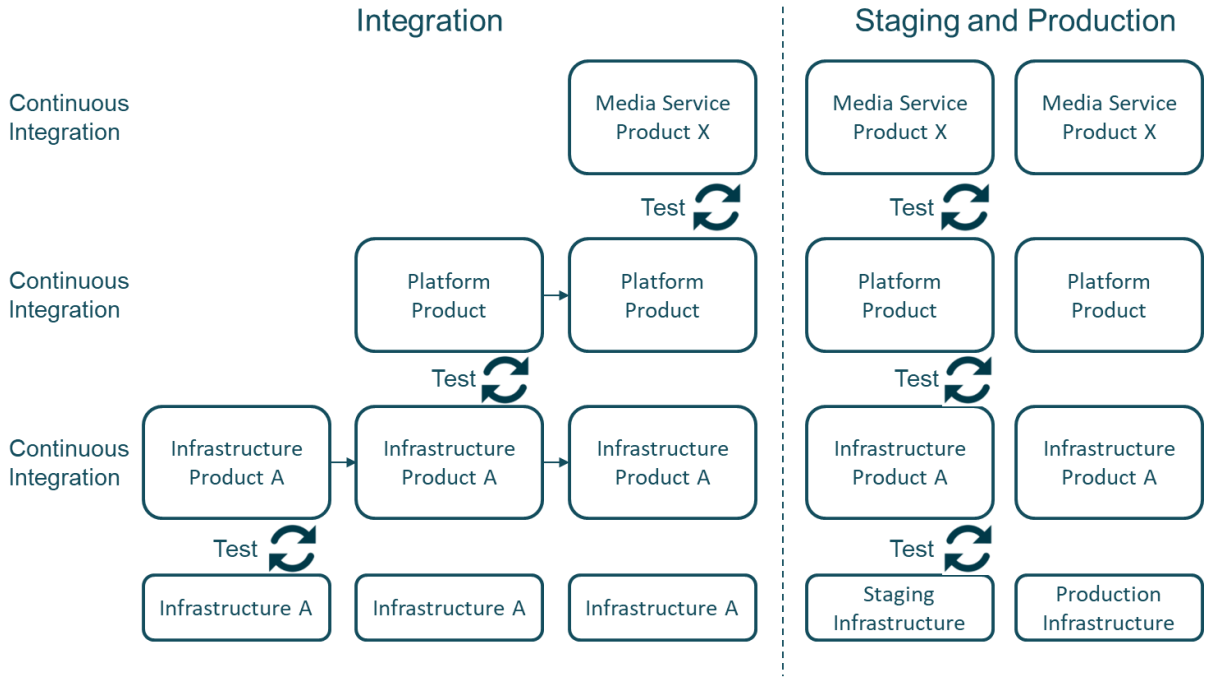

Figure 6: Overview of software product integration and release

Software Product	Project Owner	Responsibility
Infrastructure	BRISTOLOPEN	• Integration Infrastructure (ITINNOV) • Bristol infrastructure (BRISTOLOPEN) • Barcelona infrastructure (i2CAT)

Platform	ITINNOV	• Orchestration (IDE)
		  ○ TOSCA++ specification language (Atos)
		  ○ FLAME orchestrator (IDE)
		  ○ Platform orchestrator (UNIBRIS)
		  ○ Media service orchestrator (Martel)
		• SF endpoint management and control (IDE)
		• SF routing (IDE)
		• Cross layer management and control (ITINNOV)

Media Service	Atos	• Media service selection, adaptation and packaging (Atos) • Media service packaging (Martel) • Media service monitoring (Atos)

Table 3: Partner responsibilities across product implementation, integration and deployment activities

FLAME products are implemented, integrated and tested through contributions from multiple organisations. Table 3 shows the distribution of responsibilities for technical partners contributing to the implementation of the Platform product and Media Service products. Each component has an owner responsible for delivery of the components to integration based on contributions from other organisations.

Co-funded by the Horizon 2020 Framework Programme of the European Union

2.3 INFRASTRUCTURE PRODUCT ROADMAP

2.3.1 Infrastructures

FLAME will deliver a set of Infrastructure Products offering access to and management of infrastructure resources based on specific software and hardware configurations. Infrastructures are expected to support integration testing and production deployment at FLAME replication sites. Integration infrastructures must support the software product integration and testing pipeline to ensure software is acceptable for deployment in production environments. Production infrastructures must offer FLAME services for real-life trials. Infrastructures elements can be considered as software or hardware (see Table 2) although the actual distribution of infrastructure elements realised in each depends entirely on the operator's configuration.

Infrastructure Flavour	Description	Technologies
Software-based	An infrastructure that is entirely based on commodity hardware and where compute, storage and networking function is delivered through virtualisation. Software-based infrastructures offer high flexibility and lower costs but performance is degraded.	OpenStack, OpenDaylight, Open vSwitch
Hardware-based	An infrastructure that includes dedicated hardware elements such as networking (e.g. switches), general processing (e.g. Common servers), and hardware accelerators (e.g. GPUs) and bespoke processing (e.g. FPGAs). Hardware-based infrastructures offer higher performance but with increased costs and less flexibility. At each experimentation site, depending on the resource availability all or some of the hardware resources will be offered.	Cloud IT resources, edge server, network switches, radio access elements

Table 4: Infrastructure flavours

Today, there is no uniform definition of a production infrastructure in FLAME, although common specifications have been established. The FLAME replication process aims to provide guidelines for infrastructure capability, capacity and management processes to ensure a minimum level of uniformity between infrastructures. However, the implementation of such infrastructures is expected to be different even if they are offered using a common infrastructure abstraction.

The consequence is that infrastructures need to be provided that support the full pipeline from integration through to staging and production. These infrastructures need to be intelligently designed to consider appropriate function and scale in relation to production, as due to cost limitations, it is not possible to entirely replicate hardware from production environments at all stages. The expected infrastructures are described in Table 5.

Infrastructure	Description	Operator
Functional Integration Testing	An infrastructure based on commodity hardware using software defined switches. Used for functional integration testing of software products.	ITINNOV
Hardware-Based Testing	An infrastructure that clones a production infrastructure that includes dedicated networking hardware.	IDE
Media Service Sandboxing	A tiny infrastructure based on commodity hardware using software defined switches. Used for media service providers to test APIs	Media Service Provider

Co-funded by the Horizon 2020 Framework Programme of the European Union

Infrastructure	Description	Operator
Bristol Staging	A small-scale clone of the Bristol infrastructure used to perform acceptance testing of Platform and Media Service Products on the Bristol Infrastructure Product	BRISTOLOPEN
Bristol Production	An urban scale production testbed deployed throughout the city of Bristol. Used to conduct real-life trials with citizens and businesses to determine the acceptance, viability and performance of the FLAME products in Bristol	BRISTOLOPEN
Barcelona Staging (Validation)	A small-scale clone of the Barcelona infrastructure used to perform acceptance testing of Platform and Media Service Products on the Barcelona Infrastructure Product	I2CAT
Barcelona Production	A production testbed deployed in the city of Barcelona. Used to conduct real-life trials with citizens and businesses to determine the acceptance, viability and performance of the FLAME products in Barcelona	I2CAT

Table 5: Integration and production infrastructures

Each infrastructure must be defined in terms of configuration and capacity, including expected capacity changes over the lifetime of the project. Understanding the relative capacities of each infrastructure ensures that integration testing is completed at a scale appropriate for the expected usage on production infrastructures.

2.3.2 Infrastructure Products

The relationship between infrastructures and Infrastructure Products is shown in Figure 7. The diagram shows how Infrastructure Products need to be established for the different hardware environments. The FLAME replication process will ensure that Infrastructure Products offer uniform capabilities to the Platform Product and where possible share a similar technology baseline, although the baseline needs to be adaptively configured according to the variation in hardware setups.

The distribution and reuse of Infrastructure Products depends on how integration and replication is implemented. The Infrastructure Product targeting commodity hardware will be distributed considering the need for media service providers to establish their own testing sandbox and the likelihood of establishing small scale infrastructures for evaluation and demonstrations at events. The reuse of Infrastructure Products between production sites is currently unknown, although the infrastructure at BRISTOLOPEN and i2CAT are sufficiently different that different Infrastructure Products are needed. It's unclear at this stage that the products will converge, or if replication will be achieved using a defined FLAME Infrastructure Product or further additional Infrastructure Products offered by replicators.

Regarding the specific infrastructure technologies, FLAME is built on a SDN-enabled networking fabric and implements a stateless switching solution[4] which requires the switches and controller(s) to be at least OpenFlow 1.3 compatible. As there is no capability verification alliance for OpenFlow (e.g. Wi-Fi alliance) and the OpenFlow 1.3 features being considered as "experimental", it is highly recommended to double check with the vendor of the fabric if the following two features are supported:

[4] Martin J. Reed, Mays F. Al-Naday, Nikolaos Thomos, Dirk Trossen, George Petropoulos and Spiros Spirou, "Stateless multicast switching in software defined networks", Online: https://arxiv.org/abs/1511.06069

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Switches support arbitrary bitmask matching via semantically overloaded IPv6 fields
- Controller supports handling (read and insert) of arbitrary bitmask matching rules

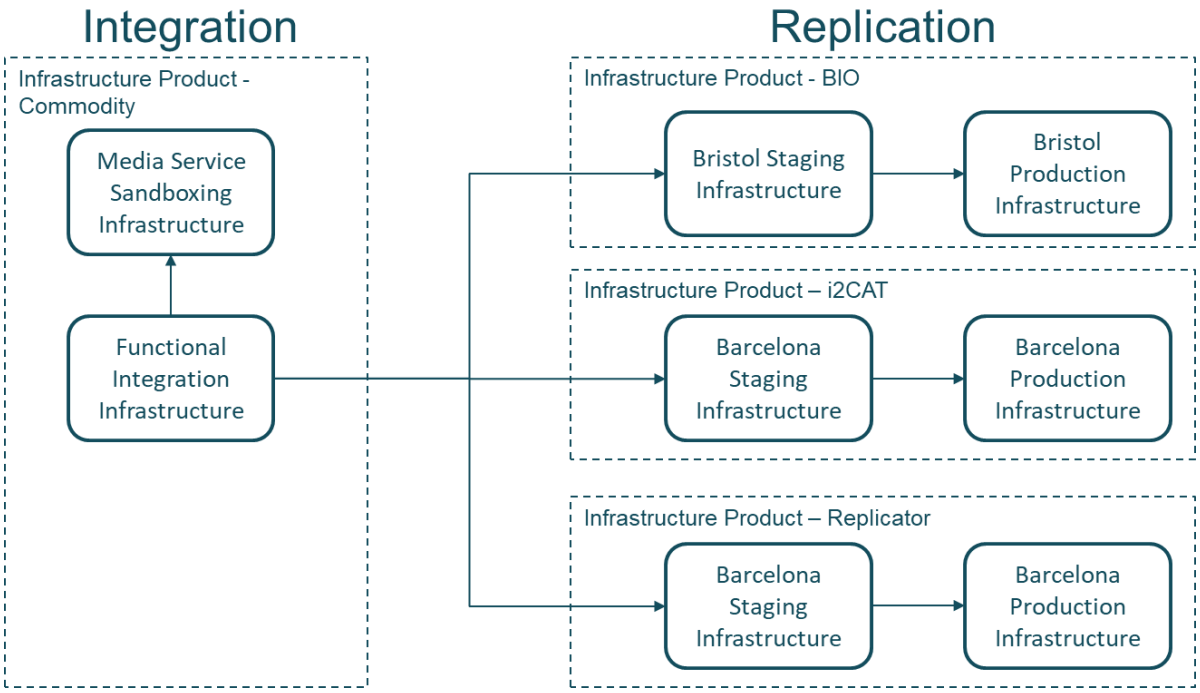

Figure 7: Relationship between infrastructure and Infrastructure Products

If the infrastructure is based on a software-based switching, it is recommended to use Open vSwitch which has not shown any compatibility issues. However, hardware switches implement the actual switch in their TCAM tables which have an OpenFlow compatible API and only one switch is known to support arbitrary bitmask matching, i.e. PICA8[5]. As mentioned before, the chosen SDN controller must

- Accept the rules communicated via the REST API; and
- Insert them into the switches.

The following controllers have been successfully tested: Floodlight and OpenDaylight. ONOS does not support arbitrary bitmasks yet.

For compute and storage resources OpenStack is adopted and will adapted to meet the needs of the platform. The adaption expects to focus on specific configurations of OpenStack rather than additions to source code. This will involve replacing OpenStack's networking module Neutron with FLIPS to control data plane routing between virtual machines. The OpenStack distribution will be based on OpenStack Ansible[6]

[5] Pica8, "PICA8: Programmable Internetworking &Communication Architecture, Infinite(8)", Online: http://pica8.com

[6]

Co-funded by the Horizon 2020 Framework Programme of the European Union

2.3.3 Bristol Infrastructure Product

2.3.3.1 Bristol Staging Infrastructure Specification

The purpose of the infrastructures is to support the acceptance testing of Platform Product and Media Service Products on the BRISTOLOPEN Infrastructure Product.

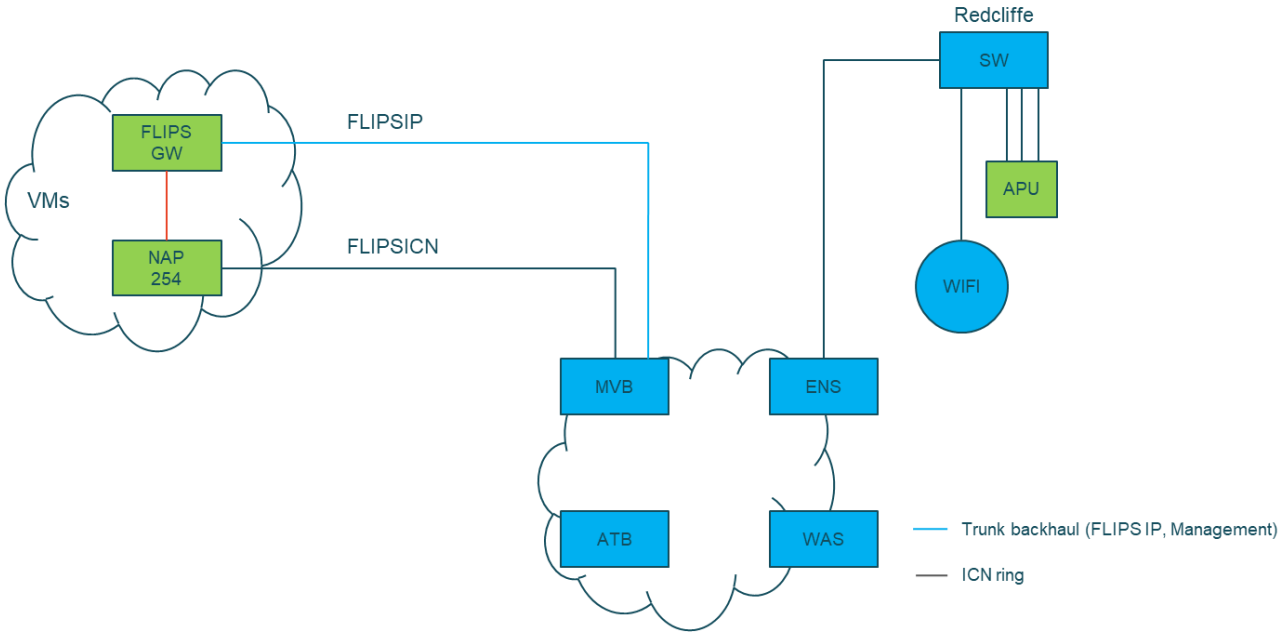

Figure 8: Bristol staging infrastructure configuration

Resource	Capacity	Availability Constraints
Compute	1 x OpenStack Compute Node, Intel 2630V3s currently with 2x8 cores, 16GB RAM but hardware specification under review.	These resources are shared across projects.
Storage	750GB HD RAID 1	These resources are shared across projects.
Networking	4x 48 port NEC 5459 fibre switches 1 x 10 port Brocade switch	These resources are shared across projects.

Table 6: Bristol staging infrastructure resource specification

Staging is intended to mirror the key aspects of the functionality within the city so it has edge connectivity and compute storage but is generally only required to allow initial testing of areas prior to deployment. As such once an experimenter has done some basic testing on staging they are migrated to the production platform.

