Deliverable D6.6

Single-Domain Slice FCAPS management

Editor:	Luca Baldini, TEI
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31/05/2019
Actual delivery date:	07/06/19
Suggested readers:	Network Administrators, Vertical Industries, Telecommunication Vendors, Telecommunication Operators, Service Providers
Version:	1.0
Total number of pages:	70
Keywords:	Network Slicing, Management Plane, Network Slice, 5G, SDN, FCAPS

Abstract

This document reports the principles, design and prototype implementation of the fault detection, configuration, performance management, accounting and security (FCAPS) related subsystem within the SliceNet single-domain management framework. The subsystem includes modules that collect monitoring information relating to each slice according to the defined Service Level Agreements (SLAs) and apply policies upon detection of certain fault or

performance related events so that either corrective actions can be applied on time or inter-domain management modules are properly informed. Accounting of the usage of available domain resources is also covered. The single domain management subsystem provides the means for slice specific customization of the FCAPS administration through an abstraction of the underlying technologies, via the usage of appropriate descriptors, that allow the resource offerings of Network Service Providers (NSPs) to be addressed by Digital Service Provider (DSP) processes in a common way.

This document also describes the security and privacy mechanisms for the establishment of end-to-end encrypted channels between the different architectural elements of the SliceNet architecture, including Virtual Network Functions (VNFs), security for control and data planes, security and privacy for the management plane, etc. This approach to slice security aims to mitigate risks associated with lack of authentication, encryption and security by design, protection against attacks and other threats. Security is also possible to be accommodated in the context of the FCAPS descriptor-based management for threat detection and the related control loop automation achieved via policy definitions.

Disclaimer

The research leading to these results has received funding from the European Union Horizon 2020 Programme under grant agreement number 761913.

Impressum

[Full project title] End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualized Multi-Domain, Multi-Tenant 5G Networks

[Short project title] SliceNet

[Number and title of work-package] WP6- 5G Multi-Domain Slice Management Plane

[Number and title of task] T6.6 – Single-Domain FCAPS Slice Management

[Document title] Single-Domain, Multi-Tenant Network Slice Management (Fault, Configuration, Accounting, Performance / SLA and Security)

[Editor: Name, company] Luca Baldini, Ericsson Telecomunicazioni SpA (TEI)
[Work-package leader:] Thuy Truong, Dell EMC (EMC)

Copyright notice

Executive summary

The single domain FCAPS management framework in SliceNet has been designed to allow for slice operation automation as this is required in a modular environment in which a layered service model is required to deal with the heterogeneity of requirements as these are posed by Vertical actors and are clarified through the interactions between digital service providers (DSPs) and network service providers (NSPs). This layered model assumes that provider domains are abstracting and exposing as domain offerings management capabilities along with domain functionalities that can be exploited in automation loops.

The intra-domain FCAPS management focuses, therefore, on the support of automation of the operation tasks by limiting human interaction in the process of maintenance of the service level agreement (SLA) by embodying engineering knowledge about information collection, processing and conditional decision making in customizable artifacts. Engineering knowledge is embedded into sensing and actuation descriptors and workflows that contain two parts. One relates to the technology specific details while the other part abstracts these details so that they can be exported to DSP. Each descriptor is linked with network functions (NFs) that are available to be orchestrated according to the Orchestration sub-plane process and are activated upon activation of the related NFs. Similarly, upon availability of sensing information, policies can be defined to drive reaction practices to known or predicted situations. The descriptor and policy model is expected to be subject to the processing that will be applied in the context of the One Stop API workflows.

Several design principles of the FCAPS framework have been influenced by the Phase I 5G-PPP project SELFNET practices. Concepts have been revisited and re-engineered to cater for additional requirements that emerged in the context of multi-domain slicing.

In particular, this deliverable reports the following highlights in the design and prototyping of the single-domain FCAPS management framework and components:

- Definition of the automation principles that can allow the FCAPS management to be activated per slice in parallel to the orchestration workflows
- Definition, modelling and implementation of abstraction techniques that allow separation between business and technology domains and also enable easier interactions among different business roles
- A complete control loop framework that can enforce regular and well known FCAPS practices but also support interaction with cognitive processing components
- A granular and configurable framework for collection of information relating to fault, performance and security that can be used to trigger decisions on applying certain configuration actions or even support accounting processes.

List of authors

Company	Author
TEI	Luca Baldini, Ciriaco Angelo
CSE	Konstantinos Koutsopoulos, Athanasios Kokkinis
Dell EMC	Thuy Truong
IBM	Kenneth Nagin
NXW	Giacomo Bernini

Deliverable Reviewers

Company	Author
UPC	Fernando Agraz
UWS	Jose Alcaraz-Calero, Qi Wang

Table of Contents

List of figures

List of tables

Abbreviations

API	Application Programming Interface
BBU	Base Band Unit
CN	Core Network
CP	Control Plane
CPS	Control Plane Service
CQI	Channel Quality Indicator
CQI	Channel Quality Indicator
CSP	Communication Service Providers
DSC	Digital Service Customer
DSP	Digital Service Provider
E2E	End-to-End
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
ID	Identifier
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IMSI	International Mobile Subscriber Identity
KPI	Key Performance Indicator
LCM	Lifecycle Management
MANO	Management and Orchestration
ME	Mobile Edge
MEC	Mobile/Multi-access Edge Computing
MEO	Mobile Edge Orchestrator
NE	Network Element
NEF	Network Exposure Function
NF	Network Function
NFV	Network Function Virtualisation
NFVI	NFV Infrastructure
NS	Network Service; Network Slice
NSaaS	Network Slice as a Service
NSD	Network Service Descriptor
NSEP	Network Slice Provider
NSI	Network Slice Instance
NSP	Network Service Provider
NST	Network Slice Template

OAM	Operations, administration and maintenance
OSS	Operations Support System
P&P	Plug & Play
PNF	Physical network Function
PoP	Point-of-Presence
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational State Transfer
RO	Resource Orchestrator
RRH	Remote Radio Head
SBA	Service Based Architecture
SBI	Service Based Interface
SDN	Software Defined Networks
SDO	Standards Developing Organization
SLA	Service Level Agreement
SliceNet	End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualized Multi-Domain, Multi-Tenant 5G Networks
SW	Software
SS-O	Slice Service Orchestrator
UC	Use Case
UE	User Equipment
URI	Uniform Resource Identifiers
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	VNF Descriptors
VNFFG	VNF Forwarding Graphs
WG	Working Group

1 Introduction

The current document summarizes the design principles and functional details of the SliceNet single-domain FCAPS framework and its components. Aiming at addressing more accurately and more consistently the operation management of Slices, it was considered as more appropriate that the initially defined Tasks 6.1 and 6.2 should be merged so that a uniform approach can be followed. Additionally, single-domain security aspects defined in Task 6.5 were also included in this new joint task to complete the entire FCAPS aspects.

Any findings and updates relating to design and functionalities of the FCAPS framework, which will emerge during the integration phase, will be documented in integration reports along with validation and performance evaluation activities.

It is worth mentioning that the Network Function (NF) Configuration aspects have been addressed in the context of the Control Plane Services in WP4 according to SliceNet Control Plane services as identified in Deliverable 2.3 [2].

1.1 Scope of Single Domain FCAPS Management Framework

The overall SliceNet architecture foresees different layout of components for DSP and NSP domain. Figure 1 depicts the layout for DSP domain as this is presented in the SliceNet revised architecture. From FCAPS perspective, the components foreseen for the DSP are operating on top of the information provided by underlying NSPs with respect to QoE evaluation for delivered slices. Intra-domain FCAPS addresses particularly this requirement for the transactions annotated accordingly in the figure.

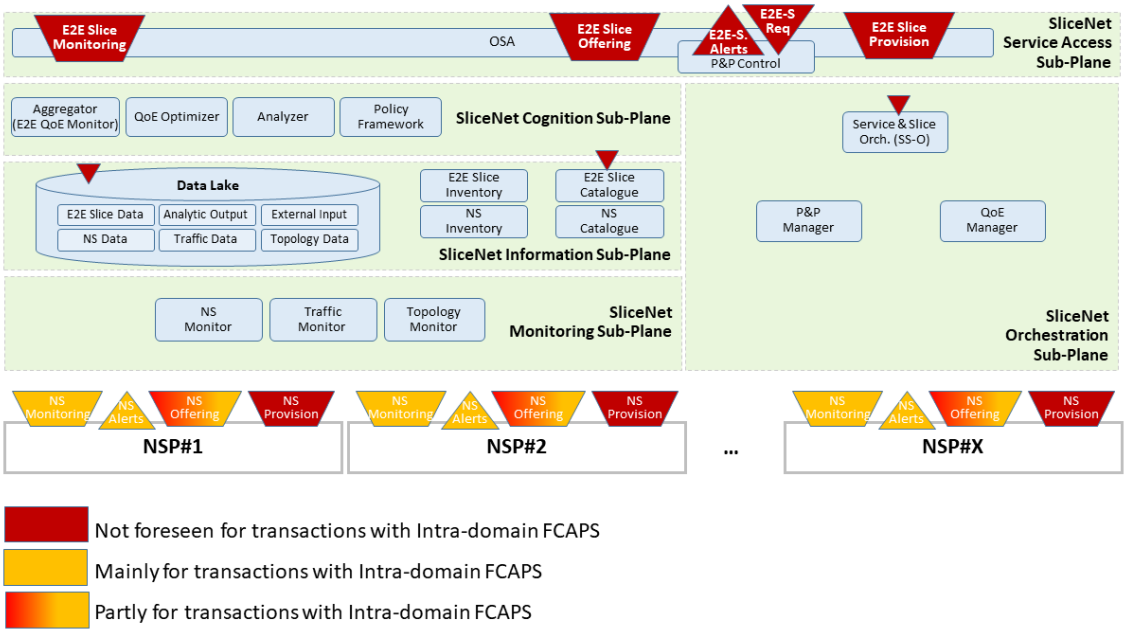

Figure 1: DSP level SliceNet system architecture

Figure 2 is the high level representation on the NSP SliceNet architecture. The FCAPS components are outlined with dashed blue colour and other related components (common for other functionalities) are outlined with dashed green colour, whereas there is again annotation about the transactions relating to intra-domain FCAPS. The Information Sub-plane is also

outlined to indicate involvement of the FCAPS framework which, however, is not exclusive. The proposed architecture layouts, with the differences between DPS and NSP, try to follow the separation of roles as foreseen by 3GPP. In addition to the separation of roles, multi-domain slicing aspects are also considered. Among these aspects, and in relation to FCAPS management, it is worth stressing the fact that NSP architecture needs to address technology specific aspects and to abstract them as federated offering to be utilised in the context of business logic requirements. This aspect of the business logic processing is reflected on the DSP architecture layout where QoE is at the centre of processing in contrast to the NSP architecture layout where lower level resource and performance metrics, alarms and counters must be considered. Moreover, the fact that a DSP may have to interact with several NSPs for provisioning of an end to end slice for a vertical is an additional factor that leads to the indicated differences.