2.3.3.2 Bristol Production Infrastructure Specification (BRISTOLOPEN)

The purpose of the infrastructures is to support real-life trials and experiments to explore the acceptance, viability and performance of FLAME products in Bristol. This diagram is a current snapshot showing the locations with edge storage only, BRISTOLOPEN is currently deploying LTE into the harbour side area and installing a high-performance computer.

Co-funded by the Horizon 2020 Framework Programme of the European Union

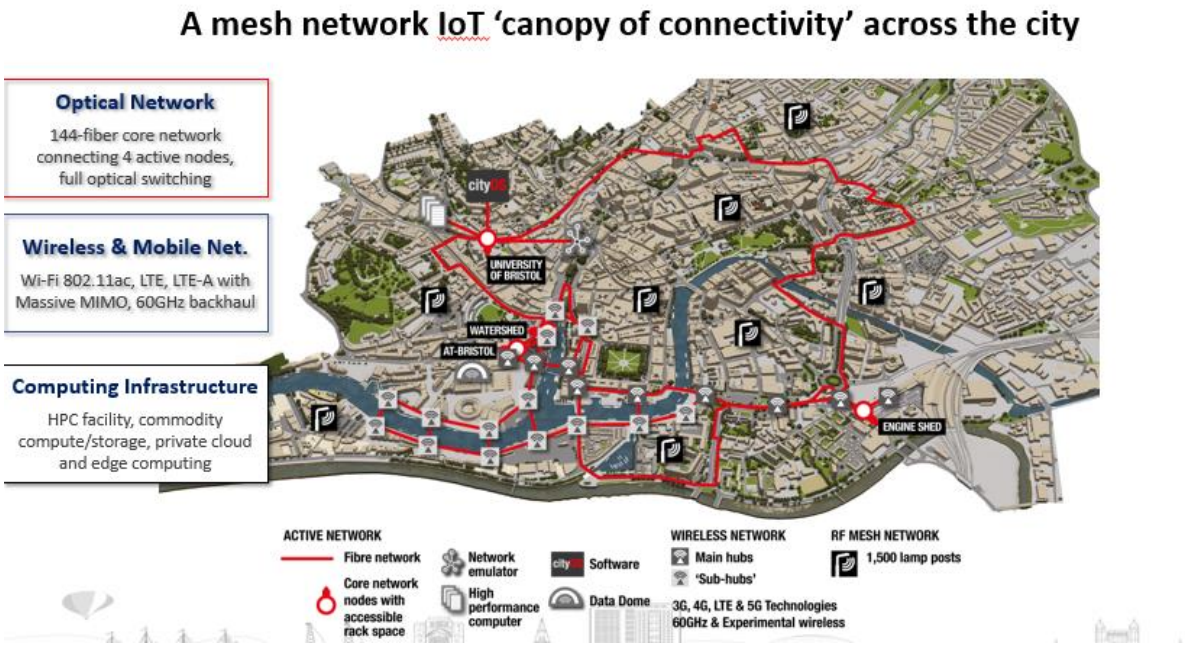

Figure 9: Bristol production infrastructure configuration

Resource	Capacity	Availability Constraints
Compute	4 x OpenStack Compute Nodes, Installed across 4 locations around the city each with 2680v3 x 2 48 cores, 192GB RAM	These resources are shared across projects.
Storage	1TB storage per location	These resources are shared across projects.
Networking	4x48 port NEC 5459 fibre switches. Brocade 10 port SDN cabinet switch. Wi-Fi connectivity and LTE.	These resources are shared across projects.

Table 7: Bristol production infrastructure resource specification

BRISTOLOPEN is investigating the expansion of the network to include 4 new active nodes, and a high-performance computer and is currently deploying LTE access. Within some existing street cabinets there is limited edge computing, but this is not shared across projects. BRISTOLOPEN is investigating the deployment of shared Mobile edge computing capabilities in 2 locations in the city centre to aggregate traffic. Mobile edge computing capability will be subject to discussions with site owners.

2.3.4 Barcelona Infrastructure Product

2.3.4.1 Barcelona Staging Infrastructure Specification

The purpose of the infrastructures is to support the acceptance testing of Platform Product and Media Service Products on the Barcelona Infrastructure Product. The Barcelona staging infrastructure configuration emulates the scenario deployed in the production environment (see Figure 10). All the on-street hardware devices (edge cabinet and lampposts) will be deployed in a controlled environment at i2CAT premises providing a full clone of the production infrastructure.

Resource	Capacity	Availability Constraints
Compute	Cloud: i2cat cloud resources subject to the experiment requirements and cloud resource availability. At minimum two medium size servers. Edge (emulates cabinet): 128GB RAM, 12 cores	Cloud resources shared with the production infrastructure

Co-funded by the Horizon 2020 Framework Programme of the European Union

Resource	Capacity	Availability Constraints
Storage	Cloud: 2TB SSD Edge (emulates cabinet): 2x9600GB SSD	Cloud resources shared with the production infrastructure
Networking	Cloud: 3xPronto TN3290 switches; 10Gbps wired connectivity Edge (emulates cabinet): 10Gbps wired connectivity On-street equipment emulation: Up to 5 WLAN devices installed in the laboratory premises: • Access network: IEEE 802.11n • Multi-hop backhaul network: IEEE 802.11ac • 10Gbps wired connectivity to the cabinet (only for control/management traffic) • 1 to 2 devices also with 10Gbps wired connectivity to the cabinet for data traffic	

Table 8: Barcelona staging infrastructure resource specification

Barcelona site will offer a small size validation testbed placed at i2cat premises. This facility will be used to test and validate all FLAME offerings at the Lab level. Next step is to deliver the same set up on a real-life environment, i.e. Pere IV Street, where the FLAME offerings will be examined against real life traffics and situations.

2.3.4.2 Barcelona Production Infrastructure Specification

The purpose of the infrastructures is to support real-life trials and experiments to explore the acceptance, viability and performance of FLAME products in Barcelona

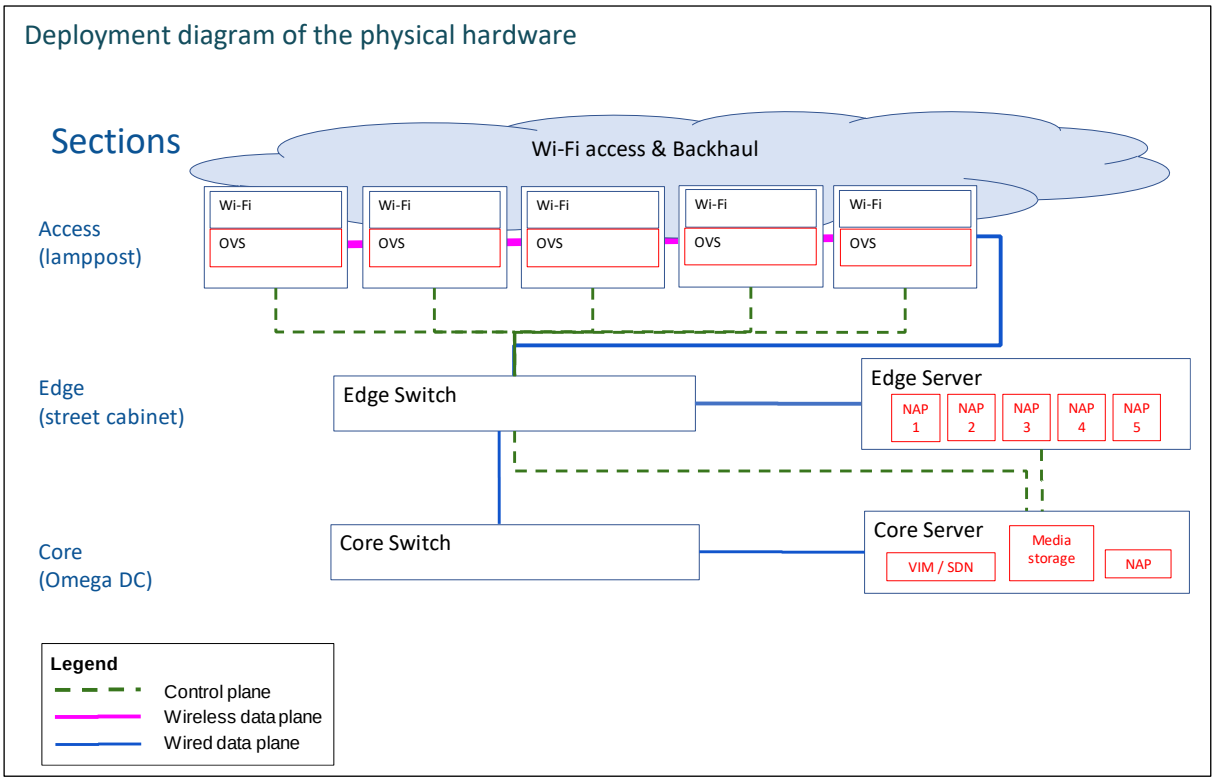

Figure 10: Barcelona production infrastructure configuration

Co-funded by the Horizon 2020 Framework Programme of the European Union

Resource	Capacity	Availability Constraints
Compute	Cloud: i2cat cloud resources subject to the experiment requirements and cloud resource availability. At minimum two medium size servers. Edge (cabinet): 128GB RAM, 12 cores	Core resources shared with the staging infrastructure
Storage	Core: 2TB SSD Edge (cabinet): 128GB RAM, 12 cores	Core resources shared with the staging infrastructure
Networking	Cloud: 3xPronto TN3290 switches; 10Gbps wired connectivity Edge (emulates cabinet): 10Gbps wired connectivity On-street equipment emulation: Up to 5 WLAN devices installed in the laboratory premises: • Access network: IEEE 802.11n • Multi-hop backhaul network: IEEE 802.11ac • 10Gbps wired connectivity to the cabinet (only for control/management traffic) • 1 to 2 devices also with 10Gbps wired connectivity to the cabinet for data traffic	

Table 9: Barcelona production infrastructure resource specification

The production infrastructure will be deployed in Barcelona during the first stage of the project as a replication of Bristol FLAME infrastructure. Plans to extend the hardware infrastructure in Barcelona is left beyond the scope of the project.

2.4 PLATFORM PRODUCT ROADMAP

A Platform Product offers flexible management and delivery of media services deployed on Infrastructure Products described in Section 2. The Platform Product is the major software outcome of FLAME providing advanced service management through Orchestration, Service Function Endpoint Management and Control, Service Function Routing and Cross Layer Management and Control. The overall benefits of the Platform are delivered through an aggregation of component features.

2.4.1 Platform Components and Features

This section describes the feature roadmap, service function chain, implementation technologies, ownership of components that will form part of the Platform Product. Each service function chain is analysed to determine the background technologies and the expected enhancements and adaptations needed to deliver the features. The Technology Readiness Level[7] is provided to give an indication of the level of work that needs to be completed to ensure the component is ready for integration into the Platform Product.

The ownership and licensing situation for components is identified including 3rd party licenses to identify restrictions on access to the Platform Product. The Platform Product will be distributed as software for deployment on production infrastructures by infrastructure providers initially (e.g. BRISTOLOPEN and i2CAT) and then 3rd parties. The Platform Product will also be made available for evaluation by 3rd parties for evaluation and trials. If restrictions are identified then design and implementation decisions will be needed to isolate such components or seek alternative implementations that are consistent with the usage objectives.

[7] https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf

Co-funded by the Horizon 2020 Framework Programme of the European Union

2.4.1.1 Orchestration

The Orchestration component supports the interaction with Cross-Layer Management and Control (CLMC) and media components, leading to an orchestration of compute, storage, and communication resources, including the suitable configuration for SF endpoint control policies.

The features of the CLMC are defined in Table 10. These features are organised in accordance with the interfaces towards other system components including:

- ETSI NFV MANO APIs, used to receiving and parsing a suitable TOSCA template that outlines the required resources to be orchestrated
- Resource APIs: Used to receiving a suitable TOSCA-based infrastructure resource catalogue that can be used to match against orchestration requests
- Orchestration APIs: Used to supporting the various orchestration frameworks and platforms being utilised for FLAME, specifically those at the infrastructure, platform and media services level. Figure 11 shows these levels of orchestration being realised through this feature.

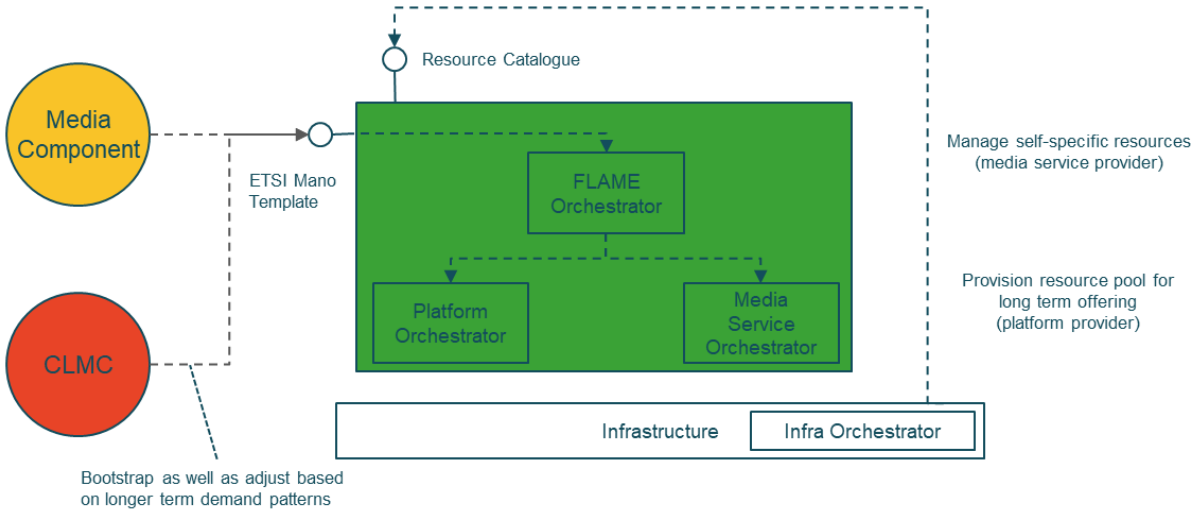

Figure 11: Supporting Orchestration at Different Levels of the overall FLAME system

Feature ID	Req	Feature Description (still high level and may need hierarchy in practice but for this doc high level should be sufficient)	Component Interface	Release
ETSI MANO				
ORCH-1	Req-O1	Provide TOSCA template to OSM-based platform orchestrator	ETSI MANO	Alpha
ORCH-2	Req-O1 Req-O2 Req-O3 Req-I1	Parse TOSCA++, as defined in T4.1, template and check for consistency	ETSI MANO	Beta
Resource				
ORCH-3	Req-I1	Receive TOSCA template as infrastructure catalogue information	Resource	Alpha
ORCH-4	Req-O2 Req-O3	Provide topology information towards SF routing component	Resource	Alpha

Co-funded by the Horizon 2020 Framework Programme of the European Union

Feature ID	Req	Feature Description (still high level and may need hierarchy in practice but for this doc high level should be sufficient)	Component Interface	Release
ORCH-5	Req-I1	Receive TOSCA++ template as infrastructure catalogue information	Resource	Beta
Orchestration				
ORCH-6	Req-O1 Req-O2 Req-O3	Support Docker/container based media service orchestration	ETSI MANO	Beta
ORCH-7	Req-O1 Req-O2 Req-O3	Full consistency check of TOSCA++ template, including consolidating deployment state with orchestration request	ETSI MANO	Beta
ORCH-8	Req-O2 Req-O3 Req-SEM1 Req-SEM3	Provide SF endpoint control policies in TOSCA template extensions towards SFEMC component	ETSI MANO	Alpha
ORCH-9	Req-O2 Req-O3 Req-SEM1 Req-SEM3	Provide SF endpoint state information in TOSCA template extensions towards SFEMC component	ETSI MANO	Beta

Table 10: Orchestration features

The Orchestration critical feature path for the Alpha release is shown in Figure 12. The critical path clusters the features into swim lanes and shows the dependencies, including dependencies with other components of the platform. The Orchestration component delivers features to the SF Endpoint Management and Control as well as SF Routing component as part of the overall orchestration process.

The delivery is expected to be organised around key interface features. The SF Endpoint management and control component is responsible for changing resourcing configurations in response to demands expressed in the orchestration process by TOSCA [ETSINFV] templates being provided to the orchestration component. Said TOSCA templates, which will be based on existing specifications for the alpha release while envisioned to be extended for FLAME-specific requirements (e.g., to support geo-location constraints) in the beta and RC release, are referred to as TOSCA++ in our feature table. The orchestration feature will initially merely separate the management from the control parts in the extended TOSCA template and provide the former to the platform as well as media service orchestrator (see Figure 11). It will utilize existing platforms for those, while providing the latter to the SF Endpoint management and control component for the initialisation of the SF endpoint state. The orchestration feature provides the suitable control policies to the SF Endpoint management and control component, while the resource feature provides the suitable topology information to the SF Routing component.

Co-funded by the Horizon 2020 Framework Programme of the European Union

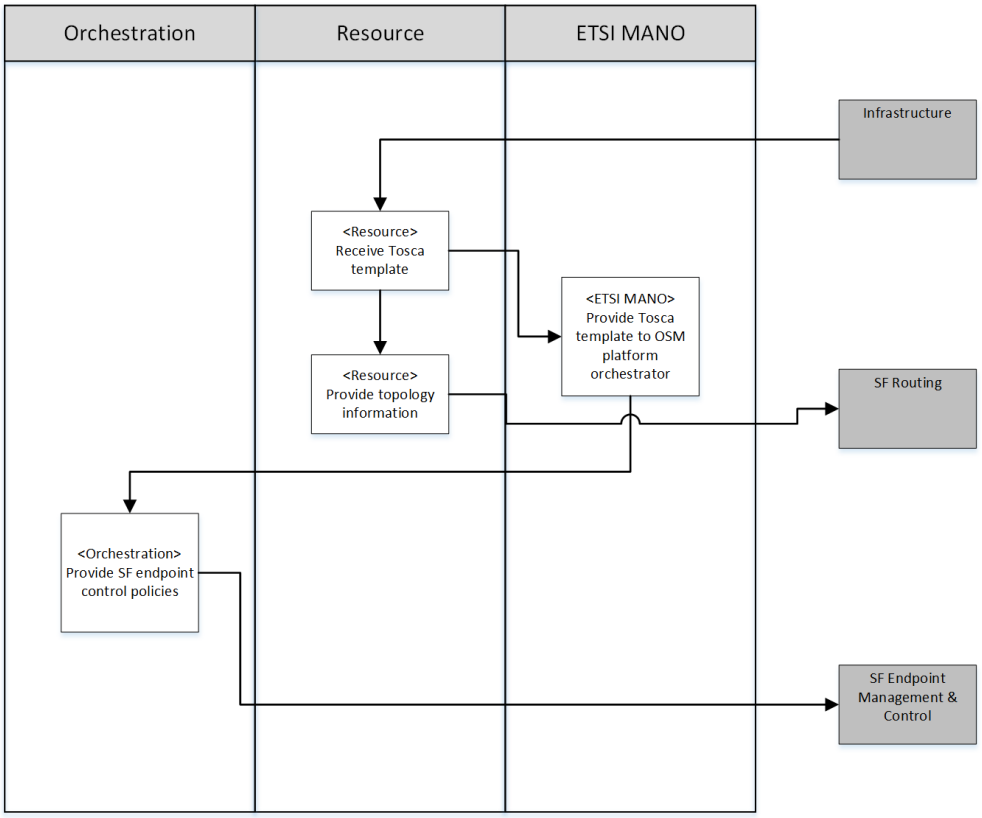

Figure 12: Orchestration Critical "Feature" Path for Alpha Release

The relevant SFC for the alpha release is shown in Figure 13. The main interactions of the orchestration component are illustrated there, i.e., first, the distribution of information derived from the received TOSCA template to sub-components of the SF Endpoint management and control, second, as the SF Routing components, specifically for the SF endpoint control policies, and finally the topology information, obtained through the infrastructure provided resource information. As shown in Figure 11, we expect to distribute the orchestration functionality to existing platforms, such as Open Source MANO [MANO], deployed over separate VMs.

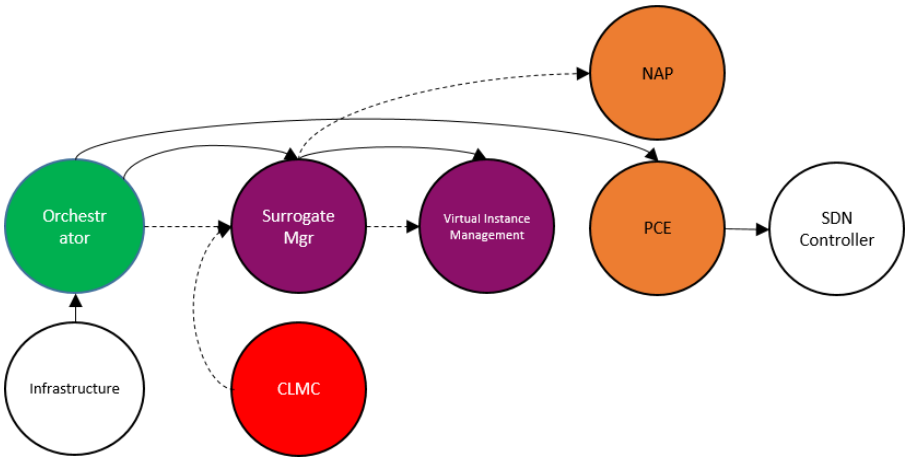

Figure 13: Relevant Service Function Chain for Alpha Release

The Orchestration component will be implemented through adaptation and enhancement to existing open source software that has been developed for the purpose of orchestration, following the split

Co-funded by the Horizon 2020 Framework Programme of the European Union

shown in Figure 11. The key for the alpha phase is to interface with the orchestration platforms developed at the infrastructure level in Bristol and Barcelona. We will align the technology platform used in Bristol for the platform orchestrator, while initially using the same platform for media service orchestration. In later releases, we will move to container-based platforms for media service orchestration.

Table 11 provides a summary of the orchestration implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the orchestration are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
Orchestration	Open Source MANO	ASLv2	Parsing of TOSCA extensions to include control policies	IDE	6

Table 11: Orchestration implementation technology summary

2.4.1.2 SF Endpoint Management and Control (SFEMC)

The SF Endpoint management and control component supports the orchestration process by adding the flexible control capabilities outlined in D3.3 “FLAME Platform Architecture and Infrastructure Specification V1” by maintaining SF endpoint instance state in collaboration with the SF Routing component.

The features of the SFEMC are defined in Table 12. These features are organised in accordance with the interfaces towards other system components including:

- Surrogate Policy Control: Used to receive and parse a suitable control policy from the orchestration component, querying required monitoring data pertaining to such control policy and realising a decision logic that matches the monitored data against the policy provided.
- SF Endpoint Allocation: Used to initialise and maintain an SF Endpoint specific state as well as the compute/storage images that define the SF Endpoint functionality, while also realising delegated name authorisation for the SF Endpoint.

Feature ID	Req	Feature Description	Component Interface	Release
Surrogate Policy Control				
SFEMC-1	Req-SEM1	Parse surrogate policy based on TOSCA template extension	Surrogate Policy Control	Alpha
SFEMC-2	Req-SEM1	Parse surrogate policy based on TOSCA++ template extension	Surrogate Policy Control	Beta
SFEMC-3	Req-SEM1 Req-SEM4	Query monitoring data	Surrogate Policy Control	Alpha
SFEMC-4	Req-SEM1 Req-SEM4	Decision logic matching monitoring data against policy constraints	Surrogate Policy Control	Alpha
SE Endpoint Allocation				
SFEMC-5	Req-SEM4	Initialise and maintain SF endpoint state	SF Endpoint Allocation	Alpha

Co-funded by the Horizon 2020 Framework Programme of the European Union

Feature ID	Req	Feature Description	Component Interface	Release
SFEMC-6	Req-SEM4	Maintain SF endpoint compute/storage images	SF Endpoint Allocation	Alpha
SFEMC-7	Req-SEM2	Allow for delegated name registration for SF endpoint images	SF Endpoint Allocation	Beta

Table 12: SF Endpoint Management & Control Features

The SFEMC critical feature path for the Alpha release is shown in Figure 14. The critical path clusters the features into swim lanes and shows the dependencies, including dependencies with other components of the platform. The SFEMC component delivers features to the SF Routing component as part of the overall orchestration process in general and the control process in particular. The delivery is expected to be organised around key interface features. The SF Endpoint management and control component is responsible for changing resourcing configurations in response to demands expressed in the orchestration process by having received the suitable control policies from the orchestration component. The surrogate policy control feature will establish suitable monitoring capabilities aligned with the surrogate policy constraints defined. It will also realise the decision logic to match the monitored data against said policy constraints. The SF endpoint allocation feature maintains the SF endpoint state according to the control policy provided while utilizing the SF Routing component for service routing related state changes of the SF endpoint.

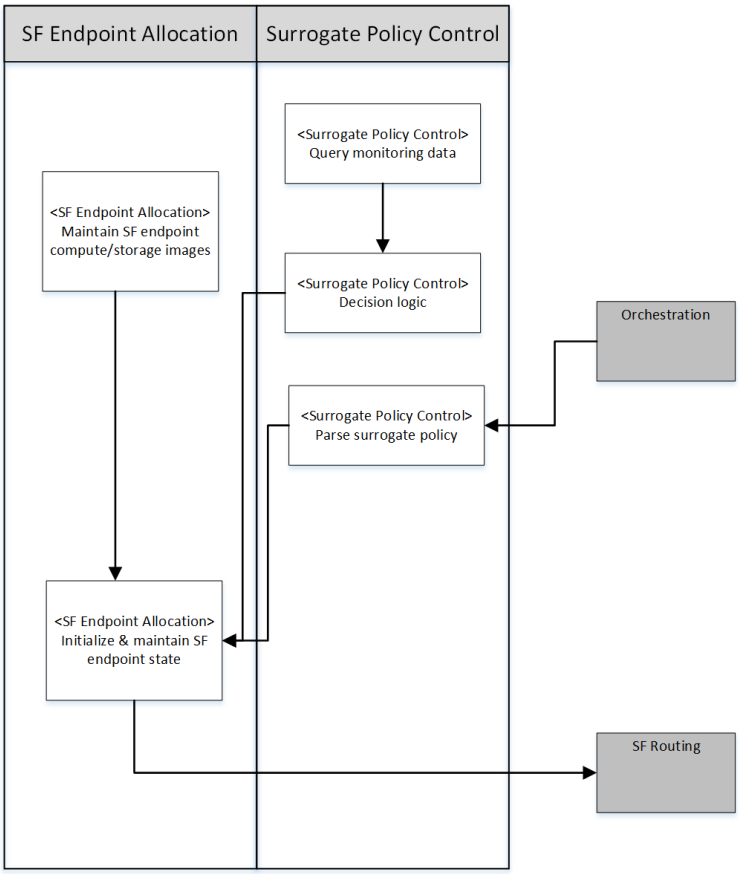

Figure 14: SF Endpoint Management & Control Critical "Feature" Path for Alpha Release

The relevant SFC for the alpha release is shown in Figure 13 with the surrogate manager SF representing the surrogate policy control and SF Endpoint Allocation features of the SFEMC in Table 12. The VIM SF represents the functionality being used by available virtual instance platforms, such as

Co-funded by the Horizon 2020
Framework Programme of the European Union

OpenStack or Docker Swarm, in our realisation. As can be seen, we foresee the interaction between orchestration component and SFEMC to be realised between our extended functionality (i.e., the orchestration and the SFEMC features of Table 12), while realising the initialisation and maintenance of the SF endpoint state through suitably interfacing with existing VIM solutions. While an ultimate deployment of the SFEMC would foresee a single VM for this purpose, it is likely to utilise several VMs for the FLAME-specific extensions and the re-used VIM parts.

Table 13 provides a summary of the orchestration implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the orchestration are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
Surrogate Manager	FLIPS	Access via Consortium Agreement	Realization of features according to alpha feature table	IDE	6
VIM	OpenStack	ASLv2	Integration	IDE	6

Table 13: SFEMC implementation technology summary

2.4.1.3 Service Function Routing (SFR)

The SF routing component realises the service request routing at the data plane between media components, including all operational and management features for supporting route changes, registration of SF endpoints, etc.

The features of the SFR are defined in Table 14. These features are organised in accordance with the interfaces towards other system components including:

- *Protocol mapping*: Used to terminate IP-based protocols at the ingress of the FLAME network, mapping onto Layer2 only transactions and restoring the IP-level interactions at the egress of the FLAME network.
- *Routing*: Used to support various constraint-based routing decisions as well as manage the topology and forwarding information used for the data plane, including the assignment of IP addresses towards media components in the FLAME platform.
- *Registration*: Used to support the registration of fully qualified domain name (FQDN) based services.
- *Resource management*: Use to support link failover and QoS through traffic classes
- *Diversity support*: Used to support multi-source retrieval, net-level indirection as well as in-session switching for HTTP
- *Mobility*: Used for support direct path mobility of users as well as NAP mobility use cases
- *Security*: Used for support encryption at the data plane as well as failure recovery for logically centralised sub-components.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Feature ID	Req	Feature Description	Component Interface	Release
Protocol Mapping				
SFR-1	Req-SR1 Req-SR2 Req.SR3	Implement HTTP level protocol mappings according IDE specifications for HTTP-over-ICN	HTTP/IP	Alpha
SFR-2	Req-SR1	Implement IP level protocol mappings according to IDE specifications for IP-over-ICN	HTTP/IP	Alpha
SFR-3	Req-SR1	Implement IP multicast protocol mappings according to IDE specifications for IP-over-ICN	HTTP/IP	Beta
Routing				
SFR-4	Req-SR7	Support for shortest path routing	HTTP/IP	Alpha
SFR-5	Req-SR8	Support for geo constrained routing	HTTP/IP	Beta
SFR-6	Req-SR9	Support for policy routing	HTTP/IP	Beta
SFR-7	Req-SR7 Req-SR8 Req-SR9	Parse topology information model	Topology target	Alpha
SFR-8	Req.SR1 Req.SR2	Support for topologies larger than 256 links	Topology target	Beta
SFR-9	Req.SR1	Managed DHCP-based IP address assignment	HTTP/IP	Beta
Registration				
SFR-10	Req.SR11	FQDN registration based on configuration	FQDN Registration	Alpha
SFR-11	Req.SR11	FQDN registration based on registration distribution protocol	FQDN Registration	Beta
Resource Management				
SFR-12	Req.SR10	Support for traffic classes based on protocol classes or FQDN	HTTP/IP	Beta
SFR-13	Req.SR12	Support for link failure through path updates	HTTP/IP	Alpha
Diversity support				
SFR-14	Req.SR13	Support HTTP in-session switching	HTTP/IP	Alpha
SFR-15	Req.SR5	Support HTTP multi-source retrieval	HTTP/IP	Beta
SFR-16	Req.SR4	Support HTTP net-level indirection	HTTP/IP	Beta
Mobility				
SFR-17	Req.SR6	Support UE-level inter-NAP mobility	HTTP/IP	Alpha
SFR-18	Req.SR6	Support NAP mobility	HTTP/IP	Beta
Security				
SFR-19	Req-SR3 Req-S1	Support for HTTPS & TLS	HTTP/IP	Alpha
SFR-20	Req.SR1	Support against PCE failure	HTTP/IP	Beta
SFR-21	Req.SR1 Req.SEM2	Support for FQDN authority delegation	HTTP/IP	Beta
SFR-22	Req.SR1 Req.SEM2 Req.SR3 Req.S1	Support for manual content certificate distribution	HTTP/IP	Alpha
SFR-23	Req.SR1 Req.SEM2 Req.SR3 Req.S1	Support for automatic content certificate distribution	HTTP/IP	Beta

Table 14: SF Routing features

Co-funded by the Horizon 2020 Framework Programme of the European Union

The SFR critical feature path for the Alpha release is shown in Figure 15. The critical path clusters the features into swim lanes and shows the dependencies, including dependencies with other components of the platform. The SFR component delivers features to the media component in the form of data plane connectivity at the level of HTTP/IP based protocols. The delivery is expected to be organised around key interface features. The SF routing component is responsible for realising such data plane connectivity based on the availability of SF endpoints in the FLAME network and the current conditions of the transport network, e.g., in the form of available links being available. For this, the routing feature parses the topology information model provided by the orchestration component to suitably configure the infrastructure component, while initially providing shortest-path routing functionality to the protocol mapping feature. The latter realises the media component facing IP protocol termination and mapping onto Layer 2 protocol exchanges. It utilises the registration information realised by the registration feature, while providing the basis for in-session switching for HTTP, realised by the diversity support feature, and for encryption support, provided by the security feature.