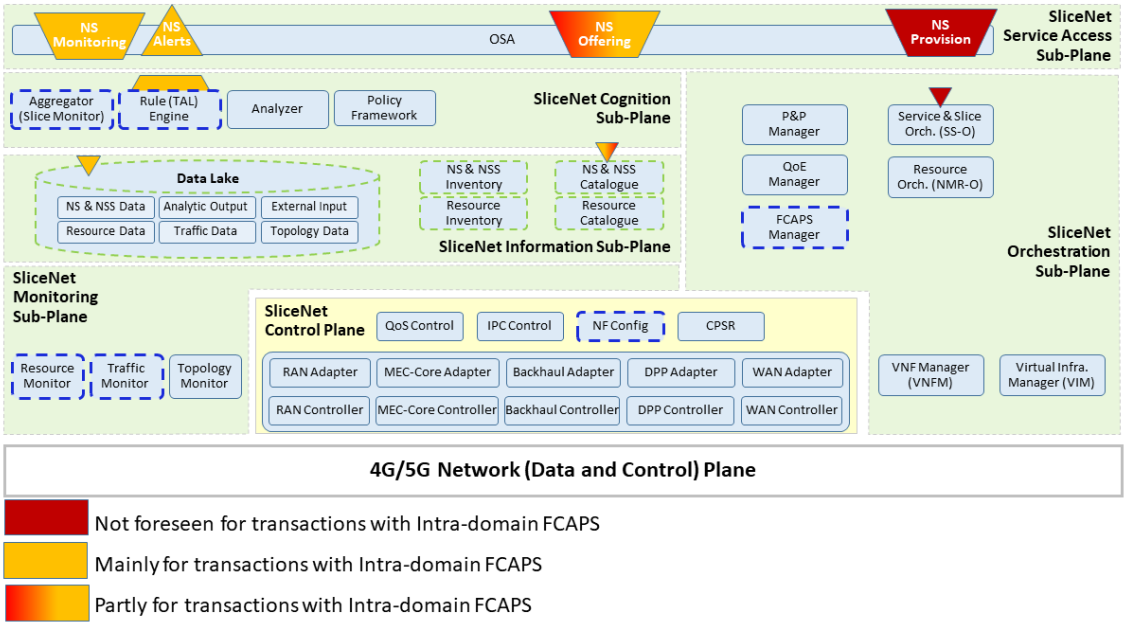

Figure 2: NSP level SliceNet system architecture

1.2 Document structure

This document is structured as follows:

- Section 2, summarizes relevant standardization topics relating to the design choices adopted for SliceNet FCAPS Framework.
- Section 3, provides the description of the design principles and architecture of the framework
- Section 4, explains the operation of the framework
- Section 5, provides the insight of the framework components and summarizes some prototype results
- Section 6, lists the conclusions stemming from the work carried out

2 Related Standardization

This chapter provides a study with respect to 3GPP and ETSI Technical Specifications about aspects relating to FCAPS management in the context of multi-domain slicing. The standardisation is not reviewed to identify those parts that are potentially relevant with slicing as this is approached and attempted to be delivered by SliceNet project. It is reminded that SliceNet approaches slicing by federating domain resources through abstracting and bundling these in a form of offerings that can be further aggregated at a higher level in order to bridge the business view with the technical realisation. Therefore, FCAPS management in SliceNet is defined in terms of the automation of the management and control loops foreseen in an administrative domain so that Verticals can benefit from the fact that business requirements can be easily and dynamically delivered in the context of a properly prepared communication overlay. With the introduction of service-based architectures (identified already in the specifications in the case of 5G and feasible to be adopted also in 4G systems due to the availability of virtualisation techniques) that can handle on-demand deployment and orchestration of service components from a variety of NFs onboarded and available per domain, a provisioned network or communication overlay template can be annotated with monitoring and actuation capabilities that can be utilised in the context of the support of the agreed level of service quality. In this context, related 3GPP and ETSI specifications were analysed and an overview of the related topics are provided as relevant with the SliceNet principles and design of the FCAPS components.

2.1 Network Slice Management Overview

3GPP TR 28.801 [22]identifies 3 management functions related to network slicing management:

- Communication Service Management Function (CSMF): this function is responsible for translating the communication service related requirement to network slice related requirements. The CSMF communicates with the Network Slice Management Function (NSMF).
- Network Slice Management Function (NSMF): this function is responsible for the management (including lifecycle) of NSIs. It derives network slice subnet related requirements from the network slice related requirements. NSMF communicates with the NSSMF and the CSMF.
- Network Slice Subnet Management Function (NSSMF). This function is responsible for the management (including lifecycle) of NSSIs. The NSSMF communicates with the NSMF.

As shown in Figure 3, the Os-Ma reference point can be used for the interaction between 3GPP slicing related management functions and NFV-MANO. To properly interface with NFV-MANO, the NSMF and/or NSSMF need to determine the type of NS or set of NSs, VNF and PNF that can support the resource requirements for a NSI or NSSI, and whether new instances of these NSs, VNFs and the connectivity to the PNFs need to be created or existing instances can be re-used.

NOTE 1: In order to use the NS, the NSMF and/or NSSMF would have to maintain an association between Network Slice Templates (NSTs), and NFV Network Service Descriptors (NSDs) with applicable deployment flavour identifiers, as well as an association between NSS instance identifiers (NSSI) and NS instance identifiers (NSI).

NOTE 2: The 3GPP slice-related management functions are still under definition in 3GPP SA5 and future updates might require further analysis about the interaction between 3GPP slicing related management functions and NFV-MANO.

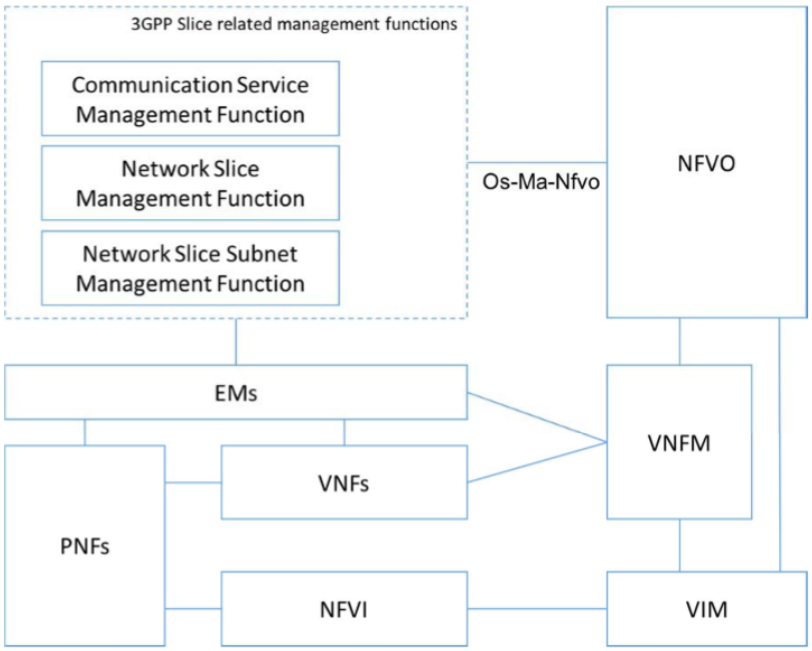

Figure 3: Network slice management in an NFV framework

From a resource management viewpoint, NSI can be mapped to an instance of a simple or composite NS or to a concatenation of such NS instances. From a resource management viewpoint, different NSIs can use instances of the same type of NS (i.e. they are instantiated from the same NSD) with the same or different deployment flavours. Alternatively, different NSIs can use instances of different types of NSs. The first approach can be used if the NSIs share the same types of network functions (or a large common subset) but differ in terms of the performance expected from these network functions (and from the virtual links connecting them) and/or the number of instances to be deployed for each of them. If slices differ significantly, mapping to different NSs, each with its own NSD can be considered. The same mapping principles might apply to NSSIs.

From the above descriptions it is evident that slice management is expected to be a highly dynamic process in the following years requiring equally flexible OSS/BSS processes to cater for the differentiation of end user services. This granularity is a key element in future workflows for the provisioning of slice and sub-slice components within NSP domains challenging the FCAPS systems with new requirements relating to the automated engagement of artifacts and processes. However, considering the fact that resources have to be optimally allocated

avoiding unnecessary waste computation, storage and networking resources, it is important that the automation is able to process SLA aspects and manage properly the required FCAPS artifacts.

2.2 NFV-MANO Interfaces and Specification

As NFV-MANO specifications are continuously adopted by modern telecommunication systems, it is worth considering the related specifications regarding management endpoints and interfaces (Figure 4) as these can be exploited in the context of FCAPS fault and performance sensing workflows.