Table 15 provides a summary of the orchestration implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the orchestration are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
NAP & PCE	FLIPS	Access via Consortium Agreement	Realization of features according to alpha feature table	IDE	6
SDN Controller	FloodLight	ASLv2	Integration	IDE	6

Table 15: SFR implementation technology summary

Co-funded by the Horizon 2020 Framework Programme of the European Union

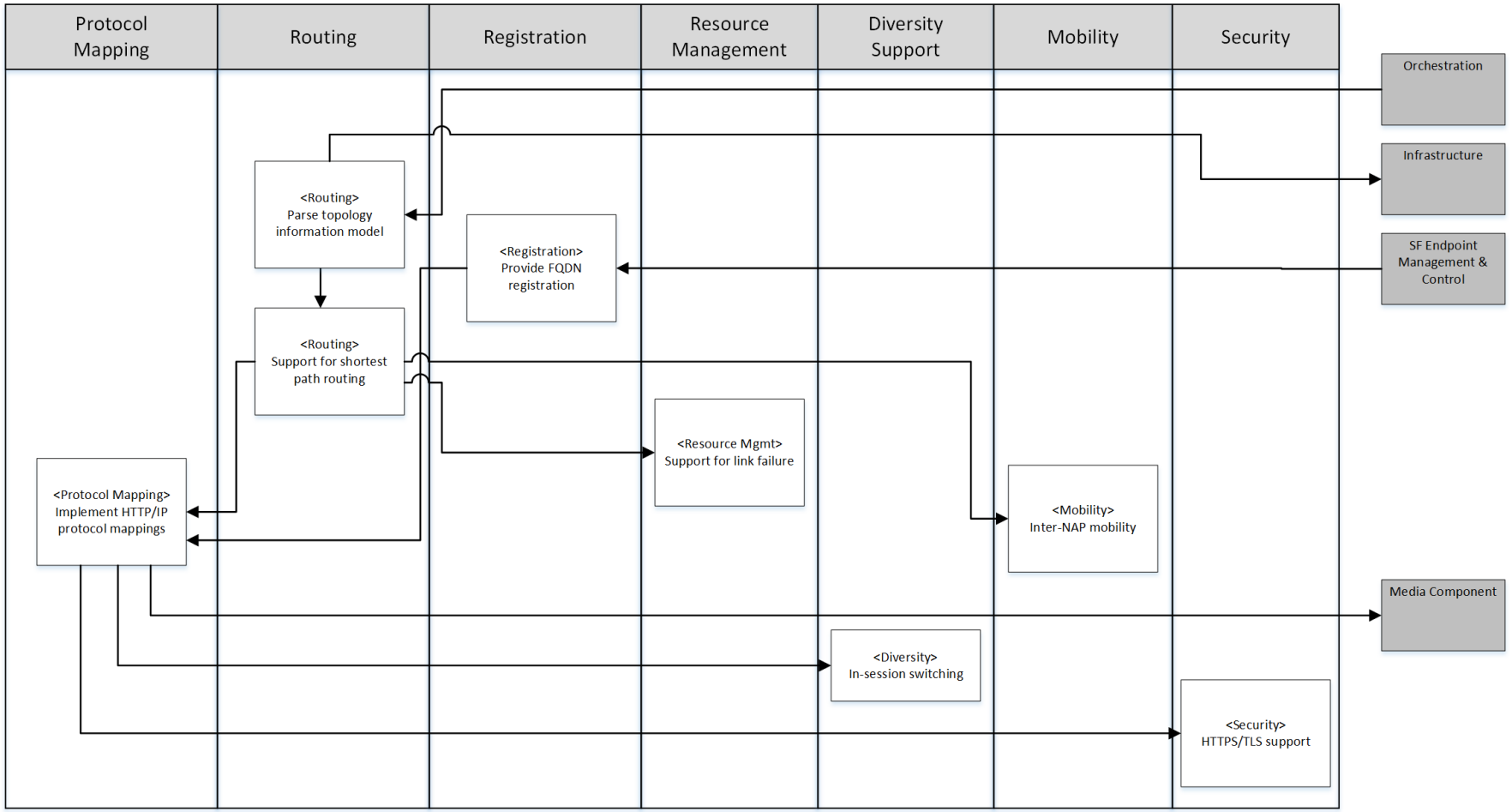

Figure 15: SF Routing Critical "Feature" Path for Alpha Release

Co-funded by the Horizon 2020 Framework Programme of the European Union

2.4.1.4 Cross Layer Management and Control

The CLMC component supports the monitoring, measurement and assessment of media service and platform performance, in addition to configuration of processes supporting the integration and organisation of data for analytics.

The features of the CLMC are defined in Table 16. These features are organised in accordance with the interfaces towards other system components including:

- *Config*: Used to monitor changes in media service configuration including the lifecycle of a media service instance, media components and service functions
- *Monitoring*: Used to monitoring the state and performance of media services including all configured items (media service instance, media components and service functions) and the performance of underlying infrastructure resources allocated to them
- *Analytics*: Used to integrate and aggregate higher-level facts about media service KPIs and dimensions
- *Query*: Used to query, filter and visualise media service and platform information
- *KPI monitoring*: Used to specify and monitor specific performance metrics of interest
- *Security*: Used to control access to information to those that are authorised to do so.

Feature ID	Req	Feature Description	Component Interface	Release
Config				
CLMC-1	Req-C1	Define media service information model	Config	Alpha
CLMC-2	Req-C1	Define configuration information model including failure taxonomy	Config	Alpha
CLMC-3	Req-C1	Store configuration data	Config	Alpha
CLMC-4	Req-C1	Monitor media service lifecycle config events	Config	Alpha
CLMC-5	Req.C6	Monitor SF lifecycle config events (incl geolocation) for NAPS, hosts and service function instances	Config	Alpha
CLMC-6	Req.C2	Flexible configuration of dimensional data abstractions	Config	RC
Monitoring				
CLMC-7	Req-C1	Define monitoring information model	Monitoring	Alpha
CLMC-8	Req-C1	Monitoring data acquisition for media component, service function endpoint and service function routing	Monitoring	Alpha
CLMC-9	Req-C1	Store monitoring data	Monitoring	Alpha
CLMC-10	Req-C1	Delete monitoring data	Monitoring	Alpha
Analytics				
CLMC-11	Req.C2	Basic monitoring data aggregation functions	Analytics	Alpha
CLMC-12	Req.C2	Dimensional data abstraction across (time, space, content representation, content navigation, resource configuration, etc.).	Analytics	Beta
CLMC-13	Req-C2	Define data quality model for accuracy, completeness, timeliness and consistency	Analytics	Beta

Co-funded by the Horizon 2020 Framework Programme of the European Union

Feature ID	Req	Feature Description	Component Interface	Release
CLMC-14	Req.C2	Generation of new media service templates with human in the loop	Analytics	Beta
CLMC-15	Req.C2	Generation of new media service templates through machine learning	Analytics	RC
Query				
CLMC-16	Req.C2	Query monitoring data	Query	Alpha
CLMC-17	Req.C2	Visualise monitoring data	Query	Alpha
CLMC-18	Req.C2	Query by KPIs and dimensions	Query	Beta
KPI				
CLMC-19	Req.C2	Specification of KPIs for measured facts	KPI	Alpha
CLMC-20	Req.C2	Monitor KPI events based on measured facts	KPI	Alpha
CLMC-21	Req.C2	Monitor KPI events based on aggregated facts	KPI	Alpha
CLMC-22	Req.C7	Publish and subscribe to KPI events	KPI	Alpha
Security				
CLMC-23	Req.C3	Define data subject information model	Security	Alpha
CLMC-24	Req.C3	Define information security model	Security	Alpha
CLMC-25	Req.C3	Query for data related to a data subject	Security	Alpha
CLMC-26	Req.C3	Delete data related to a data subject	Security	Alpha
CLMC-27	Req.C5	Secure communication of personal data	Security	Beta
CLMC-28	Req.C5	Restricted access to personal data	Security	Beta
CLMC-29	Req.C7	Restrict access to stakeholder viewpoints on monitoring data	Security	Beta

Table 16: CLMC features

The CLMC critical feature path for the Alpha release is shown in Figure 16. The critical path clusters the features into swim lanes and shows the dependencies, including dependencies with other components of the platform. The CLMC implementation depends on the specification for a media service. According to the D3.3 architecture:

> *"[Media Service] Specification of the descriptors required for the definition, deployment and management of Media Services, including dynamic behaviours that can be explored within experimentation, testing and operations. Specification-Language-compliant Templates will be available for the Media Service Providers to make the definition of Media Service easier. The Specification Language will take into account current orchestration specs for cloud environments, such as TOSCA."*

The media specification provides the logical configuration structure for a media service. This structure defines context for monitoring information acquired when the media service is operated. The structure offers key relationship between information whilst the logical naming of service functions will allow for monitoring data to be integrated through the use of correct references. The overall naming scheme for items within the media service specification is a critical input for different aspects of the CLMC information model.

The CLMC delivers features to all other Platform components. The delivery is expected to be organised around key interface features. The orchestrator and SF Endpoint management and control components are responsible for changing resourcing configurations in response to media service provisioning events and media service demand. These events need to be captured by the CLMC to

Co-funded by the Horizon 2020 Framework Programme of the European Union

track the changes in service configuration over time. For the alpha release the event logs will be limited to key media service and SF lifecycle events. A protocol for reporting configuration events including the message format with pub/sub service implementation is needed. With the configuration in place, the CLMC has the context for monitoring information produced by different system components. The infrastructure, platform and media services are Monitoring Producers that depend on the availability of a pub/sub monitoring pipeline that offers a protocol and a messaging format for monitoring data. Although these two feature streams are related they can be implemented in parallel if the information model is agreed and information exchange is achieved through pub/sub protocols.

Co-funded by the Horizon 2020
Framework Programme of the European Union

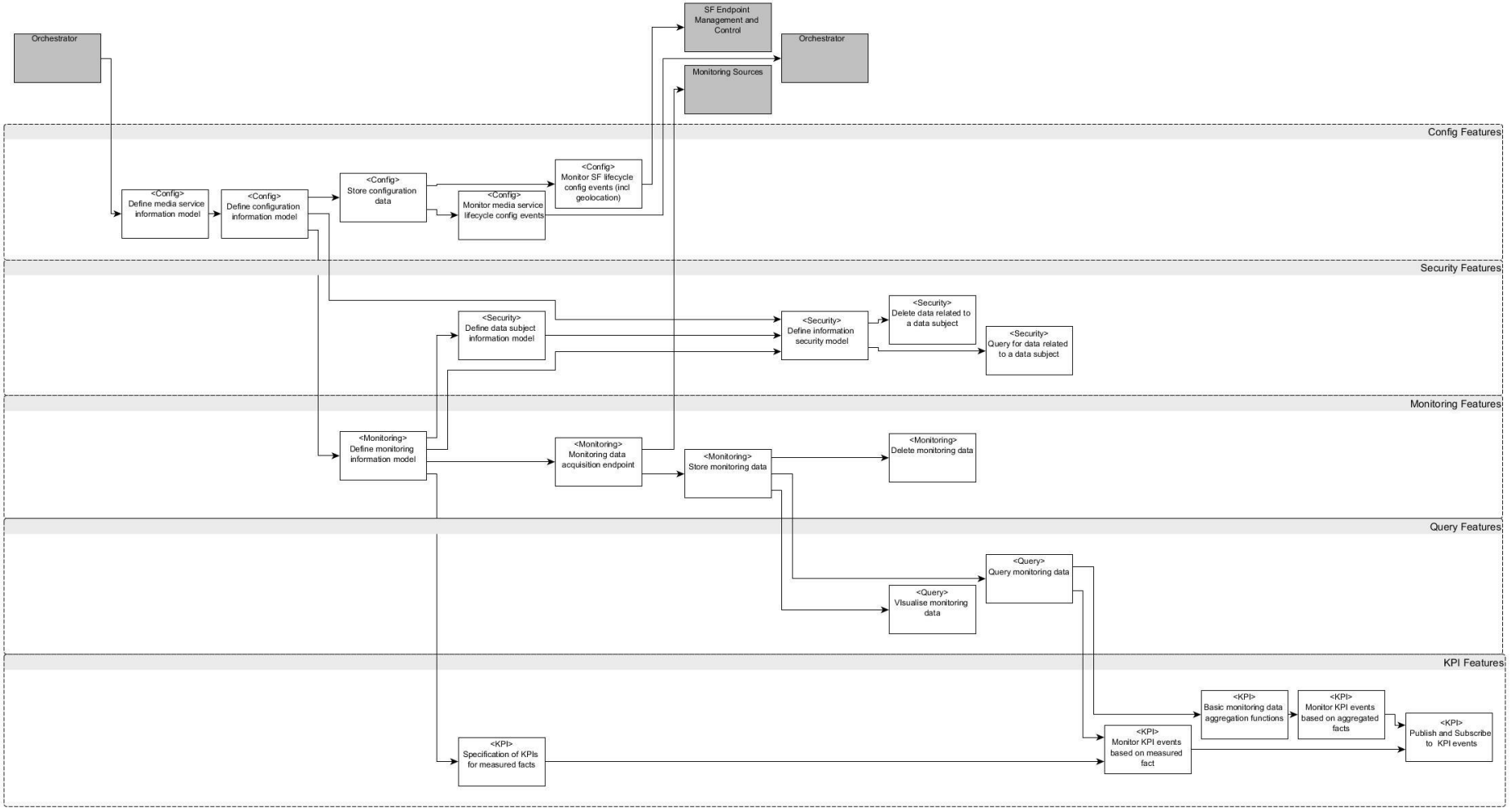

Figure 16: CLMC Critical "Feature" Path for Alpha Release

Co-funded by the Horizon 2020 Framework Programme of the European Union

The SFC for the alpha release is shown in Figure 17. The SFC is a simplified version of that described in the architecture specification and does not include the features such as dimensional analytics and data quality. The SFC is designed to allow SFs instances to be distributed in the same way as we expect for media services. During the development it may be possible to deploy SFCs 4-6 within a single VM. However, for integration, we'd expect these to be distributed across different VMs as the deployment shift towards what would be expected in production.

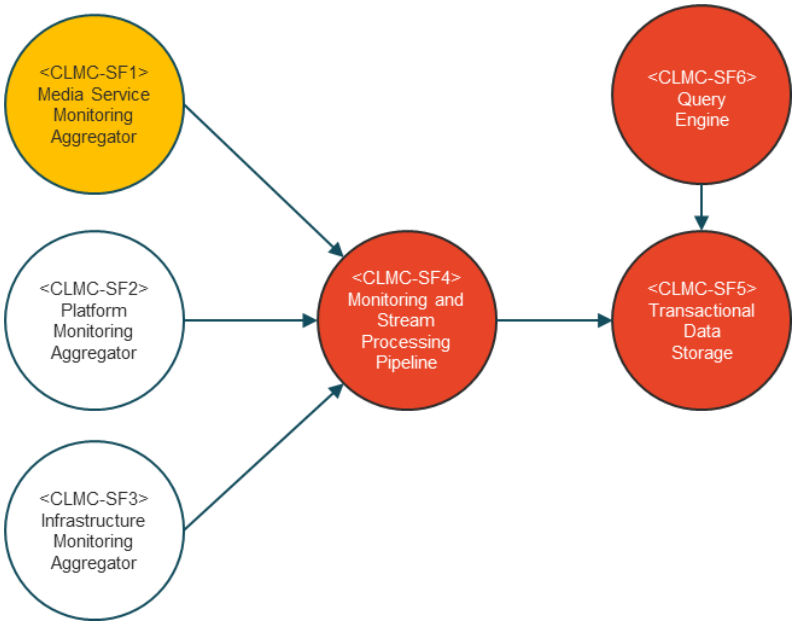

Figure 17: CLMC Service Function Chain for Alpha Release

The CLMC service function chain will be implemented through adaptation and enhancement of an existing open source software that has been developed for the purpose of service monitoring. The key for the alpha phase is to put in place the technologies supporting the acquisition of the data. Higher level analytics can then be implemented in later releases to help improve the way media services are managed. Table 17 provides a summary of the CLMC implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the CLMC are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
CLMC-SF1	TBD Dec-17	TBD	Integration	Atos	6
CLMC-SF2	TBD Dec-17	TBD	Integration	IDE	6
CLMC-SF3	TBD Dec-17	TBD	Integration	I2CAT	6
CLMC-SF4	Apache Kafka	ASLv2	Configuration of topics, monitoring producers, consumers, etc.	ITINNOV	6
CLMC-SF5	InfluxDB	MIT License	Configuration of transactional time series data model; Configuration of time series data aggregation functions	ITINNOV	6

Co-funded by the Horizon 2020
Framework Programme of the European Union

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
CLMC-SF6	Grafana	ASLv2	Configuration of dashboard; Configuration of alerts for notification of KPIs	ITINNOV	6

Table 17: CLMC implementation technology summary

2.4.2 Summary of Platform Releases

Table 18 provides a summary of component features across each platform release.