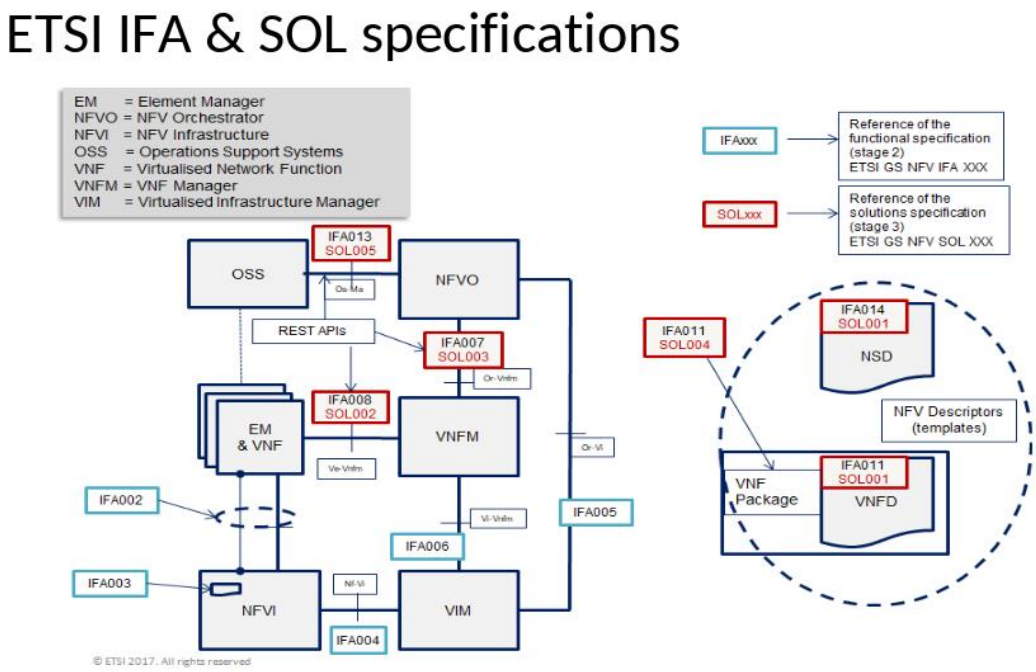

Figure 4: ETSI NFV-MANO Specifications

One example of interface relevant for this document is the NS Fault Management interface is part of the Os-Ma interface described by IFA 013 and SOL005. This interface shall allow the NFVO to provide alarms related to the NSs visible to the consumer. An alarm on a given NS results from either a collected virtualized resource fault impacting the connectivity of the NS instance or a VNF alarm, resulting from a virtualized resource alarm, issued by the VNFM for a VNF that is part of this NS instance. The fault management interface shall support the following operations:

- Subscribe operation: Subscription of OSS/BSSs with the NFVO for the notifications related to the alarms.
- Notify operation: Notifications of alarms or alarm state change from NFVO to OSS/BSS.
- Get alarm list operation: Accessing active alarms from the NFVO.
- Terminate Subscription operation: Terminating a particular subscription in the NFVO.
- Query Subscription Info operation: Querying subscription information from the NFVO.

- Acknowledge Alarms operation: Acknowledging alarms by the OSS/BSS.

2.3 General requirements

3GPP [20]identifies among the general principles for network management the support of service-based management whereas for network slicing management foresees "*a simple network slicing set of management functions to simplify the management of network function(s) from the slicing management point-of-view*" as well as that "*management services are capable to support various Network Operator deployment options to support diverse use cases, and a set of generic management services applicable to all kinds of network functions*". Additionally, supervision and performance reporting (e.g. for KPI monitoring) is expected during the operation phase of slices. Along the same lines and with respect to network slice subnet concepts it is stated that *"the grouping of the network functions allows the management of each group of network functions to be conducted independently of the network slice instance"* while management aspects are NSSI specific as dictated by the related templates that can be used to create *"different network slice subnet instances with different instance-specific information".* Finally, the role of management data analytics is also highlighted as a means "*for improving networks performance and efficiency to accommodate and support the diversity of services and requirements. The management data analytics utilize the collection of network data (including e.g. service, slicing and/or network functions related data) to perform analytics in order to assist and complement management services for an optimum network performance and service assurance."*

All the above are addressed through a subset of business level requirements as extracted from the overall set of General Requirements:

REQ-5GNS-CON-01 The network slicing management architecture shall allow **any deployment options within the Network Operator's domain**.

REQ-5GNS-CON-02 The set of network slicing **management functions shall be generic** to all kinds of network function and network function provider.

REQ-5GNS-CON-06 The network slicing management architecture should provide management **capabilities** that are **dedicated to each network slice instance**. The instance management dedicated to a network slice instance shall work independently from the instance management dedicated to another network slice instance.

REQ-5GNS-CON-07 The network slicing management architecture shall allow managing **multiple network slice instances simultaneously or independently** along with their lifecycle.

REQ-5GNS-CON-13 The 3GPP management system shall be able to **provide management data analytics** to authorized consumers.

REQ-5GNS-CON-14 The 3GPP management system shall be able to **collect and analyse relevant management data.**

In the same context, the following Network Slicing Management Requirements are also extracted:

REQ-3GPPMS-CON-02 The 3GPP management system shall have the capability to **translate the communication service requirements** to network slice related requirements.

REQ-3GPPMS-CON-05 The 3GPP management system shall have the capability to **monitor the network slice related data and provide the agreed data** to an authorized consumer.

REQ-3GPPMS-CON-11 The 3GPP management system shall be able to report performance measurement data of a network slice instance to the NOP.

REQ-3GPPMS-CON-12 The 3GPP management system shall be able to **report performance measurement data** of a network slice subnet instance to the NOP.

REQ-3GPPMS-CON-13 The 3GPP management system shall be able to **report fault management data** of a network slice instance.

REQ-3GPPMS-CON-14 The 3GPP management system shall be able to **report fault management data** of a network slice subnet instance.

REQ-3GPPMS -CON-25 The 3GPP management system shall support **collection and analysis of the status and events** of the network slice instance resources for the purpose of **fault management**.

REQ-3GPPMS -CON-26 The 3GPP management system shall support **collection and analysis of the status and events** of the network slice instance resources for the purpose of **performance management**.

REQ-3GPPMS-CON-27 The 3GPP management system shall have the **capability of exposing network slice management data** for network slice as a service to the authorized consumer.

REQ-3GPPMS-CON-30 The 3GPP management system shall be able to **expose the network slice management services such as performance management, fault supervision and provisioning management** to the authorized consumer based on the mutual agreement between consumer and operator.

REQ-3GPPMS-CON-31 The 3GPP management system shall have the capability **to expose, based on the mutual agreement between consumer and operator, the network slice assurance services** to the authorized consumers.

REQ-3GPPMS-CON-32 The 3GPP management system shall have the capability **to expose, based on the mutual agreement between consumer and operator, the network slice control and configuration services** to the authorized consumers and to resolve potential conflicts

Vertical services requirements variations are identified by 3GPP [21]as a key aspect with respect to performance assurance. The highlighted concept foresees collection of performance data that can be potentially used along with other data (configuration, conditions) for issue mitigation by dedicated applications. Collection of data is expected to be dynamically carried by what 3GPP identifies as “measurement jobs” that can produce the required performance metrics for the dedicated applications to process in the context of issue mitigation.

3 Design Principles and Internal Structure

Although FCAPS management is an aspect that has been approached in certain ways in telecommunications systems and there have been concrete solutions and tool suites by several vendors and solution providers, SliceNet introduces an approach that is not aiming at replacing Network Service Providers (NSPs) methodologies and established infrastructures but is rather targeting the multidomain slicing support of 4G and 5G systems as this is expected to be tailored to Vertical needs by considering aspects such as layered integration, abstraction, automation and self-organising principles.

SliceNet assumes that multidomain provisioning of slices involves several players from more than one layer. It considers the NSP as the lower level of slice provisioning with its offering being available for reservation and utilisation under the DSP logic. DSP is considered as the mediator between vertical and network. The processing of the request applied by the DSP aims at resolving an adequate synthesis of NSP resources that can support the service level requirements posed by the vertical. In the following paragraphs the focus will be on the principles introduced by SliceNet with respect to FCAPS management that can allow an NSP to approach federation/offering of its resources so that these can be selected for slice provisioning.

3.1 Sensing and Actuation Management

In a given NSP domain there are specific technology components that are expected to be deployed or engaged in slice provisioning. Those components can be either virtualised or physical resources for which it is assumed that there is an NSP specific orchestration approach to be followed during slice or sub-slice commissioning. Among these resources there can be a variation of sensing or actuation capabilities. Those capabilities are subject to be processed in the context of the DSP slice request processing. For example, upon a slice template request which involves specific service resilience requirements, a DSP might be searching among the NSPs offerings for those resources that provide CPU utilisation and temperature monitoring information so that a DSP defined mitigation policy can be implemented/enforced. The DSP is not aware how these metrics are collected, e.g. as SNMP or Ceilometer counters, but it only states that these should be available for higher level analytics processing. Similarly, the DSP might be requiring that particular configuration options should be available as actuation/mitigation options. For example, the option of certain flow acceleration or blocking should be available. Again, the DSP only needs to know that this option is available by a particular NSP without knowing that this is implemented as specific NFs configuration or as an SDN related configuration. This concept, highlighted in the following figure (**Error! Reference s ource not found.**), allows targeting management functionalities in an abstract way by giving room for the development of several alternatives that serve the same purpose. Those alternatives can be onboarded in the form of monitoring and actuation descriptors that are focusing on separation of autonomic and cognitive management from measurement and metric collection as well as from the orchestration/enforcement processing of the mitigation reactions. Thus, autonomic or cognitive procedures are abstractly defined on the basis of available metrics and operations, whereas measurement collection and action enforcement are resolved and applied in the context of the protocols defined for the involved resources.

As it is shown in the figure below (Figure 5), it is expected that the NSP, being aware of the capabilities of the resources it is federating, maintains a catalogue repository of the offerings (NF types) by associating the monitoring and actuation options with the resources. Additionally, from the inventory repository allocated resources are possible to be associated with their types. The catalogued and inventory part of the information is expected to be hosted in the Information sub-plane. Inventory information can be also retrieved either directly by the Information sub-plane or through references from information kept at the Orchestration sub-plane. The top part of the figure provides the functional decomposition of the concept of descriptors whereas the bottom part provides the details how this concept is mapped on the architectural components of the NSP domain.

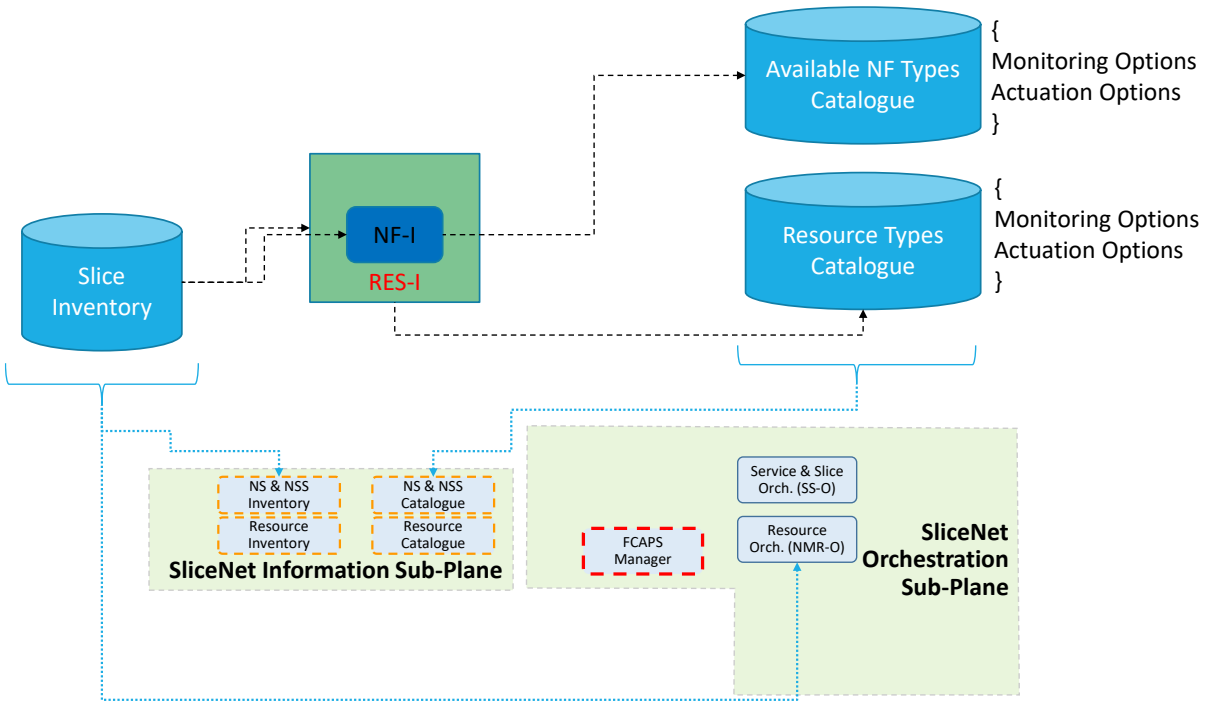

Figure 5: NFs and Resource Monitoring and Actuation Descriptors Concept

In the case of the monitoring options, it is expected that the following figure (Figure 6) structure is onboarded and maintained in catalogue repository. Monitoring regards all aspects of slice supervision tasks including faults, alarms, security alerts or even collection and storage of accounting information.