Component	Alpha Features (Feb-18)	Beta Features (Dec-18)	RC Features (Jul-19)
Orchestration	• Monitor orchestration decisions • Provide TOSCA template to OSM-based platform orchestrator • Receive TOSCA template as infrastructure catalogue information • Provide topology information towards SF routing component • Provide SF endpoint control policies in TOSCA template extensions towards SFEMC component	• Parse TOSCA++, as defined in T4.1, template and check for consistency • Receive TOSCA++ template as infrastructure catalogue information • Support Docker/container based media service orchestration • Full consistency check of TOSCA++ template, including consolidating deployment state with orchestration request • Provide SF endpoint state information in TOSCA template extensions towards SFEMC component	
Service Endpoint Management and Control	• Monitor service function endpoint • Parse surrogate policy based on TOSCA template extension • Query monitoring data • Decision logic matching monitoring data against policy constraints • Initialise and maintain SF endpoint state • Maintain SF endpoint compute/storage images	• Parse surrogate policy based on TOSCA++ template extension • Allow for delegated name registration for SF endpoint images	
Service Routing	• Monitor service function routing • Implement HTTP level protocol mappings	• Implement IP multicast protocol mappings according to IDE	

Co-funded by the Horizon 2020 Framework Programme of the European Union

Component	Alpha Features (Feb-18)	Beta Features (Dec-18)	RC Features (Jul-19)
	according IDE specifications for HTTP-over-ICN • Implement IP level protocol mappings according to IDE specifications for IP-over-ICN • Support for shortest path routing • Parse topology information model • FQDN registration based on configuration • Support for link failure through path updates • Support HTTP in-session switching • Support UE-level inter-NAP mobility • Support for HTTPS & TLS • Support for manual content certificate distribution	specifications for IP-over-ICN • Support for geo constrained routing • Support for policy routing • Support for topologies larger than 256 links • Managed DHCP-based IP address assignment • FQDN registration based on registration distribution protocol • Support for traffic classes based on protocol classes or FQDN • Support HTTP multi-source retrieval • Support HTTP net-level indirection • Support NAP mobility • Support against PCE failure • Support for FQDN authority delegation • Support for automatic content certificate distribution	
CLMC	• Define media service information model • Define configuration information model • Store configuration data • Monitor media service lifecycle config events • Monitor SF lifecycle config events (incl geolocation) for NAPS, hosts and service function instances • Define monitoring information model • Monitoring data acquisition for media component, service	• Dimensional data abstraction across (time, space, content representation, content navigation, resource configuration, etc.) • Define data quality model for accuracy, completeness, timeliness and consistency • Generation of new media service templates with human in the loop • Query by KPIs and dimensions • Secure communication of personal data • Restricted access to personal data	• Flexible configuration of dimensional data abstractions • Generation of new media service templates through machine learning

Co-funded by the Horizon 2020 Framework Programme of the European Union

Component	Alpha Features (Feb-18)	Beta Features (Dec-18)	RC Features (Jul-19)
	function endpoint and service function routing • Store monitoring data • Delete monitoring data • Basic monitoring data aggregation functions • Query monitoring data • Visualise monitoring data • Specification of KPIs for measured facts • Monitor KPI events based on measured facts • Monitor KPI events based on aggregated facts • Publish and subscribe to KPI events • Define data subject information model • Define information security model • Query for data related to a data subject • Delete data related to a data subject	• Restrict access to stakeholder viewpoints on monitoring data	

Table 18: Platform feature roadmap

2.5 MEDIA SERVICE PRODUCT ROADMAP

2.5.1 Media Services Overview

A media service product is a software product offering content production, management and/or distribution capabilities. Media service products are integrated, tested and packaged including a default template specification for deployment on a FLAME platform to create media services. A media service product is dependent on one or more media component products implementing underlying service functions within the overall media service function chain [FLAME-D3.3].

A media service product is no more than a set of media components described in terms of topology, performance and resourcing using templates. Media services themselves are not part of the FLAME platform but are deployed and managed by it. The FLAME platform orchestrates the deployment of media components as well as internal service functions. Throughout the project the goal is to build an initial set of foundation media services and then extend the available media services through developments conducted by 3rd parties.

Co-funded by the Horizon 2020 Framework Programme of the European Union

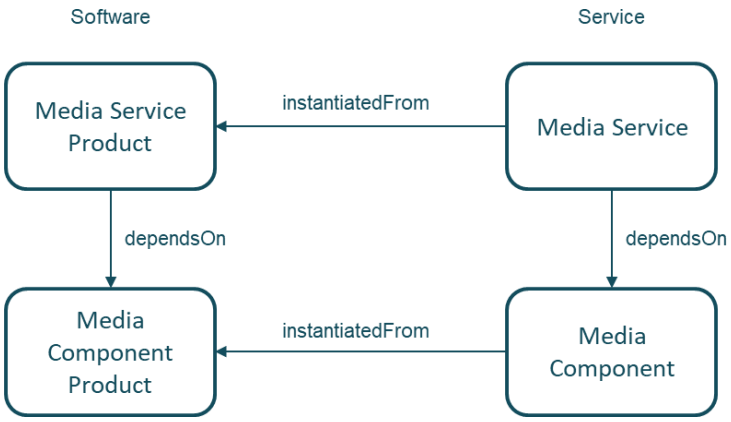

Figure 18: Media services and media components

2.5.2 Media Service Packaging

The packaging of the media services impacts in the way the media service products are made available. Although the decisions could be refined in a near future, we aim at a container-based deployment of media services to ensure service start up delay is minimised. The container technology will be based on Docker[8]. The orchestration is not implemented as a single piece of software but rather as a collaboration between a Platform Orchestrator and a Media Services Orchestrator. The selected technology for the media services orchestration is Kubernetes, as it is proven to be the most successful orchestration platform for containerised services. The definition of media services for Kubernetes is achieved with the use of Charts. Charts lets users define how different subservices (Docker images) of the whole media service are configured and linked together. We propose the use of one Charts per media service, but charts are composable and reusable, so we may consider hierarchies and collections of them. FLAME is considering OASIS TOSCA, with the required extensions, for the description of the media services via templates, which will contain all the required information for a full deployment.

2.5.3 Media Component Products

FLAME has defined a list of media component products that offer common capabilities necessary to construct media service products. For example, a content conditioning process will require transcoding and trans-rating media components. The initial media components products are used to create FLAME's "foundation media services" providing examples of capabilities that benefit from the FLAME platform. The foundation media components and services form part of the FLAME offering.

Due to the number and variety of the foundation media components criterion have been established to select components to be implemented. Firstly, the prioritisation process has carefully analysed the validation scenarios proposed in FLAME. D3.1 – FMI Vision, Use Cases and Scenarios describes the mentioned validation scenarios [FLAME-D3.1]

- Participatory media for interactive radio communities (City Fame)
- Personalised media mobility in urban environments (Follow Me)
- Collaborative interactive transmedia narratives (Interactive Storytelling)

[8] https://www.docker.com/

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Augmented reality location-based gaming (Gnome Trader).

Some of the foundation media services have been specifically defined to cover modules of these validation scenarios, as described in D3.3 FLAME Platform Architecture and Infrastructure Specification V1 [FLAME-D3.3]. Secondly, the prioritisation of the media services has considered the suitability of the media services to show the advantages of the FLAME platform benefits and technical innovations as described in Section 2.5 of D3.3. Finally, the prioritisation process has considered the terms for the different FLAME releases and the development requirements for each service. Additionally, a certain foundation media service can include different implementations along the project (e.g., involving different existing software initiatives) with the respective temporal terms.

Table 19 identifies the media components that fulfil the requirements of modules identified in the FLAME validation scenarios. These foundation media components can be clustered according to their functionality and the step they cover in the production and distribution chain, as detailed later.

ID	Name	Description
MC-1	Media content database	This service consists in a generic database, which is a required module in most of media services. Thus, the four FLAME validation scenarios include a database, as shown in FLAME deliverable D3.3. For example, the City Fame validation scenario contains a database to store and provide knowledge about states and historically measured data required to generate user profiles, interest groups and the relationship among them. Some complex media services require the stateful replication of a synchronised database. For example, a certain service may require a replicated media database in the edge to improve the availability of media contents. FLAME benefits and technological innovations enable a very efficient procedure for the replication of the database in the deployment.
MC-2	Media Quality Analysis	The objective of this service is the evaluation of the media characteristics of a certain media asset. These characteristics include data such as the resolution, the video and audio codecs and also an automatic estimation of the quality. This last functionality is required to determine the suitability of contents provided by prosumers. This is the case, for example, of the City Fame validation scenario.
MC-3	Content Ingest	This media component enables the insertion of media assets to make them available in a media service. This component will satisfy two different functionalities. On one hand, it will enable the provision of contents to deploy an experiment. In this case, the service is used before the experiment deployment. For instance, a media service provider may want to test a Video-On-Demand (VoD) service using the FLAME platform. This service would allow the provider to “upload” the assets. On the other hand, this service will enable the ingest of content as a part of an experiment, as in the City Fame scenario. In this case, the service is used during the experiment itself.
MC-4	Content Storage	This media component is in charge of storing the media assets for the provision of the services. This kind of functionality is widely required by media services. This is for example the case of a video on demand service. This component satisfies the requirements of the content provisioning module in the Follow me scenario, among others.
MC-5	Content Caching Management	This component is in charge of managing the content replication for the assets availability. For example, this component would be used by a virtual CDN to control the existence of repositories along the network, since a CDN works as a hierarchy of caching. This component satisfies the functionalities of the Caching manager module in the Interactive storytelling scenario.
MC-6	Content Management System	The content management system or CMS is a widely used media component that supports the creation and modification of digital content. These systems usually offer a web user interface to control the existence and availability of media assets.
MC-7	Content Conditioning	This component is in charge of processing the media assets to make them available. For example, assets must be split in chunks and encoded at different bitrates to offer a video-on-demand adaptive streaming service.

Co-funded by the Horizon 2020 Framework Programme of the European Union

ID	Name	Description
MC-8	Transcoding and Transrating	Transcoding consists in the change of the video or audio specification to represent the content of an asset. This kind of encoding is named source encoding. For example, this component would be in charge of converting an AVC/H.264 video clip into a HEVC/H.265 one. The use of a more recent specification (and this is the case of HEVC with regard to AVC) enables a reduction of the bitrate required to represent the information while it preserves the subjective quality. Transrating is a similar process but in this case the encoding specification does not change. It typically consists in an additional source encoding to reduce the bitrate (this processing will cause a reduction of the quality, too).
MC-9	Adaptive Streaming	Adaptation is the process that allows a player to take into account the network (and the receiver) capabilities to automatically and instantaneously adapt the transmitted bitrate (and the quality) in a streaming service. In this way, adaptive streaming optimises the instantaneous quality along the asset duration. The adaptive streaming service in FLAME plans two different implementations: one for video on demand assets and another one for live content.
MC-10	Virtual CDN	This service consists in the creation of a CDN using virtual nodes to optimise the advantages of this kind of networks, such as bitrate, low latency, load balance and scalability. CDNs perform caching of data to enable faster access by the end users. Moreover, CDNs approach content to end users with high availability and high performance. Video distribution networks are typically CDNs. FLAME benefits, via Platform products, enable new ways of implementing CDNs.
MC-11	Adaptive Data Transmission	This component extends the mentioned adaptive video concept to other kinds of data transmission. For example, a certain media service could require the transmission of 3D models to be rendered in the user equipment or in AR applications. This component optimises the bitrate (and quality) of the transmission of this additional data. The Gnome trader scenario requires this kind of functionality.
MC-12	Metadata Transmission and Management	Metadata consists of data that describe the media assets. Media systems usually manage this information to enable the deployment of services. An example of metadata is the asset information in the different validation scenarios. This component is close related to other media component and services, like the content management system (CMS).

Table 19: Media component products

As stated in the description of these media services and components, several of them are related. The mentioned list tries to present them in an order that reveals the relationships between the services. Two main clusters can be distinguished:

- Services and components related to content management and processing. This category includes: content ingest, content storage, content caching management, content management system and content conditioning.
- Services and components related to information transmission and distribution. This category includes: adaptive streaming, virtual CDN, adaptive data transmission and metadata transmission and management.

2.5.4 Releases and Media Services Product Implementation Roadmap

Table 20 summarises the roadmap for the implementation of media service products in FLAME according to the prioritisation criteria explained in the previous subsection.

Component	Alpha	Beta	Comments
Media content database	Yes	Yes	All the validation scenarios contain a database. It can show FLAME benefits.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Component	Alpha	Beta	Comments
Media quality analysis	No	Yes	This service is planned for the beta release.
Content ingest	No	Yes	Although this service is planned for the beta release, the method and protocols for content ingest will be analysed for the alpha release
Content storage	No	Yes	This service is planned for the beta release.
Content caching management	No	Yes	This service is planned for the beta release. This service will take advantage of key FLAME capabilities for content replication.
Content management system	No	Yes	This service is planned for the beta release.
Content conditioning	Yes	Yes	A first version of this service will be available in alpha release. This service will allow the automatic encoding of the audio-visual content at a variety of bitrates and resolutions and the processing of the media assets to enable adaptive streaming services.
Transcoding and transrating	Yes	Yes	Different versions of this component are planned for both alpha and beta releases, depending on 1) software to perform the encoding and 2) solution conceived for live content or video on demand. Concerning the first characteristic, we propose two different software initiatives: Wowza and FFMPEG. Wowza is a commercial and consolidated product for the deployment of adaptive streaming services, including transcoding and transrating whereas FFMPEG is an open source initiative that provides a variety of encoding tools. FLAME will offer the encoding formats covered by these external tools. Particularly, for the beta release, FLAME will include the new and efficient HEVC encoding format. With regard to the second characteristic, the key difference is due to the fact that live content requires encoding on the fly, which may need large computing capabilities. On the other hand, video-on-demand transcoding is performed off-line and it does not include real-time requirements. The alpha release will include transcoding and transrating performed by Wowza for live content and transrating performed by FFMPEG on H264 contents for video on demand. The beta release will extend the supported formats. The use of these two different tools is also conceived to enable different exploitation models in the future (with or without commercial, external software).
Adaptive streaming	Yes	Yes	A first version of the adaptive streaming service will be available in the alpha release. The service will be refined for the beta release. The alpha release will be based on Wowza. Different tools may be integrated in the beta release, according to the evolution of available streaming initiatives. This service will support different adaptive streaming technologies and particularly MPEG-DASH and HLS. Other technologies (HDS and Smooth Streaming) will be also available if required.
Virtual CDN	No	Yes	Virtual CDN. FLAME proposes an innovative implementation based on FLIPS, one of the new technologies involved in FLAME. The design of this implementation is planned for the medium term. For this reason, this component will be ready for the beta release.
Adaptive data transmission	No	Yes	This service is planned for the beta release.
Metadata management and transmission	No	Yes	This service is planned for the beta release. The implementation will be close related to the content management system.

Table 20: Media services release plan

Co-funded by the Horizon 2020 Framework Programme of the European Union

3 SUPPORTING TRIALS AND EXPERIMENTATION

FLAME aims to deliver a new media service deployment platform based on highly distributed software-defined infrastructures. The development process is grounded in the idea that the platform will be tested for function, performance and acceptance through a series of trials and experiments conducted on real-life infrastructures.

The design, implementation and execution of trials and experiments is supported by the platform APIs and tooling. As discussed in Section 6.1.3 of D3.3, FLAME's management and control is designed to support different contexts of use. The term "experiment" describes a context of use and that the management and control lifecycles are equally applicable in DevOps and business intelligence processes. D3.3 states that:

> "*The fact that we are conducting an experiment does not change the capabilities needed to manage and control the system*."

And continues to say experiments are just the motivation setting out objectives and outcomes:

> *"For an experiment, this could be to test a hypothesis or for business intelligence, this could be to investigate performance of a media service within a specific geographic region. In each case, the decision maker explores service management knowledge to understand how to establish better management and control policies in relation to performance criteria."*

What this means is that the features in the technology roadmap are designed to be generally applicable to different situations and not tied exclusively to experimentation. This approach allows FLAME to communicate the management and control features in ways that have more value and meaning to potential adopters of the technologies.

However, the connection between experimentation and management and control lifecycles is important in terms of FLAME's content and the KPIs expected by the project. In the following sub-sections we elaborate this in more details exploring how the features of the platform address the KPIs. Also, not that these KPIs will form part of the Platform product test plan so that the impacts can be provided in terms of measurable benefits.

3.1 EXPERIMENTATION WITHOUT CONSTRAINTS OF PHYSICAL LOCATION

A key barrier to reuse and replication of platforms is the tight coupling with specific physical infrastructures. FLAME addresses this barrier as follows:

Experiment Requirements	FLAME Features
FLAME will provide an experimental toolbox that supports programmatically specifying, controlling and monitoring experiments independent of physical location by exploiting key platform and infrastructure capabilities.	FLAME offers an experiment toolbox through features of the Orchestrator (TOSCA specification) and the CLMC (monitoring). Through TOSCA templates and infrastructure abstractions SUT can be controlled and monitored independently of physical location.
Building on current software defined technologies for network and cloud management, and defining an IaaS specification and capacity model for the target experiments, FLAME will provide the mechanisms	FLAME's architectural infrastructure abstraction based on common standards and specifications, along with integration and testing across different Infrastructure Products ensures technical replication at different locations

Co-funded by the Horizon 2020 Framework Programme of the European Union

Experiment Requirements	FLAME Features
necessary for replication locations to be created that are compatible with the FLAME platform.	
By addressing provisioning of mobile edge and data centre resources interconnected with software defined communications, FLAME will deliver content distribution and delivery that exploits the cloudification of networks, surrogate management and flexible service routing to improve QoE for consumers and manage QoS for operators.	FLAME's KPI and dimension information model within the CLMC allows for measurement and analysis of cross layer function and performance. The focus on KPIs and dimensional measurements in the experiment design provides the use cases and scopes testing of measure procedures for different characteristics of the system under test
Through a VSN specification language that includes experimentation constraints associated with important FMI abstractions geography, population, localisation, experimenters will be able to specify once and repeat experiments at different real-world locations	FLAME extends the TOSCA specification to include additional constraints that can be used to orchestrate media services at different geo-graphic locations

Table 21: How FLAME's features support experimentation independent of physical location

The KPIs defined for experimentation independent of location are defined in Table 22. These KPIs must form part of the detailed integration test plan that's outlined in Section 4.3.

KPI ID	Description	Target (by the end of the project)
FLAME F2.1	Virtual Service Network Specification Language for FMI experiments	1 specification language measured by peer reviewed publication and contribution to relevant spec/standards organisation (OASIS, ETSI, etc.)
FLAME F2.2	Experimentation toolbox for offline specification, real-time control and monitoring	1 toolkit available for use by FIRE+ and FMI experimentation communities.

Table 22: KPIs for trials and experiments without constraints of physical location or access to a specific experimental facility

3.2 REDUCTION IN EXPERIMENTATION TIME

Mechanisms and approaches adopted in FLAME to reduce the time required to experiment are listed in the following table.

Experiment Requirements	FLAME Features
FLAME will significantly reduce the time to experiment through a series of acceleration methodologies and supporting tools supporting the experimentation lifecycle (design, specification, provisioning, control, observation, analysis).	Features for TOSCA++ templates, management, control, monitoring and analytics features support the full lifecycle of experimentation. This covers features of the Orchestrator, CLMC, SF Endpoint Management and SF Routing.
Media services need real-time interactive behaviours that are supported by reliable, reconfigurable infrastructures offering performance guarantees. FLAME's FMI Experimentation Instance templates will be designed to support target media service workflows expected in the FMI such as those identified in Validation Experiments.	Features for TOSCA++ template support the specification of media service and SFCs. The validation experiments have been analysed to produce a set of SFC use cases for the platform, as documented in D3.3. This report provides a roadmap for the foundation media services to be integrated in support of the validation experiments.
Adaptable templates will accelerate experiment design and development, and will allow for consistent FMI performance knowledge to be captured in different situations. These insights into the operation	Features of the information model includes common specification of media service structure and resource specification through TOSCA++ whilst the definition of KPIs and dimensions (spatial, content format,

Co-funded by the Horizon 2020 Framework Programme of the European Union

of the infrastructure and platform will be captured, annotated and disseminated through an FMI Performance Knowledgebase ensuring that the patterns of usage are shared within the experimentation community. The knowledgebase will underpin provisioning policies and allow for definition of target provisioning times for FMI Experimentation Instances.	content representation, etc.) provides a consistent way of capturing and analysing FMI performance knowledge. The CLMC will persists data across multiple trials and experiments, whilst analysis of this data leads to the generation of improved media service templates. Ownership of the data is governed by data access policies and access is made based through licensing agreements.
Other acceleration measures will include preconfigured media analytics pipelines for experiment monitoring, toolkits support experiments designed to explore QoS/QoE in real-life environments, mentoring and dedicated tutorials for SMEs/Start-ups and a replication process that incorporates best practice governance and operations models for operators.	Features of the CLMC allow for pre-configuration with measurement procedures in accordance with the information model ensuring that users of the platform can explore and configure easily the types of cross-layer data available for understanding the behaviour of interactive media systems. All software products will be documented with and automated scripts established for build, provisioning and configuration of software products.
In addition, reconfiguration functions at different layers (infrastructure, platform, application) will be assessed to define runtime response times with a target error rate. To ensure reliability of the platform, a taxonomy of failure models will be identified to track the performance of different experimentation resources.	SF endpoint management and SF routing features support reconfiguration of SF instances in response to surrogate policy constraints. The Orchestrators TOSCA++ templates allow for re-specification of constraints. The CLMC's configuration interface supports notification of failure events associated with media service and SF lifecycles.

Table 23: How FLAME's features support reduction in experimentation time

The KPIs defined for reducing experimentation time are defined in Table 24. These KPIs must form part of the detailed integration test plan that's outlined in Section 4.3.

KPI ID	Description	Target (by the end of the project)
FIRE+ P1.5	Cost reduction declared by the experiments	At least 25%
FLAME F3.1	Automated provisioning of large-scale experiments across a variety of requirements (PIML-orientation)	FMI Experimentation Instance templates supporting tailoring and adaptation >=5
FLAME F3.2	Ave service-level latency for mobile interactive media services	<=5m secs
FLAME F3.3	Ave, Min, Max Provisioning Time for Standard FMI Experimentation Instances	Response Time <=60 secs
FLAME F3.4	Ave, Min, Max Real-Time Control (Reconfiguration) Functions Response Time specified for infrastructure and platform	Response Time <=30 secs
FLAME F3.5	Network capacity through traffic localization and multicast delivery in HTTP-based services	10x increase
FLAME F3.6	Concurrency for multi-tenancy sets of experiments	Experiment Concurrency >=5
FLAME F3.7	Taxonomy of platform reliability and failure states for FMI experiments	1 Deliverable and >=1 peer reviewed publication
FLAME F3.8	FMI Performance Knowledgebase providing benchmarks for cross layer operations	>=30 experimentation data sets
FLAME F3.9	FMI Performance Knowledgebase Access: Target=>100 data access requests per annum)	>=100 data access requests per annum)
FLAME F3.10	FMI open data sets published in repositories for H2020 Open Data Pilot	>=5 open data management packages with >=20 downloads

Table 24: KPIs for reduction of the time to experiment

Co-funded by the Horizon 2020 Framework Programme of the European Union

4 SYSTEMS INTEGRATION

4.1 INTEGRATION STRATEGY

The integration strategy is designed to deliver software products that are of sufficient quality to conduct urban scale trials on infrastructures deployed in real-life environments. The strategy aims to maximise software quality within the cost constraints associated with testing, including both the capital costs of testing infrastructure and labour costs of writing, maintaining and conducting the tests themselves. Where possible the strategy adopts software engineering best practices to achieve efficiency, consistency of results and to deal with the complexity of integrating complex software products.

The strategy considers the full pipeline from unit testing of components through to the launch of the EaaS offering on infrastructures. Continuous Integration (CI) is at the centre of the approach where contributions from developers are merged and integrated frequently to help reduce integration problems in software products.

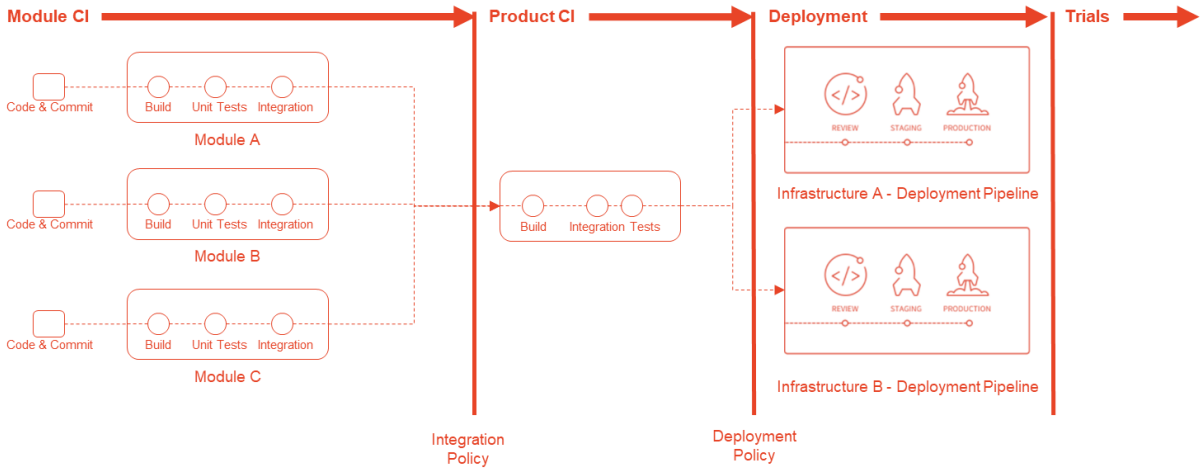

Figure 19: FLAME continuous integration and deployment pipelines

The high level view of the FLAME integration and deployment pipeline is shown in Figure 19. The pipeline is split into three phases:

Phase	Outcome	Activities	Multiplicity
Deployment	Operational systems deployed on a production infrastructure	Reviewing products; Acceptance testing on staging infrastructure; Deploying products on production	Multiple deployment pipelines, one for each FLAME Replicator. Includes Bristol and Barcelona then expanding to three further Replicators.
Product CI	Products to be deployed on production infrastructures	Build, configuration and integration testing	Multiple integration pipelines, one for each product. Includes Infrastructure Products, Platform Product and Media Service Products
Component CI	Software components contributing to products	Coding, unit testing, module integration testing	Multiple integration pipelines, one for each module. Includes modules for each Product, e.g. CLMC,

Co-funded by the Horizon 2020 Framework Programme of the European Union

Phase	Outcome	Activities	Multiplicity
			Orchestrator, FLIPS for the Platform Product

Table 25: FLAME continuous integration pipeline phases

Each phase in the pipeline is connected by workflows and policies that control the exchange of artefacts. The workflows are designed to ensure efficient working between distributed multi-organisation development teams. The policies ensure that the artefacts shared between each phase meet a minimum set of requirements for the next phase.

4.2 PROJECTS, WORKFLOWS AND TOOLING

Activities within the CI pipeline are conducted within the context of Projects. Projects provide the management tools for prioritisation of features, organising development work, and triggering test suites. Projects will be managed within GitLab[9] which offers software project management capabilities for code content management, issue tracking and continuous integration. Table 26 summaries the main capabilities of GitLab.

Activity	Description
Git Repository	Content management system used to manage collaborative development and release of software.
Issue Tracking	Used to track features and issues related to software developed within the project. Issues can be labelled and organised into Issue Boards to review and schedule development
Continuous Integration	Used to automatically trigger or schedule unit and integration tests on target environments. CI pipelines are defined through .gitlab-ci.yml in the projects route directory. Note, in the community edition that multi-project CI is not supported.

Table 26: Software project management tools

The Platform Product is made up of different distinct components as described in the Platform Product road map (see Section 2.4).

Project	Description	Owner	Expected Contributors
Platform	The parent project responsible for delivering Platform Product releases to deployment infrastructures	IT Innovation	ALL
Orchestrator	A component project responsible for delivering the Orchestrator to the Platform Product	IDE	Atos, UNIBRIS, Martel
CLMC	A component project responsible for delivering the CLMC to the Platform Product	IT Innovation	IDE, Atos, I2CAT
FLIPS	A component project responsible for delivering the FLIPS to the Platform Product	IDE	None

Table 27: Platform project and component projects

The Platform Product CI pipeline uses a multi-project structure to support the development and integration (see Figure 20).

[9] https://about.gitlab.com/

Co-funded by the Horizon 2020 Framework Programme of the European Union

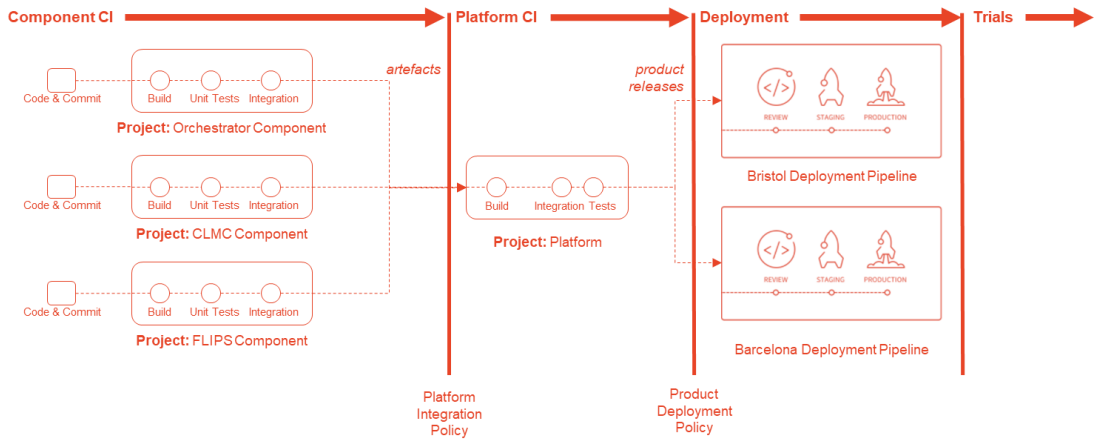

Figure 20: Platform product integration pipeline

4.2.1 Platform project

The platform project:

- Manages features and release schedules for the Platform product
- Integrates components through dependency management with artefacts deliverable by component projects and 3rd parties
- Operates a continuous integration pipeline including build, provision, configuration and integration test suites
- Releases the platform product to the deployment infrastructures

The platform product is released as a set of packages that can be built, provisioned and configured on downstream production infrastructure operating OpenStack and an SDN fabric supporting OpenFlow (see Figure 21). The integration infrastructure replicates this process and tests scenarios against the integration infrastructure defined in Section 4.4. The high level steps in the integration process are shown in Figure 22. Each component project delivers software packages for integration by publishing into an accessible package repository. The packages conform to a set of requirements (see Section 4.2.3) for platform integration. Integration then implements a process to build, provision, configure, test and release the products.

The ***Build step*** creates VM images in a way that optimally targets the infrastructure used. This process of deciding on the relation between VM images and packages is a deployment decision and should not be done within the component projects themselves. If component projects released VM images that would limit the flexibility in how the platform is provisioned and configured. It is however expected that developers within the component projects are responsible for VM image build scripts used within the platform project and not the entire responsibility of the Platform project owner. This is reflected in the distribution of resources across the FLAME work plan.