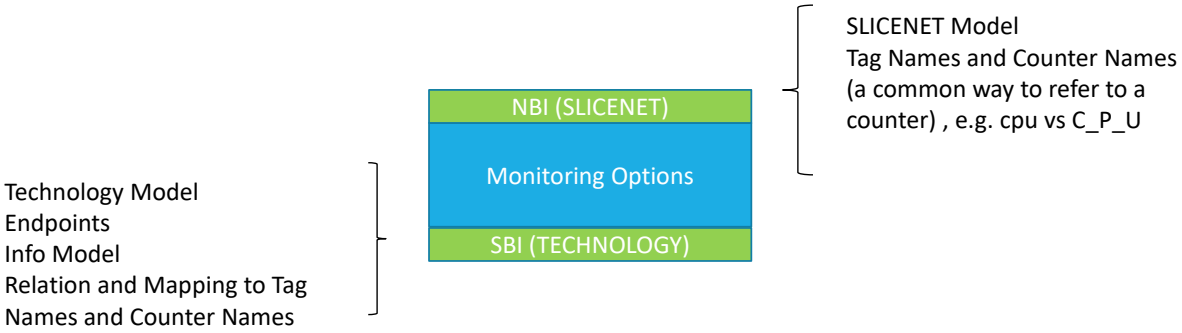

Figure 6: Monitoring Descriptor Concept

According to the Monitoring Descriptor concept, two parts are foreseen. One part relating to the information that can be exported to the DSP (NBI part) and one part relating the way the NBI abstract information is mapped to the technology specific details that characterise the NSP components that are associated with this monitoring options.

Actuation options are structured in similar way (Figure 7). The NBI part exports an operation name that can be used for issue mitigation and the Workflow part provides the details on how the operation can be provided via a workflow execution that relates to resolution of specific endpoints.

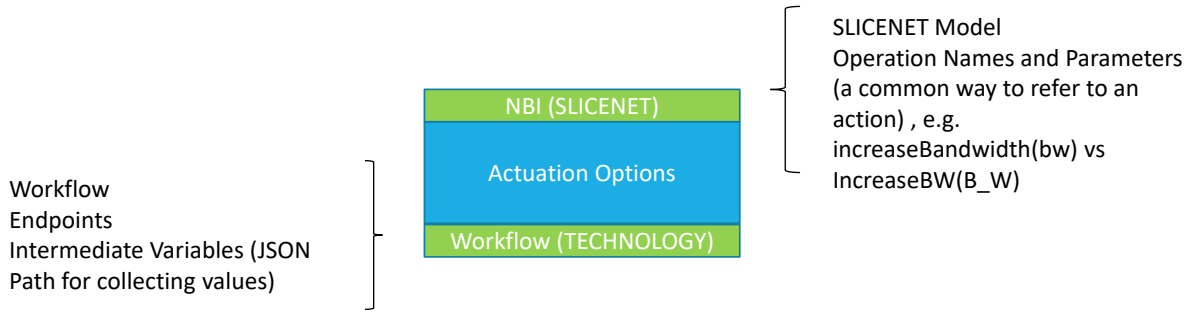

Figure 7: Actuation Descriptor Concept

3.2 Dynamic sub-slice driven Instantiation/Configuration

The availability of the descriptors and their association with resource types is used during instantiation of the sub/slices within the domain of the NSP. The slice requirements, as resolved in terms of which monitoring and actuation options have to be supported during the lifetime of a slice (NS Template and relation with NSS Templates), are reflected on inventory entries in terms of association of the resource instance identifiers with the monitoring or actuation descriptors identifiers (Figure 8). This association is captured by the FCAPS framework as a trigger for activation of actuation and monitoring artefacts. This process requires that a set of variables are available to be used for the configuration of the FCAPS artefacts. For example, an artefact activated to collect performance measurements counters provided by a particular NF type has to be configured with the details to apply for the attaching to the technology specific endpoint provided by the NF, e.g. REST or Kafka Bus details, filtering information, etc.

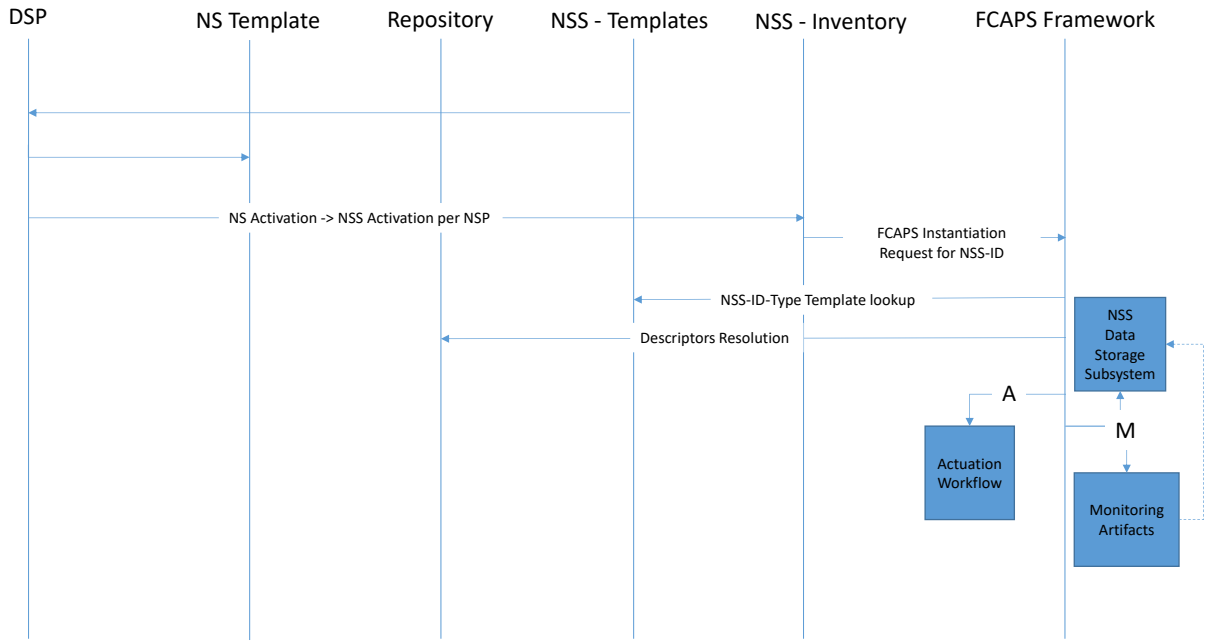

Figure 8: FCAPS Instantiation

With the previous steps (onboarding and activation) FCAPS automation can reach the point at which monitoring artefacts are instantiated and configured to start collecting information. From this point and on data sets with counters are being populated. The availability of the information is provided through a time series database on top of which aggregation tasks can be started based on statistical evaluation of the quantitative data.

3.3 Control Loop Automation

Upon data availability, analytics options should be applied to detect issues and evaluate the conditions for mitigation actions. This requires for a given sub-slice a set of rules should exist to identify how the collected information will be used. The rules should be compliant with the monitoring options resolved during slice provisioning based on vertical's requirements. Similarly, a set of rules will be defined for the handling of events and alarms related to the sub-slice. The enforced rules themselves are subject to selection of available rule templates that have been composed to reflect network administrator knowledge. The rules do not imply any cognitive processing, but they can be composed so as to allow feeding of cognitive modules with data to be processed for prediction of failure to be mitigated. The mitigation itself is part of the rule. Rules can be also defined not to close the control loop internally, within the NSP, but their trigger should lead to data forwarding, as potentially allowed by agreements between DSP and NSP, of data to DSP monitoring infrastructure.

3.4 Intra-domain FCAPS Framework Functional Layout

The above described principles have led to the definition of the following (Figure 9) architecture outline of the single domain (NSP) FCAPS management subsystem. In this architecture there two main components. The FCAPS manager that is responsible for activating all the monitoring artefacts that can be composed either by properly configured

Telegraf instances that populate the measurement time series in an InfluxDB, or by custom artefacts that provide a specific adaptation with respect to sensing sources. Telegraf, InfluxDB and Kapacitor are all modules of the TICK stack as explained in 5.1. Custom artefacts can be either coupled with a Telegraf instance for information persistence or directly connect and populate the InfluxDB. For both Telegraf and custom artefacts the FCAPS manager creates, configures and starts Docker containers. The other main component in the architecture is the Rule Engine. This engine is responsible for applying the aggregation rules through Kapacitor and also for maintaining per slice a set of rule trigger objects that apply the required processing according to the rule definition.

The figure provides at the top part the functional components of the framework with links to architecture components at the bottom.

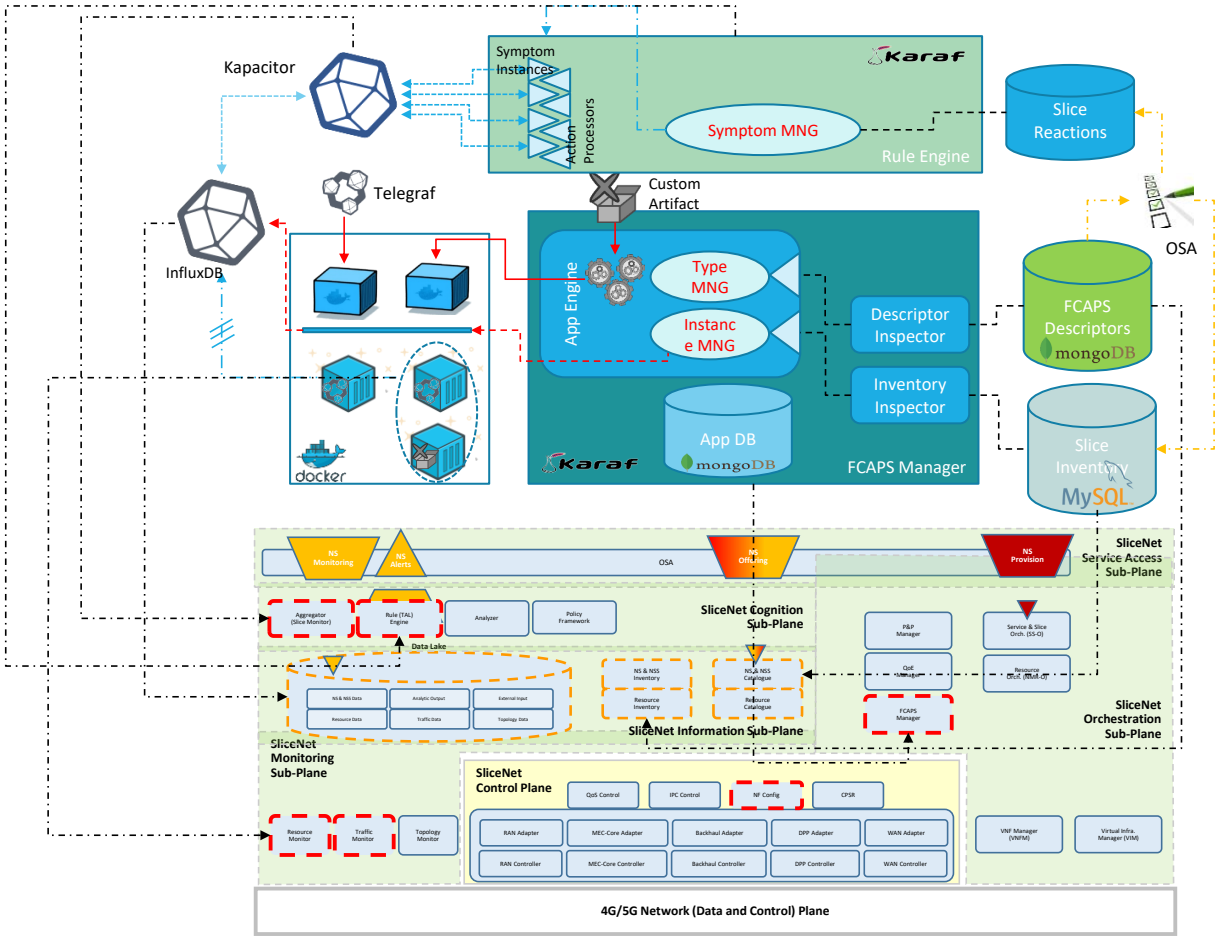

Figure 9: FCAPS Framework and Architecture Relations

4 FCAPS Workflows/Operation

The SliceNet intra-domain FCAPS framework has been designed to allow for automated deployment and configuration management of all the required artefacts that relate with the sub-slice requirements as far as intra-domain issue detection and mitigation is concerned. The intended automation is achieved by implementing a framework to be triggered through the analysis of inventory information that is provided as the result of the sub-slice commissioning process. In the next paragraphs, the various workflows relating to the operation of the FCAPS framework are presented.

4.1 Descriptor Context and NF Association

As stated earlier, every NSP, participating in multi-domain slicing, is expected to be federating its offerings on the basis of options that can be processed in the context of optimal slice composition as this is carried at the DSP level for delivering slices to Verticals that meet the indicated performance and quality levels. Although capacity and scale options are subject to orchestration decisions for concluding to a resource adequate slice synthesis, maintenance of the delivered quality within acceptable limits involves active FCAPS participation for collection and analytical processing of monitor-able information that aims at the detection of situations that can cause deviation from the levels of the agreed quality so that mitigation strategies can be applied.

The first step to achieve information provisioning upon which analysis can be applied is to identify the potential sources of that information. This identification relates with two aspects: i) what can be collected, and ii) how this can be collected. Both aspects relate with the infrastructure of the NSP and the virtualised and physical resources that can be orchestrated for the delivery of a sub-slice. For any of these resources the NSP is obviously aware of the tools and platforms that are involved and how the information can be extracted. SliceNet requires that this knowledge is expressed in terms of a parameterised set of details so that it can be instantiated as many times as the related type of resource is involved in a sub-slice provisioning, and there is interest in collecting this information for the particular sub-slice. This optional interest in collection of information relates with the (sub-)slice template that the Vertical requirements indicate. From the FCAPS perspective this is reflected on inventory entries that establish an association between an instance of a resource type and the corresponding FCAPS descriptor.

The above steps are highlighted in the following workflow (Figure 10).

1. Network Function Types are identified for being sub-slice components
2. FCAPS descriptors are composed and registered per NF type
3. During sub-slice provisioning, NF types are either instantiated or referenced by a sub-slice instance
4. Each NF instance has a variable set relating with its role in the sub-slice and with the running instance details
5. An association between descriptor identifier with NF identifier (intra-slice) is created to activate the FCAPS related monitoring that this NF can provide. Absence of the

association for an NF for which there is a descriptor is considered as not selected monitoring option.

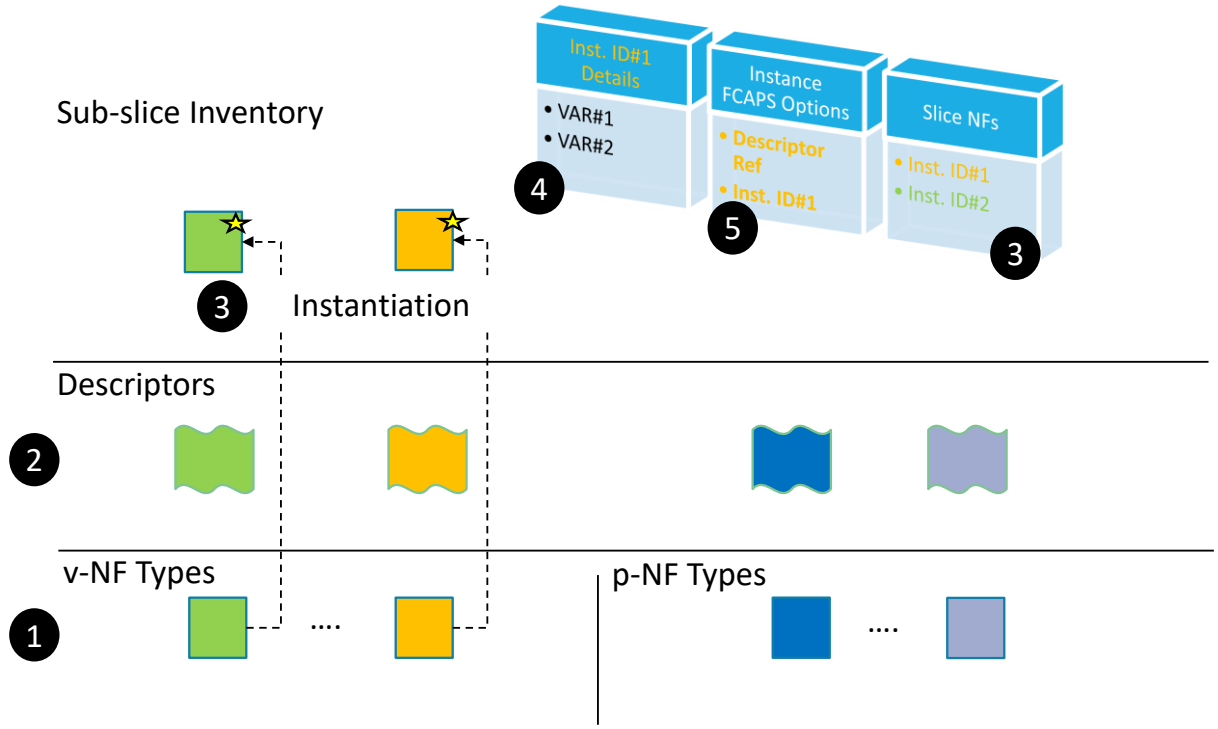

Figure 10: NF Association with FCAPS Descriptor

4.2 Monitoring Artifacts Activation

Once a monitoring descriptor (Table 1) is detected in relation with a physical or virtual function a monitoring artifact is activated.

Field	Type	Details
name	String	The name of the sensor to be used. The name is exposed along with the description and counter list as NSP offering. The name is also subject to sensor identification for aggregation purposes (4.3).
description	String	A text stating the capabilities of the sensor
counters	Array	A list of counter names (SliceNet common names) that the sensor makes available. This part along with the name forms the NBI part.
protocol	Object	The SBI part as explained in Table 4
artifact	Object	The custom artifact as explained in Table 2

Table 1: Descriptor Structure

The descriptor covers two cases, pure Telegraf based approach or custom component assisted approach. In case of a custom component the descriptor contains an “artifact” entry containing the following entries:

Field	Type	Details
location	String	The location of the binary artefact to fetch from
username	String	Any username to use for authentication
password	String	Any password to use for authentication
path	String	The path inside the container image where the binary artefact should be stored
launchCommand	String	The launch command to use for starting the binary
configuration-vars	Array	A set of configuration parameters as indicated in the next table (Table 3).

Table 2: Artifact Entry Structure

The configuration variables are structured as follows:

Field	Type	Details
file	String	The file path within the image into which this variable should be appended
name	String	The name of the variable to be appended
value	String	The value of the variable to use. The value can be static or enclose within curly brackets ({}) to identify the fact that the value that has to be used can be retrieved from an inventory lookup. The enclosed content identifies how the resolution of the value should occur. The most common setting can be a reference to one of the variables identified for the NF inventory content that was indicated in step 4 in the previous workflow.