Co-funded by the Horizon 2020 Framework Programme of the European Union

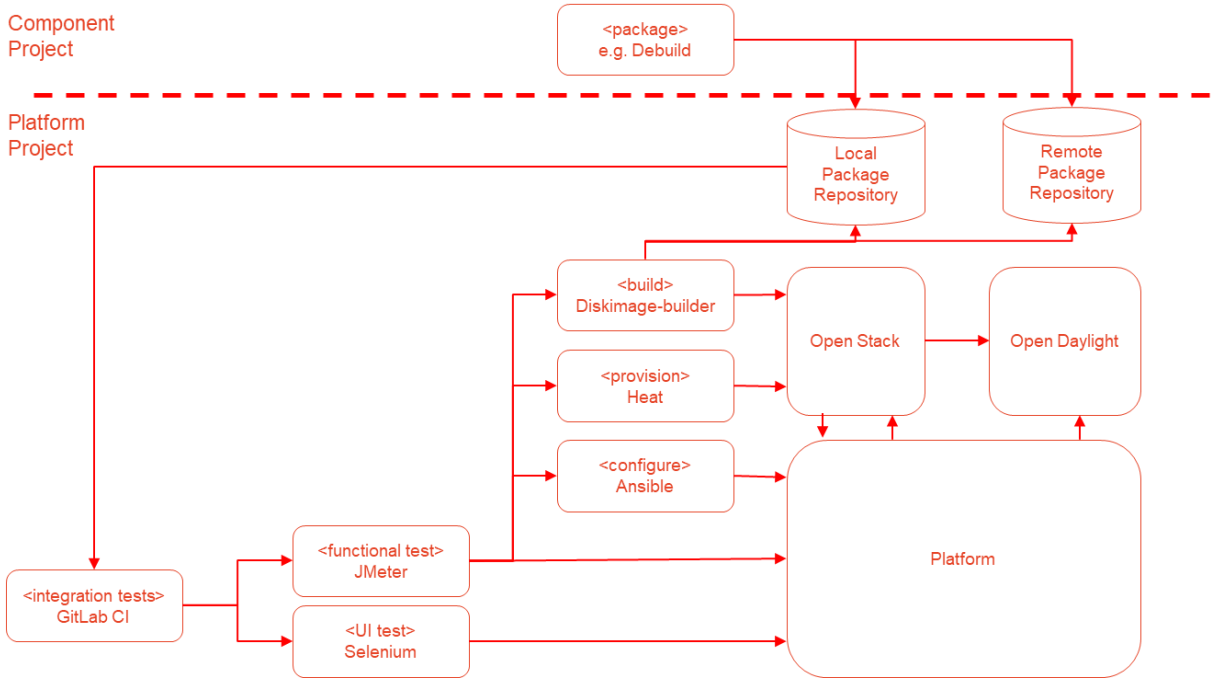

Figure 21: Platform integration tools

The ***Provision step*** creates a Platform stack on an OpenStack deployment according to the needs of one or more integration test scenarios. The provisioning is achieved using the Heat orchestrator[10] through the definition of Heat templates that describe the infrastructure allocated to Platform VMs. Heat then manages the full lifecycle of allocating the infrastructure and provisioning the Platform.

The ***Configure step*** configures the Platform components within a stack for a specific test scenario. Configuration is achieved using Ansible[11] through Playbooks supporting the management of configuration on remote services.

The ***Test step*** includes the execution of the integration test suite on a built, provisioned and configured platform. The tests conducted are designed according a test plan and the defined integration test scenarios covering functional, load and UI testing. The testing framework is expected to be JMeter for functional and loading testing, and Selenium for UI testing primarily focused on any management dashboards provided to users of the Platform.

Finally, the ***Release step*** packages the Platform Product for distribution to deployment sites.

The content within the Platform project repository will consist primarily of scripts to build, provision and configure the platform and an integration test suite. The integration test suite will implement a test plan designed to ensure an acceptable level of quality in product releases. The test plan will include a set of test scenarios that provides an acceptable test coverage for the product features. The test coverage will be defined based on the requirements in the technical roadmap and will be checked by the stakeholder deployment infrastructures to ensure acceptance on delivery of the product.

[10] https://wiki.openstack.org/wiki/Heat

[11] https://www.ansible.com/

Co-funded by the Horizon 2020 Framework Programme of the European Union

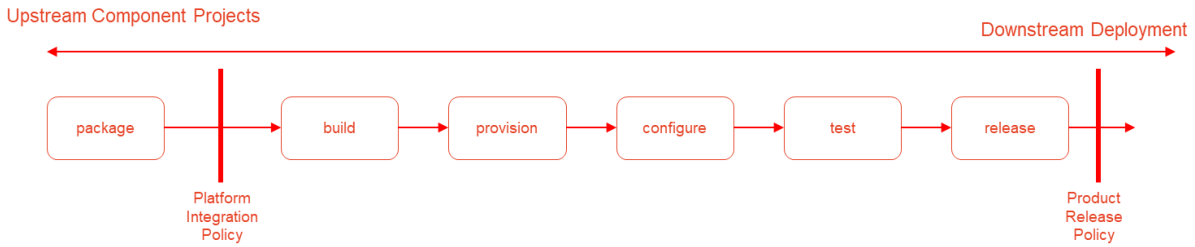

Figure 22: High level product integration process

The workflow for the management of content within the repository and product release is shown Figure 23. The diagram shows a set of branches each created to incrementally integrate features and bug fixes towards stable releases. The approach is based on the Git feature workflow where for the Platform product features reflect test scenarios to be built, provisioned, configured and tested. The workflow includes two main branches (master and develop) and a set of supporting branches for feature, release and hotfix. The purpose and rules associated with each type of branch are described in Table 28.

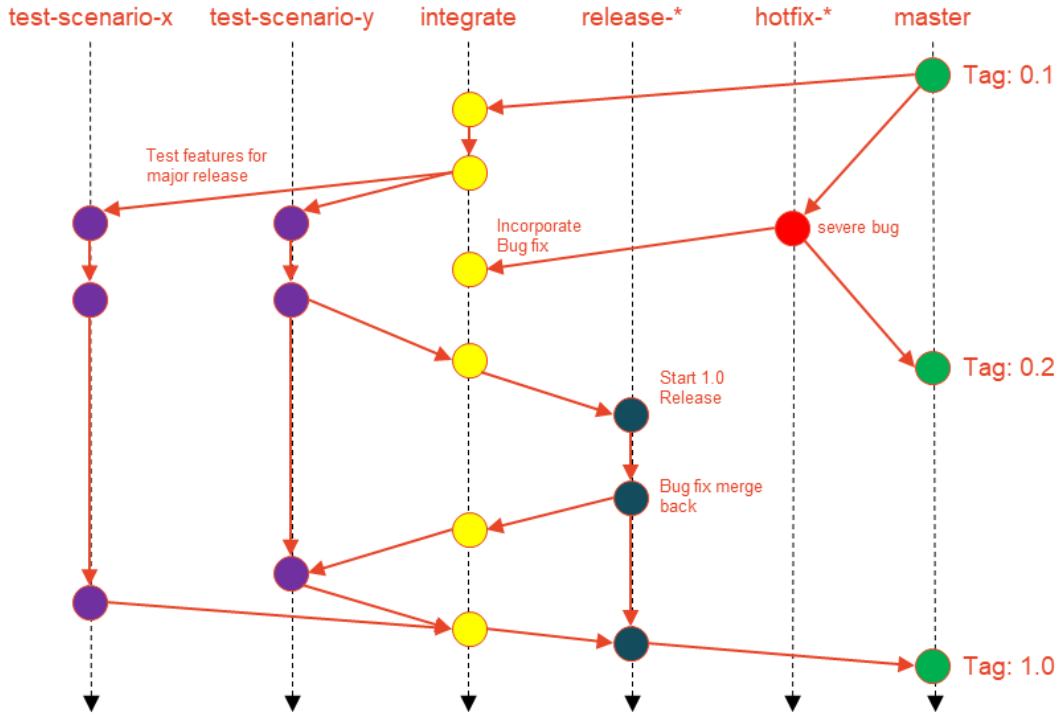

Figure 23: Platform project workflow

Name	Purpose	Branch off	Merge back	Naming
master	The main branch where the code of HEAD always reflects a production-ready state	-	-	Master
integration	The main branch where the code of HEAD always reflects a state with the latest delivered development changes for the next release. Used to trigger nightly builds.	master	release-*	Integration
feature	Feature branches are used to develop new scripts for build, provision and config along with integration test scenarios covering	integration	integration	anything except master, develop,

Co-funded by the Horizon 2020 Framework Programme of the European Union

	functional or load tests defined in the test plan.			release-*, or hotfix-*
release	Used to support preparation for production releases such as minor fixes, preparing metadata, allowing integration to continue	integration	integration, master	release-*
hotfix	Used to prepare for an unplanned production release from the necessity to act immediately upon an undesired state of a live production version. Typically this will include integration of an upstream package including a bug fix.	master	develop and master	hotfix-*

Table 28: Platform repository branches

The Platform project has the roles and responsibilities as described in Table 29.

Project Role	Responsibilities	Partner
Project Owner	Responsible for the overall platform project and the delivery of product releases to production. Responsible for the development of test plans and test scenarios in accordance with features within the technical roadmap and the acceptance criteria of deployment infrastructures	IT Innovation
Continuous Integration Monitor	Responsible for monitoring the nightly integration builds and investigating in the morning who caused the integration failures. The frequency of the e	Allocated to integration developers on a weekly rotational basis
Integration Developer	Responsible for developing and maintaining build, provisioning and config scripts, and integration tests for component integration in accordance with the test plan. Also bug fixing following nightly build failures if commits cause the integration to fail.	Test implementation allocated to partners according to roadmap and resources

Table 29: Platform project roles and responsibilities

4.2.2 Component projects

Each component of the Platform is developed within a component project with a dedicated component project operating its own CI pipeline. The outputs of the component project are software packages for integration delivered in accordance with the Platform Integration Policy. Each component project is the responsibility of the component owner and includes contributions by one or more organisations. However, the operation of the component CI pipeline including the development workflows, the schedules for integration and the development environment is all defined by component project owners.

This approach allows for component project owners to manage CI pipelines in ways that suit the policies of contributing organisations. For example, the FLIPS component will only include contributions by IDE and will be released for Platform Integration as a binary. Here IDE can develop FLIPS at organisation level Git repo and releases can be packaged and published to integration package registry. For other components, collaborative development from multiple contributing organisations is expected and the Git repo will be hosted centrally.

Each component project conducts the following activities:

- Manages features and schedule of implementation for the component
- Manages the environments used by developers

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Operates a CI pipeline including build, unit test suits and integration test suits at the component level

- Delivers integrated components as packages to the Platform project

The infrastructures used to develop the components cannot easily replicate the integration or production infrastructures. Where possible approaches are needed to develop and test components within development rather than integration environments. For example, the CLMC can be developed and tested on a develop machine using a Docker environment without any dependency on OpenStack or other platform components such as FLIPS. The key is that the CLMC is designed and implemented to be run as a Platform SF using only supported base images and protocols. Of course, in such development environments the CLMC cannot take advantage of SF routing or management offered by FLIPS, however, the programme logic and integration between different services can still be implemented and tested. The integration with FLIPS can occur within Platform integration.

Ideally the component project workflows should mirror the best practice described for the Platform project (see Section 4.2.1). Each component project should make available the latest build (integration), release candidate (release-*) and production version (master) to the Platform project for integration. The release of a package into the Platform project notifies the Platform project owner who implements the integration policy, e.g. automatically update dependencies and trigger integration testing on the integration branch. In some circumstances, it may be advisable to create a feature branch within the Platform repo for the new component package so that scripts and tests affected can be updated without affecting the integration branch.

Even though it is undesirable, a Platform Product production release may include a release candidate of a component rather than a production release of that component if that is a version that includes a time critical feature or bug fix. However, a platform product release should not include an integration release of a component due to the instability of such packages.

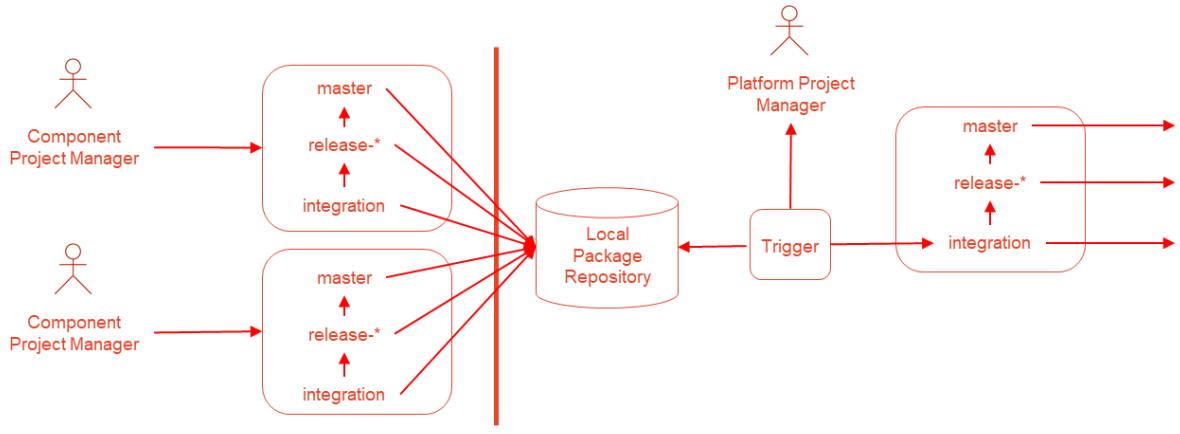

Figure 24: Sample component project workflow

4.2.3 Product Project Policies

Each Product Project implements policies that define the requirements that components must fulfil for integration into FLAME products. The policies included aspects of technical constraints of the environment, configuration scripts, documentation and minimal acceptable testing criteria

ReqID	Description
INT-1	All packages MUST be published in an agreed format to the integration registry.

Co-funded by the Horizon 2020 Framework Programme of the European Union

ReqID	Description
INT-2	Alpha packages MUST be deployable as virtual machines within OpenStack.
INT-3	All packages MUST only communicate using the protocols supported by the FLIPS component.
INT-4	All packages MUST be documented according to the product documentation standards
INT-5	All packages MUST be unit tested to ensure the basic correctness of code. Unit tests are run frequently to detect bugs early on in business logic, so that the developer who introduced the bug can fix it immediately. Unit tests must be well encapsulated and don't use external resources or additional components such as databases and infrastructures. Unit test target small and distinct part of code, they must be simple to write and maintain. Unit testing is the responsibility of the developer, unit testing policy must be enforced by the project owner and automatically triggered as part of the CI pipeline. Unit tests must be kept separate from integration tests (See next section), they should not run together. Developers working on code must be able to run unit tests and get immediate feedback to ensure that they haven't broken anything before committing. If test suites take too long developers are likely to stop running and maintaining tests. This will result in delivery delays due to the effort required to bring unit test suites up to date with the code. The technology used to manage unit test suits depends largely on the programming language and the software development environment. FLAME mandates that unit testing must be conducted but does not mandate any specific unit testing framework.
INT-6	All product pipelines MUST operate a master, stable and development branch and make this available for integration
INT-7	Master branch releases MUST be tagged and versioned using the {major}.{minor}.{patch}, e.g., • 3.2.1 Release branch releases MUST be tagged {major}.{minor}.{patch}-{releaseTag.preRelease}.{commits}, e.g. • 2.2.0-alpha1.50 Development branch release MUST be tagged {major}.{minor}.{patch}-{releaseTag}-{commits}, e.g., • 1.1.0-rc100 – alpha release for 1.0.0. at 100 commits to the development branch Please note that the tagging of software products using development phases (alpha, beta and rc) can be run on a different schedule to the project milestones identified in Section 2.1
INT-8	All packages must include an intellectual property registry and licensing

4.3 INTEGRATION TEST PLAN

4.3.1 Assumptions

- Software inputs
    - All software must be released in accordance with Product Policies defined in 4.2.3
- Test plan
    - Test case design must be focused on meeting the business objectives, cost efficiency, and quality.
    - Test case design must consider the features within the technical roadmap for each release and constraints of production infrastructures
    - Test case design must consider the project KPIs defined in the description of action or any updates subsequently agreed.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- The test plan must be reviewed by production infrastructure owners prior to the start of testing
- Performance tests should provide insights for a limited software-based infrastructure with the expectation of improved performance on production.