Table 3: Artifact Configuration Section Structure

In case a Telegraf instance has to be activated, the descriptor provides a “protocol” section with the following content:

Field	Type	Details
type	String	The type of the Telegraf plugin to activate, e.g. REST, Kafka
address	String	The endpoint to retrieve the data from
path	String	Any extra info for interfacing with the endpoint. E.g. REST path or Kafka topic
tag	Array	The list of tag keys to persist in the database
merge_inkeys	Array	The list of the primary keys the resulting set can have
merge_outkeys	Array	The list of secondary keys that can be resolved from the set
metadata	Array	Any extra fields to be marked as dimensions. These are expected in string format in the data and inclusion in this list avoids them from being omitted when the time series is populated.
format	String	The format of the data with JSON as the most common option

fields	Array	The list of the key:value entries with keys identifying the fields that are allowed to be collected from the monitoring source. Values identify filtering content. Values can be indicated in curly bracket form to indicate inventory resolution or variable replacement.
mappings	Array	A set of key:value entries. The keys are fields provided by the monitoring source. The values are the final fields to be stored in the time series database.

Table 4: Protocol Section Structure

An example descriptor is provided next:

```
{
    "name": "ueCounter",
    "description": "Collects downlink and uplink traffic counters per UE",
    "counters": [
        "ip",
        "imsi",
        "slice_id",
        "downlinkBytes",
        "uplinkBytes"
    ],
    "protocol": {
        "type": "rest",
        "address": "http://{artifact}:2000/",
        "path": "{extraPath}",
        "tag": [
            "imsi"
        ],
        "merge_inkeys": [
            "imsi"
        ],
        "merge_outkeys": [
            "ip"
        ],
        "format": "json",
        "metadata": [
            "ue_ip",
            "imsi"
        ],
        "fields": {
            "ue_ip": "",
            "imsi": "{slicenet.slice.id.imsi}",
            "slice_id": "",
            "dl_byte_count": "",
            "ul_byte_count": ""
        },
        "mappings": {
            "ip": "ue_ip",
            "uplinkBytes": "ul_byte_count",
            "downlinkBytes": "dl_byte_count"
        }
    },
    "artifact": {
        "location": "http://127.0.0.1/FMM-1.0-SNAPSHOT-onejar.jar",
        "username": "sdf",
        "password": "sfd",
        "path": "/opt/myFaultDetector",
        "launchCommand": "java -jar /opt/myFaultDetector/FMM-1.0-SNAPSHOT-onejar.jar",
        "configuration-vars": [
            {
                "file": "/opt/myFaultDetector/my.conf",
                "name": "topic",
                "value": "{KafkaTopic}"
            },
            {
                "file": "/opt/myFaultDetector/my.conf",
                "name": "bootstrapServer",
```

```
                "value": "{bootstrapServer}"
            }
        ]
    }
}
```

4.3 Automation Rules/Policies

While the automated provisioning, by the FCAPS Manager, of all the required artefacts for adapting and exploiting monitoring sources and capabilities ensures the availability of the quantitative data, an additional framework for combining these data for issue detection and activation of foreseen reactions is complementing the overall approach. This framework is delivered by the Rule Engine that is able to be configured with a number of rule definitions per sub-slice. These rules are structured according to an event-condition-action workflow. The engine extracts from the various parts of the rules either configurations to be applied via the Kapacitor module or internal definitions of workflows to be applied for delivering the appropriate automation control loop. For any rule, two branch categories are foreseen to define a reaction: the diagnosis (sensing and analysis) branch and the tactic branch (Figure 11). The two branches are linked through a conditional logic that ensures the proper matching of the mitigation plan with the analysis of the situation.

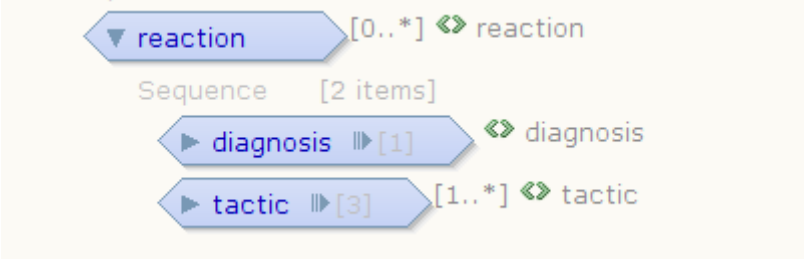

Figure 11: Reaction Definition

The content of the rules to be applied are formatted as XML according to a XML schema that is based on the definition of the Tactical Autonomic Language (TAL) developed in SELFNET [25]. The schema has been enriched with the condition support to better address the SliceNet requirements. Although TAL defines a layered model that reflects the interrelations between monitoring or actuation sub-planes, it does not indicate how the definitions should be processed and from which component. This logic is applied by the involved components that either consume the TAL based definitions or, based on them, create intermediate configuration artefacts to be forwarded to other components. In the following subparagraphs, a description of the logic that is intended to be applied is presented by highlighting the expected elements from the TAL model that are relevant to SliceNet. A complete example is appended in Annex A.1.

4.3.1 Sensing/Analysis Branch

The sensing or analysis branch foresees a hierarchical association of the information as this is formed through the various components starting from the sensing sources, passing through the aggregation processing for metric calculation, comparison with threshold levels and, finally, any analytical processing in the context of cognitive management of the information. Steps can

be omitted but in any case the rules should be relevant with the activated descriptors and particularly with the descriptors NBI definition.

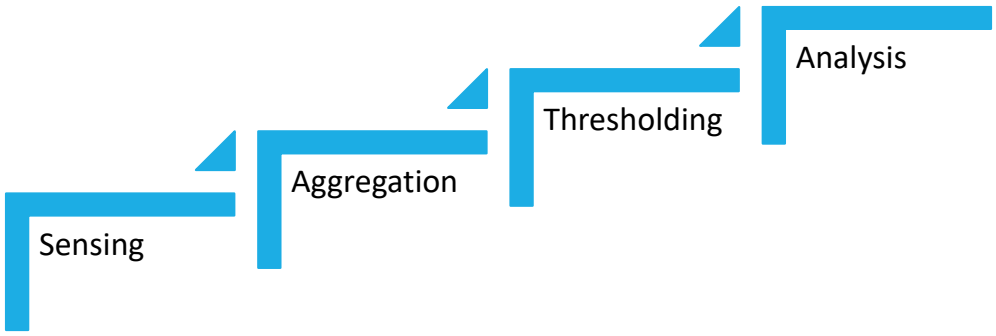

Figure 12: Automation Rule Sensing Branch

The Sensing part (Figure 13) of the branch identifies the source of the information that has to be further processed. The object identifier of the sensor (**OID** attribute) indicates the measurement's time series content (database table) from where data should be used. This is a set of field names (**parameter**) as they appear as column names (parameter **name**) in the related table. In SliceNet, the configuration part of the sensor is not mandatory as the definitions are not foreseen to be influencing orchestration.

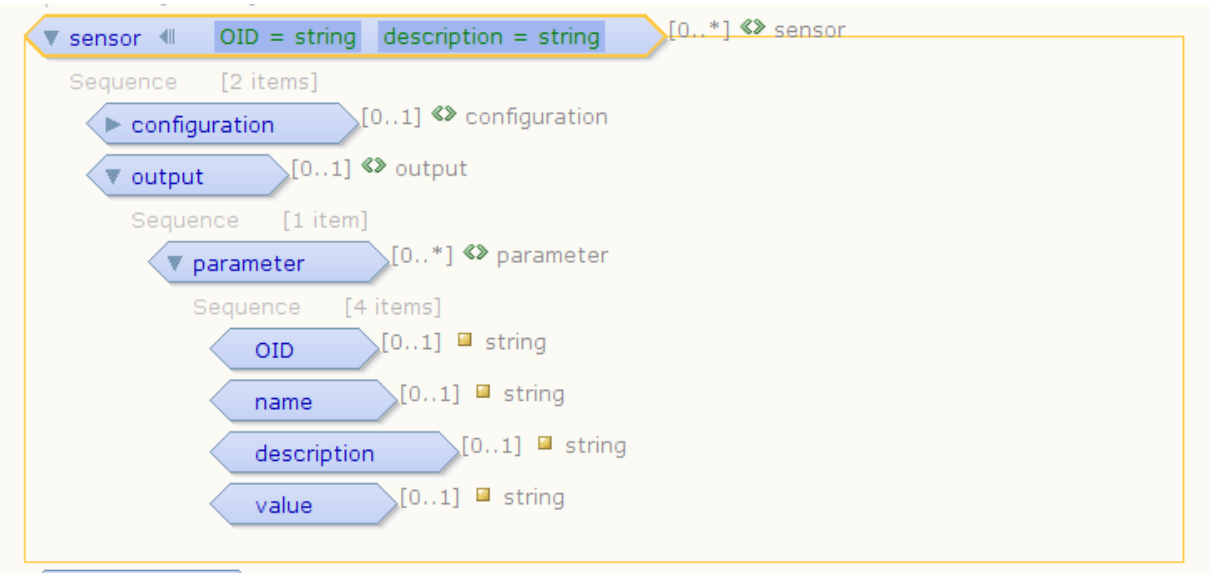

Figure 13: Sensor Definition

The sensing section may be wrapped within an aggregation section (Figure 14) which defines the calculation of a metric as an intermediate variable. The sensing input is identified as **aggregation data** over which an **aggregation rule** has to be applied.

Figure 14: Aggregation Definition

The **aggregation rule** (Figure 15) is defined as a statistical or mathematical processing of a sensing variable over a **period** of time. The processing is indicated as **rule item** which can be selected from a number of options. The **parameter** structure is a key aspect in TAL and therefore it is maintained at all levels (configuration, input, output). In the specific case, the parameter field is used to identify the input that has to be used in the aggregation processing. The name of the parameter identifies the name of measurement field to be used.

Figure 15: Aggregation Rule Definition

The result of the aggregation is indicated in the **metric** (Figure 16) element (a newly defined parameter identified by a specific name) along with any accompanying dimensions that have been used to group the data from the measurement series. The newly defined metric may be subject to be compared (**threshold level name**, **comparisonType**: less, equal, more, etc.) with a **threshold level** (**value**). Threshold comparisons can be linked with logical associations (and, or) and also the **occurrences** of the threshold condition may be defined (mainly consecutive).

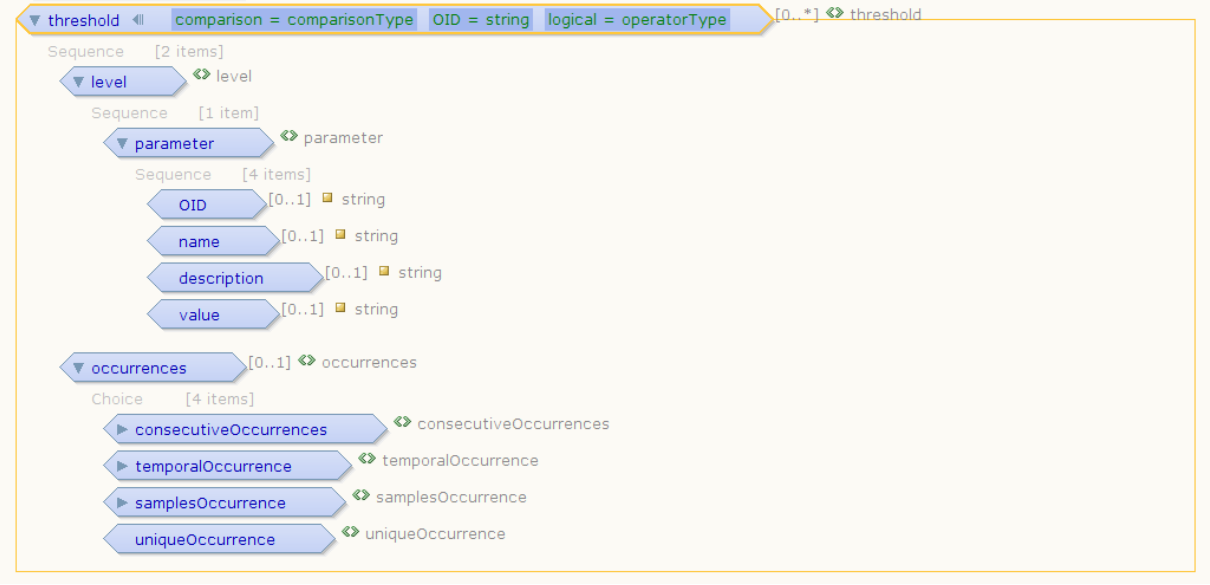

Figure 16: Threshold Definition

The outcome of this comparison is considered as a trigger for the rule engine in case no further processing is foreseen. However, there may be an optional **analysisRule** which can be used to indicate that the results of the previous steps should be forwarded to an external module that can be NSP's cognitive mechanisms. The external module is identified by an object identifier, mainly to resolve the way the information should reach the external module, and by a set of input and output parameters. The trigger in this case is associated with the detection of the output that has been highlighted in the definition.

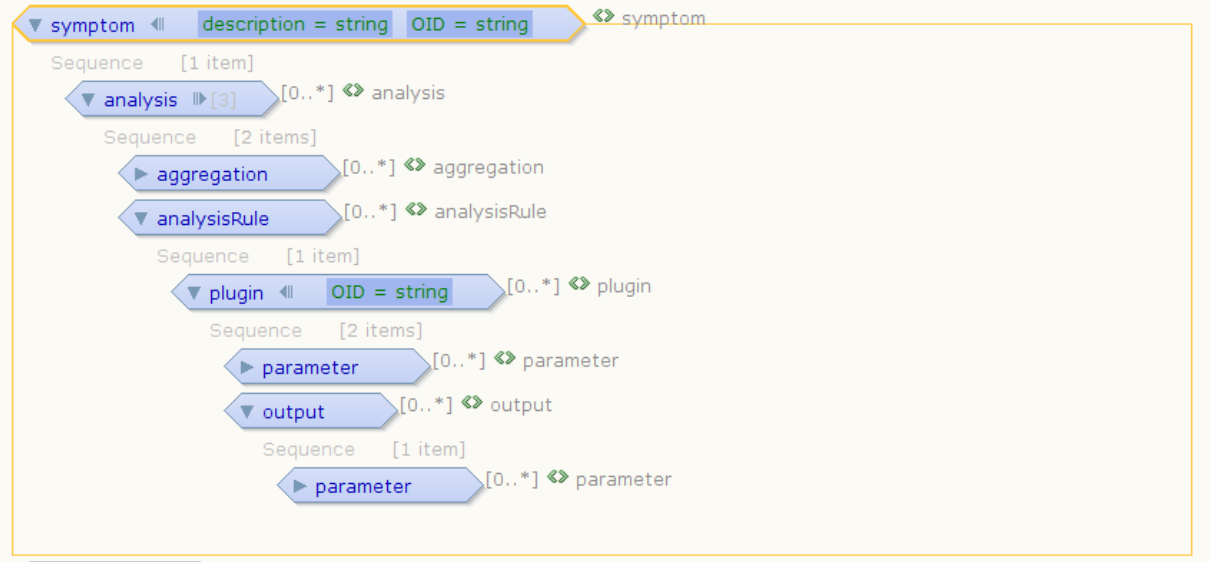

Figure 17: Symptom Definition

With the above overall schema (Figure 17), a **Symptom** is defined. The term symptom is adopted from SelfNet to highlight a detectable situation for which an FCAPS management automation can be applied. In case the trigger of a symptom is detected, SliceNet identifies a condition concept (Figure 18) according to which a **cause** matching functionality has to be applied to resolve the remedy that has to be applied.

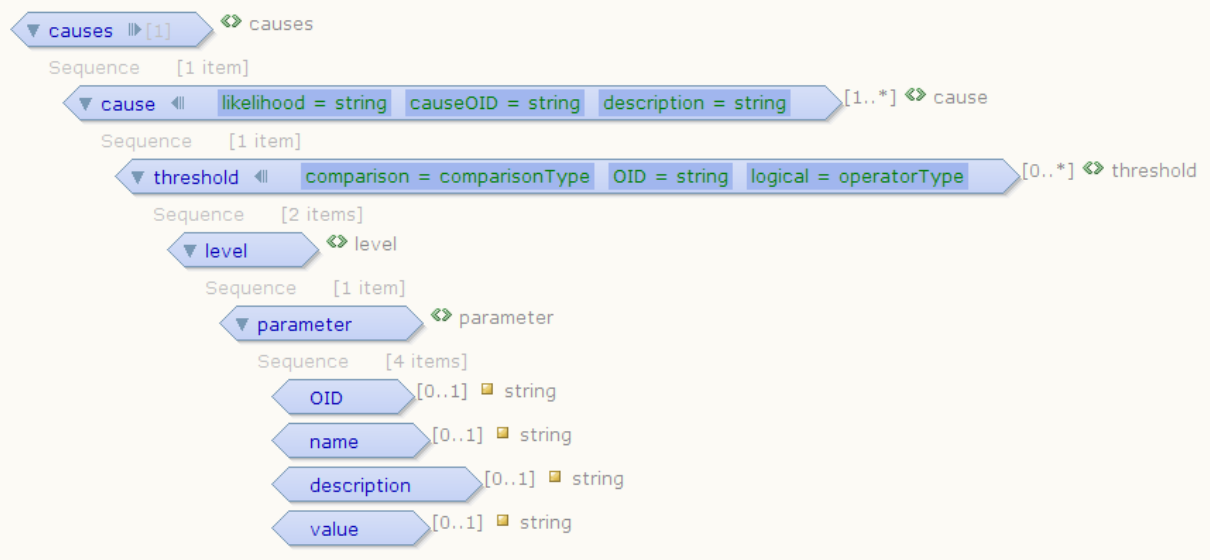

Figure 18: Condition Definition

The concept is the same with the threshold option for produced metrics. In this case, however, the condition should be evaluated either for any parameter that is available from previous steps or a special notation can be applied in the **name** of the **level parameter** to indicate that a value that has to be retrieved from another inventory or data set relating to the slice should be subject to the condition verification. The condition is associated with the **cause** object that is wrapping it. The notation for identifying the source of the value to be used in the condition verification will be further elaborated during later integration phases of the project, once the inventory and data sets are more elaborated and stable.

An example of the sensing part, without the option of a cognitive module is given in the following extract where the *uplinkBytes* field of a *ran* sensor are subject to be aggregated using an *average* function applied every minute. The result is stored as an *avgUp* metric subject to comparison with a threshold value that expects to trigger if it is found 20 consecutive times below it.

```
                    <ns0:aggregation>
                        <ns0:aggregationItem>
                            <ns0:aggregationData>
                                <ns0:sensor OID="ran">
                                    <ns0:output>
                                        <ns0:parameter>
                                            <ns0:name>uplinkBytes</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>imsi</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>ip</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>pduId</ns0:name>
                                        </ns0:parameter>
                                    </ns0:output>
                                </ns0:sensor>
                            </ns0:aggregationData>
                            <ns0:aggregationRule>
                                <ns0:ruleItem>
                                    <ns0:average>
                                        <ns0:parameter>
                                            <ns0:name>uplinkBytes</ns0:name>
                                        </ns0:parameter>
                                        <ns0:period duration="00:01:00Z"/>
                                    </ns0:average>
                                </ns0:ruleItem>
                                <ns0:metric>
                                    <ns0:name>
                                        <ns0:parameter>
                                            <ns0:name>avgUp</ns0:name>
                                        </ns0:parameter>
                                    </ns0:name>
                                    <ns0:dimensions>
                                        <ns0:parameter>
                                            <ns0:name>imsi</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>ip</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>pduId</ns0:name>
                                        </ns0:parameter>
                                    </ns0:dimensions>
                                </ns0:metric>
                            </ns0:aggregationRule>
                            <ns0:threshold comparison="lessOrEqual">
                                <ns0:level>
                                    <ns0:parameter>
                                        <ns0:name>avgUp</ns0:name>
                                        <ns0:value>65536</ns0:value>
```

```
                                </ns0:parameter>
                            </ns0:level>
                            <ns0:occurrences>
                                <ns0:consecutiveOccurrences>
                                    <ns0:numberOfOccurrences>
                                        20
                                    </ns0:numberOfOccurrences>
                                </ns0:consecutiveOccurrences>
                            </ns0:occurrences>
                        </ns0:threshold>
                    </ns0:aggregationItem>
                </ns0:aggregation>
```

Each rule is processed in a separate workflow thread which maintains an environment with all the variables that have been either collected or produced in the various steps. This environment's variables can be referenced by subsequent steps by defining in the parameter values the name of the referenced variable in curly brackets. This should lead to dynamic replacement of the current value so that the workflows can be executed as intended.

4.3.2 Tactic Branch

In the context of the rule definition, the tactic branch provides the strategies to be applied once a cause has been verified for a detectable situation. A single situation can be handled in different ways based on the condition that has been verified. As it was stated above, a condition is associated with a cause. In the tactic branch (Figure 19) one cause is associated with the adjacent **action**. The action can be a sensor configuration, an actuator invocation or a resource action. A sensor configuration can indicate either the modification of the sensor tuning or deployment of a new one among those available in the slice offerings catalogue. Actuation option can hold any operation invocation, either this is provided by the orchestration subplane or by the Control Plane services. Resource option is a generic placeholder. The initial purpose of this option was to indicate resource management requests like scaling or migration. This, however, is not applicable in SliceNet as a direct request as it is expected that it can be indicated via operations provided by the orchestrator. Re-engineering this field was an option, however, it was decided that the option is maintained as such and has to be used to indicate a data management option. This option can be either an additional data entry in the time series tables allocated to the slice, to be processed in the context of other rules, or an export option of the collected/produced information towards the DSP data sets to be processed in higher level workflows.