- Testing process
  - Testing must be conducted with a common, consistent procedures, yet flexible, with the ability to change as needed.
  - Testing within the CI pipeline must be automated.
  - Testing must be a repeatable, quantifiable, and measurable activity.
  - Testing environment and data must emulate a production as much as possible.

- Resourcing
  - Testing will be incremental building upon previous stages to avoid redundancy or duplication of effort.
  - Testing effort must be managed in close coordination with the project management to ensure appropriate balance between research, innovation and software quality.
  - All team members must have an overall knowledge of product features even if deep knowledge is known by specialists at the component level.

4.3.2 Integration testing

The outcome of the integration tests are software products released to production deployments. Integration includes a test suite for verifying and validating the interaction between components and infrastructures. Integration tests focus on finding faults in the environment and configuration, they do not test the business logic within code as that's the responsibility unit tests. Integration tests span several software components, devices and hardware components, in any functional flow. As a result, if an integration test fails, it is complex to identify the cause. Components must use a logging framework that can be controlled via flags that allow for minimal logging during normal production usage and progressively more detail to be logged in the event of a problem. For some problems, exhaustive logging is the only way to analyse a failure and discover the problem, however, logging can affect performance so it is important to be configurable.

Integration testing is managed using a test plan that defines different test scenarios. The test scenarios execution paths through different platform components necessary to deliver function and performance. There are many possible paths through a complex system so the goal of the test plan is to determine an appropriate test coverage considering the resources available for testing and the desired quality of the software product.

Test Areas	Description	Tooling
Integration	Test the function and interaction between two or more components	JMeter
Spatial configuration	A type of integration test that tests features depending on the relationship between physical and virtual geolocation.	JMeter
Load	Tests the performance of two or more components	JMeter

Co-funded by the Horizon 2020 Framework Programme of the European Union

Test Areas	Description	Tooling
Hardware	Test the function or performance of one or more components on specific hardware	TBD
UI	Tests the function of user interfaces delivered through web browsers	Selenium

Table 30: Integration testing types

FLAME will use an automated integration testing framework to create test plan and execute testing scenarios relevant to production deployments. Tests will be developed, maintained and managed as part of the Platform project.

4.3.3 Test areas

Test areas determine interactions between components that require integration testing. Table 31 shows the high-level test areas for the platform integration and the components involved in the test. Initially FLIPS needs to be tested against the integration infrastructure to ensure SF Routing and Endpoint Management functions correctly. FLIPS can then support management of other Platform SFs (Orchestrator, CLMC). Orchestration can be tested across the full Media Service lifecycle with FLIPS. Finally, both FLIPS and Orchestration can be tested with the CLMC for monitoring and configuration events. The specific tests within each test area will be determined by the features in the technical roadmap.

Test area	Infra	FLIPS	Orch	CLMC	Summary
SF Routing	X	X			SF routing features on the integration infrastructure
SF Endpoint Management	X	X			SF endpoint lifecycle management features on the integration infrastructure
Media Service Orchestration	X	X	X		Media service lifecycle management features on the integration infrastructure
Infrastructure monitoring	X	X		X	Measurement procedures for infrastructure metrics
SF Routing Configuration	X	X		X	Event logging for SF routing configuration
SF Routing Monitoring	X	X		X	Measurement procedures for SF routing metrics
SF Endpoint Management Configuration	X	X		X	Event logging for SF endpoint configuration
SF Endpoint Management Monitoring	X	X		X	Measurement procedures for SF endpoint metrics
Orchestration Configuration	X	X	X	X	Event logging for media service lifecycle configuration

Table 31: High level test areas for Platform integration

4.4 INTEGRATION INFRASTRUCTURE

The purpose of the integration infrastructure is to support functional testing and limited load testing of the Platform Product and Media Service Products. The infrastructure is designed to test product features within an infrastructure that replicates key aspects of production. The infrastructure only replicates part of the production infrastructure due to cost constraints. However, by using a software-based infrastructure it can be flexibly configured to support different test cases representative of those expected in real-life production trials.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Figure 25 shows the target logical topology of the integration infrastructure data plane. The design is based on supporting a core network of three switches with each switch offering two NAPs. Each NAP represents a connection to a data centre (DC) or edge node. DC NAPs offer connectivity to media services, edge NAPs allow IP Endpoints representing one or more end user devices to access the network. Some edge NAPs have limited compute capacity to host media services.

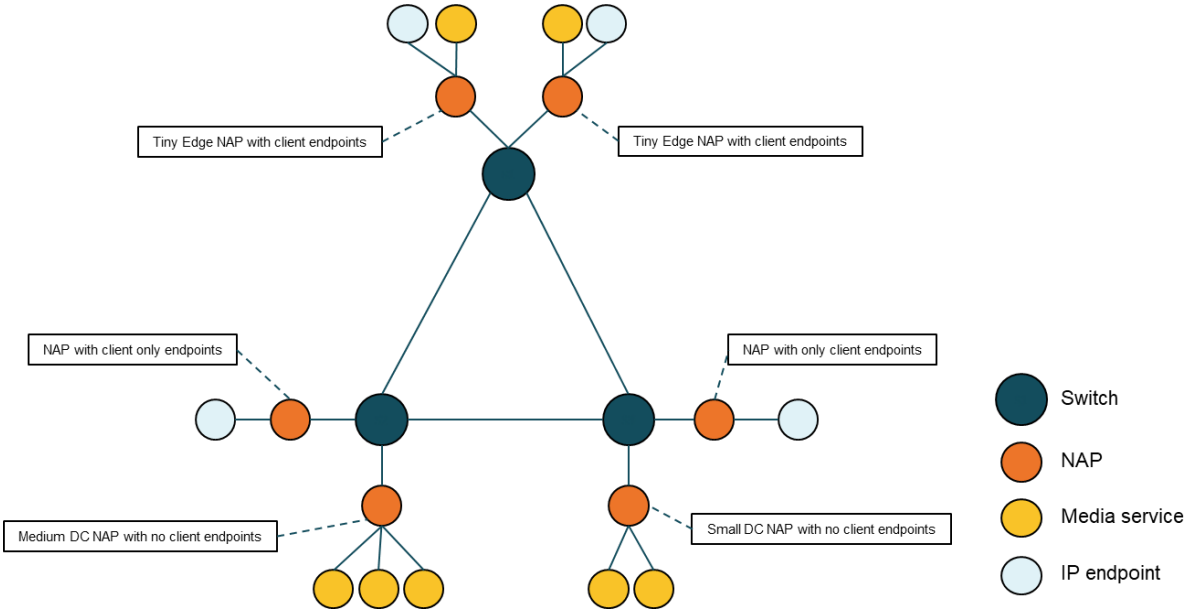

Figure 25: Logical topology of integration infrastructure data plane

This configuration offers a practical baseline for testing scenarios. The use of three switches allows different SF routes to be explored including cases of routing loops. The heterogeneity in DC and Edge resources allows SF endpoint management policies to be explored under different resourcing constraints. The distribution of IP endpoints allows for demand to be generated from different parts of the network.

The configuration is not fixed and different setups may be established for specific test cases or when resources need to be shared. For example, the media service resources may be aggregated to create a larger data centre or it in some situations may be more optimal to allocate resources to specific integration tests rather than offer the entire set up for each test.

The integration infrastructures management and control plane is shown Figure 26 covering infrastructure and platform services. These services are based on the Infrastructure Products and Platform products described in Section 2.3 and Section 2.4 respectively. At the lowest level OpenStack and OpenDaylight are deployed to provide management of virtual compute and the SDN fabric. OpenStack is used to allocate virtual slices of the integration infrastructure to the Platform for give integration scenarios. This is achieved through OpenStack's project concept allowing different tenancies to be managed. The tenancy is used primarily to manage different virtual slicing configurations required by integration tests rather than to support multi-tenant testing. However, there may be situations where multiple platforms need to be deployed across the integration infrastructure if concurrent testing is required by different partners. Using OpenStack's project and concept allows such configurations to be supported if needed.

The Platform services are then deployed as a stack of VMs within an OpenStack tenancy. This includes all of the SFs required for FLIPS, orchestrator and CLMC components as defined in Section 2.4. Figure 26 shows the platform SFs deployed on two servers. Even though the integration infrastructure is small

Co-funded by the Horizon 2020 Framework Programme of the European Union

scale and won't be implementing load balancing and clustering it's important to provide a reasonable mirroring of separation between components. To achieve this the infrastructure controllers and platform controllers are separated whilst the CLMC is also isolated considering the requirements for low contention on storage I/O.

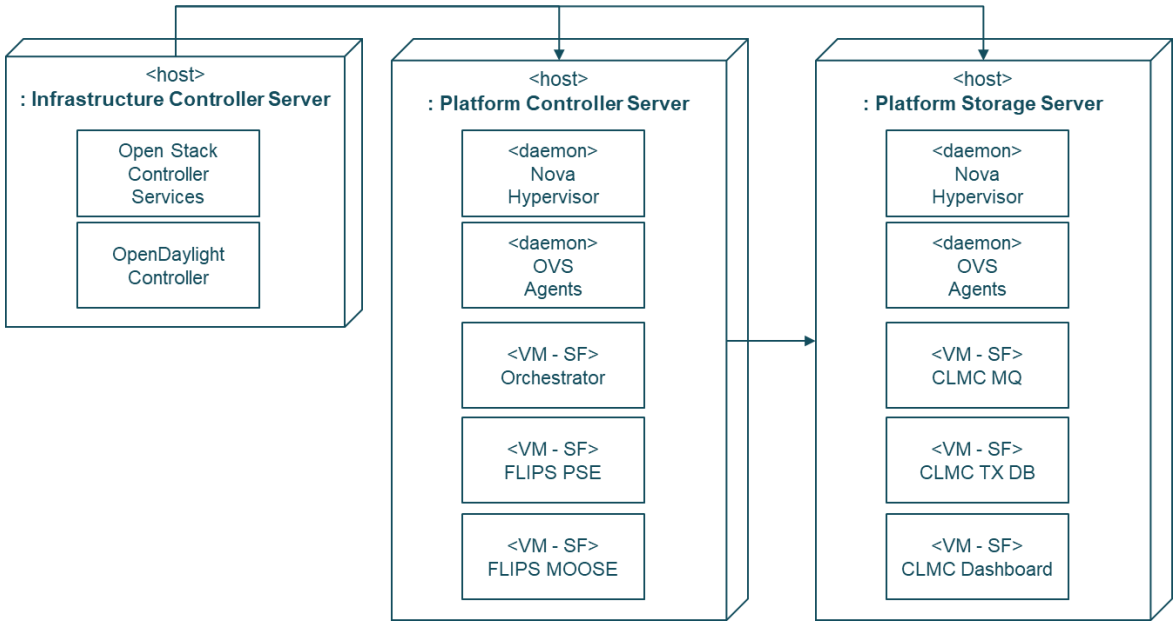

Figure 26: Integration infrastructure management and control plan

Table 32 shows an estimated capacity plan for the integration infrastructure considering the logical topology (Figure 25), the services required for the management and control plane (Figure 26) and continuous integration services including load test drivers. The full workload assumptions underpinning these estimates are described in Appendix A. Overall the total capacity required for integration is:

- number of VMs = 27
- number of CPUs = 49
- RAM = 103 GBytes
- storage = 5 TBytes

These estimates consider a standard media service size that can be used as the basis for functional testing of the Platform. The capacity plan does not consider how to support the requirements for all media services as these requirements have a high degree of variation and often require significant capacity beyond what can be provided in integration. Media services with service function chains requiring significant resources can only be tested on staging infrastructures where such capacity exists.

VMTypes	#elements (VMs)	#CPUs per VM	Total CPUs	RAM	Storage	Storage Type	Server	Workload assumptions
Integration Services								

Co-funded by the Horizon 2020 Framework Programme of the European Union

VMTypes	#elements (VMs)	#CPUs per VM	Total CPUs	RAM	Storage	Storage Type	Server	Workload assumptions
Continuous Integration	1	2	2	2	1000			Intermittent load based on build frequency, storage of build artefacts and test data
Infrastructure Product								
OpenStack Controller	1	2	2	8	1000			Baseline estimate
OpenDaylight Controller	1	1	1	8	100			Baseline estimate
OVS switch	3	1	3	3	20			Baseline estimate
Platform Product								
FLIPS PCE	1	1	1	1	20			Baseline estimate
FLIPS NAP	4	1	4	4	80			Baseline estimate
FLIPS MOOSE	1	1	1	1	100			Baseline estimate
Orchestrator OSM	1	8	8	16	100	Disk		
CLMC MQ (KAFKA)	1	4	4	32	500	Disk		Buffer Mem = write_throughput * buffer_seconds Storage = write_throughput * log_retention_hours Separate drive to avoid disk IO contention with other services
CLMC TX DB (INFLUX)	1	4	4	4	1000	SSD, IOPS 500		Assuming a single node Low = 5K writes a second, 5 queries a sec, 100K unique series Med = 250K, 25K, 1M Low Compute CPU: 2-4 cores RAM: 2-4 GB IOPS: 500 Storage Size Non-string values require approximately three bytes. String values require variable space as determined by string compression.
CLMC Dashboard (Grafana)	1	1	1	2	1	Disk		
Media Service Products								
Media Service Functions	6	2	12	12	600	Disk		Depends on scenarios
IP Endpoints	4	1	4	8	200	Disk		Depends on the test scenarios, could be co-located with the CI server will most likely be idle when the tests are running unless we run parallel build and integration tests
Totals	**27**	**29**	**49**	**103**	**4821**			

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 32: Integration infrastructure capacity planning

The final allocation of VMs to servers will be a trade-off between performance, isolation and cost. Consolidating VMs on fewer servers will reduce the cost of the integration infrastructure. However, this will result in poorer performance and more contention between the components during tests increase difficulty and time to resolve defects on test failure.

Co-funded by the Horizon 2020
Framework Programme of the European Union

5 CONCLUSIONS

This report has described the first technical roadmap for the FLAME project. The report describes a set of interdependent software products that together will provide a ground-breaking media service delivery platform exploiting the benefits of highly-distributed software-defined infrastructures.

The Platform product has been elaborated in detail being to core output of the project. A feature analysis is described for the FLAME platform covering Orchestrator, SF Endpoint Management and Control, SF Routing and Cross-Layer Management and Control. A feature critical path is provided for the alpha release and subsequent features distributed across future releases. The Infrastructure Products are described to provide the target deployment environment for products. The media service product roadmap is also included, identifying the foundation services that will form part of the FLAME offering. The relationship with experimentation and project KPIs is elaborated to explicitly show how features of the Platform address the key objectives of experimentation independent of physical location and reducing the time to perform experiments.

A systems integration and testing plan is defined detailing the DevOps processes including multi-project structure, development workflows, and testing tools. A software based integration infrastructure is specified that offers the ability to conduct integration tests that cover the expected features of the platform, are representative of the production infrastructures and allows for concurrent integration tests if needed for the different integration activities expected within the project.

Co-funded by the Horizon 2020
Framework Programme of the European Union

REFERENCES

[ETSINFV] ETSI GS NFV-SOL 004 Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification, available at http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/02.03.01_60/gs_nfv-sol004v020301p.pdf

[FLAME-D3.1] D3.1 FMI Vision Use Cases and Scenarios v1.0, available at https://www.ict-flame.eu/download/d3-1-fmi-vision-use-cases-and-scenarios-v1-0/

[FLAME-D3.2] D3.2 Experimental Methodology for Urban-Scale Media Trials v1.0, available at https://www.ict-flame.eu/download/d3-2-experimental-methodology-for-urban-scale-media-trials-v1-0/

[FLAME-D3.3] D3.3 FLAME Platform Architecture and Infrastructure Specification v1.0, available at https://www.ict-flame.eu/download/d3-3-flame-platform-architecture-infrastructure-specification-v1-0/

[MANO] Open Source Mano, available at http://www.etsi.org/technologies-clusters/technologies/nfv/open-source-mano

Co-funded by the Horizon 2020 Framework Programme of the European Union