The tactic branch is expected to be elaborated in the next integration phases, once catalogue, inventory and data lake components will be more mature.

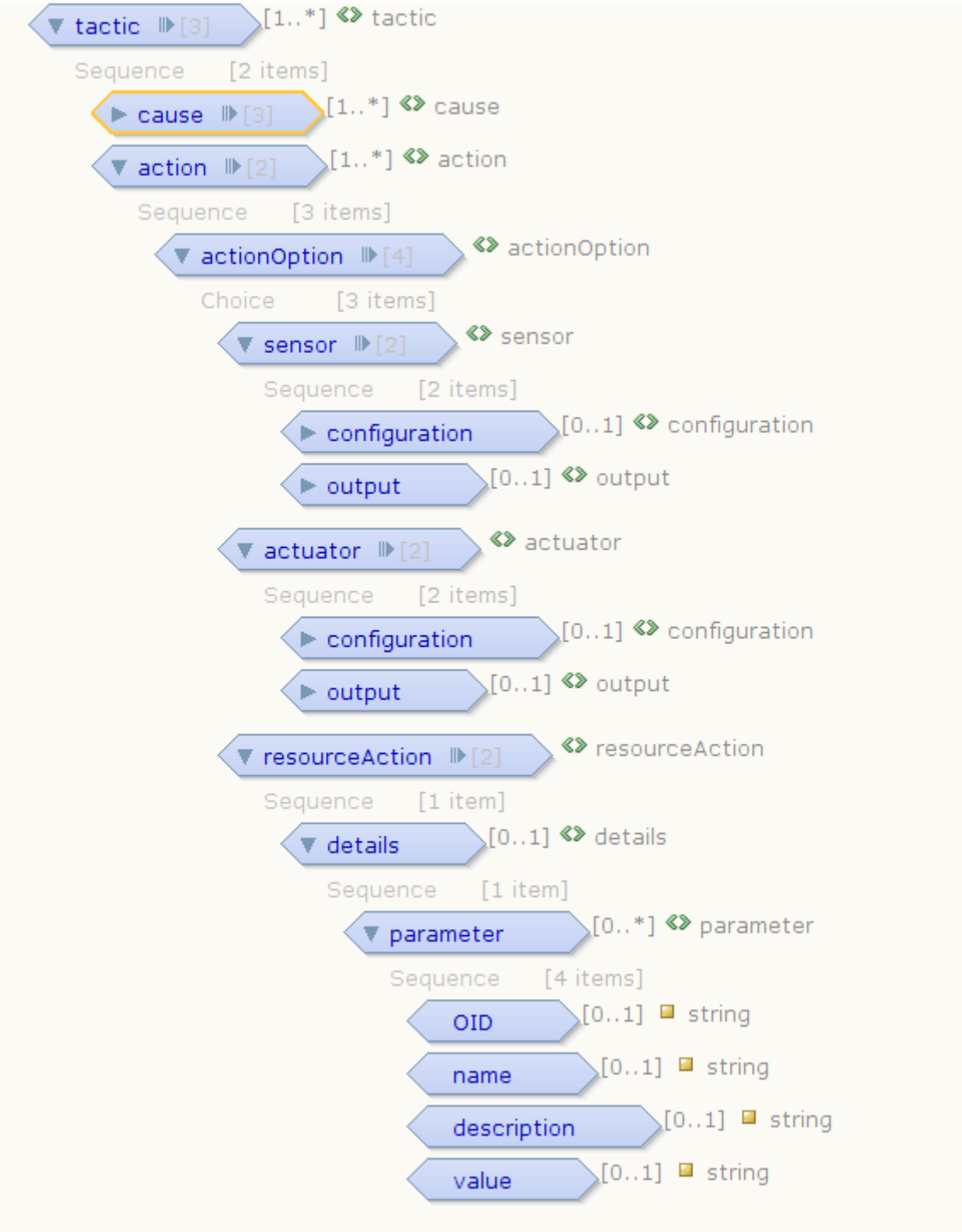

Figure 19: Tactic Definition

4.4 Faults and Alarms

The fault management systems traditionally adopted by network operators and service providers are based on monolithic and all-in-one frameworks that display alarms from a single point supervising a heterogeneous plethora of technologies, resource domains, vendor solutions. This approach must be adapted in modern 5G network where more programmable and softwarized solutions are adopted, leveraging on the NFV and SDN software networks

principles, which enable abstraction and unified control and management procedures and APIs with the aim of avoiding any vendor lock-in for network operators and service providers.

In addition, the future 5G network will enable multiple management entities playing different roles in the shared network. E.g. Infrastructure Provider (InP), Mobile Network Operators (MNOs), Virtual Network Operator (MVNO), Over-The-Top (OTT) service providers. In all cases the allocated network slices can be provided on a permanent basis or on-demand. SliceNet aims to realize future flexible, on-demand multi-tenant networks designed as signaling-based, i.e. for minor or no human intervention. According to the network evolution described above, SliceNet assumes that multidomain provisioning of slices involves several players from more than one layer such as NSP, DSP. Faults and alarms management is fully integrated in the SliceNet architecture and follows the general SliceNet architecture principles such as: on demand sensing and actuation management, a dynamic sub-slice driven instantiation and configuration and a control loop automation.

The above described principles have led to the definition of the following architecture outline of the single domain (NSP) fault management function that is fully integrated in the FCAPS management subsystem described in the previous sections. Figure 20 depicts a representation of the architecture of Fault Management, evidencing the main modules responsible for the Fault Management per a single network sub-slice instance (NSSI).

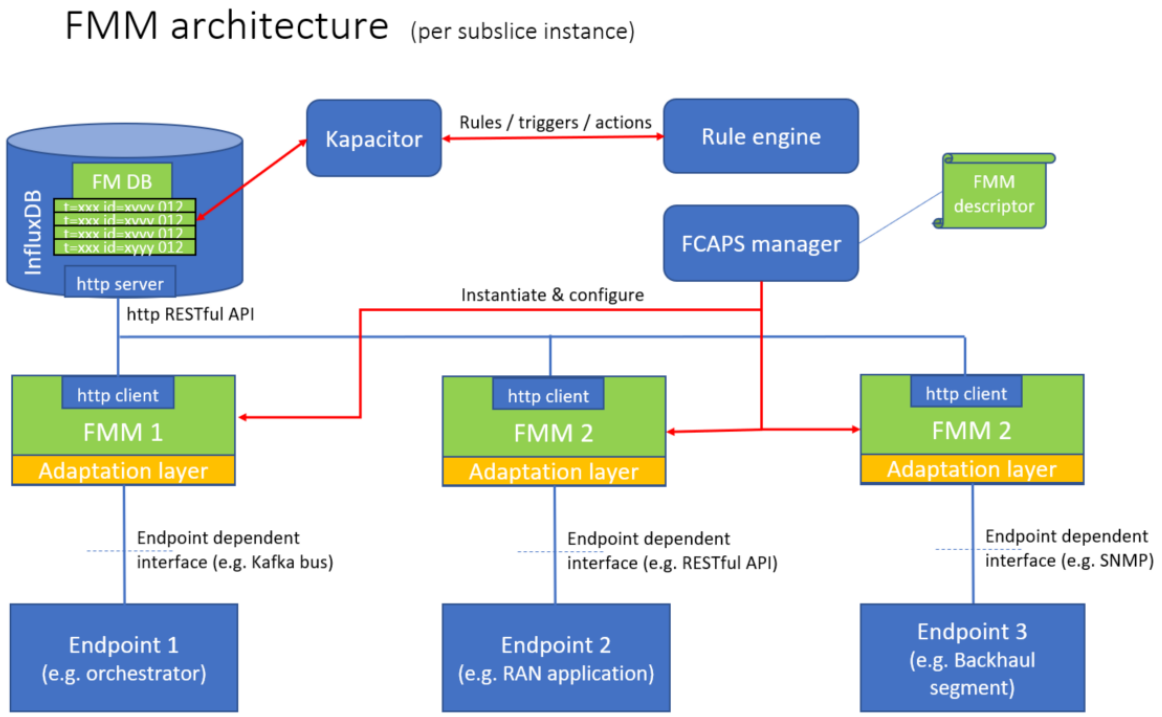

Figure 20: FMM general architecture per NSSI

In Figure 20 the lower blocks represent the endpoint generating events and alarms. There could be several endpoint/sensors involved in a single NSSI, e.g. RAN segment, Core segment, Backhaul segment, etc. Each endpoint is interfaced by a specific module called Fault Mediation Manager (FMM), which has the role to interface the endpoint adapting to the endpoint specific interface (e.g. Kafka bus, SMNP, API, etc…). FMM aggregates the alarms

(e.g. per Slice id) and populates InfluxDB data base for the relevant slice. While doing this FMM has to transform the raw events received by the endpoint in a general alarm format, agnostic to the specific endpoint (See 5.3 for further details on FMM and agnostic alarm format).

The event and alarm data collected by all the endpoints for the relevant slide are stored in InfluxDB and exposed to the consumer entities such as Kapacitor / Rule engine. Actions could be triggered according to the event-condition-action workflow as described in section 4.3 . The FMM descriptor for the relevant NSSI contains the information needed by FCAPS framework (e.g. APP engine) for the specific NSSI. E.g. FCAPS manager should be informed about which end point the particular slice has to supervise in order to instantiate the relevant FMM module. FCAPS manager has also to be instructed to properly configure the endpoint (e.g. subscribing to a set of alarms via FMM).

4.5 Accounting

The proposed Accounting management system is following the general SliceNet FCAPS framework and can be regarded as a simplified case of the monitoring management system. The overall objective is to collect relevant accounting data that can be grouped per slice or sub-slice within the NSP domain and permanently stored in InfluxDB as e.g. Charging Data Records (CDR) that are exposed to SliceNet external Charging/Billing Systems like e.g. BSS for further processing. As the accounting workflow is following the performance monitoring workflow, just a brief description is reported below avoiding repetition of details: Given the large amount of data that can be potentially used for accounting, possible alternatives can be onboarded in the form of accounting descriptors that are used during instantiation of the (sub-) slices within the domain of the NSP. In this way, accounting artefacts are instantiated and configured to collect information from the indicated end-point and, then, per single slice accounting records are populated and stored into the InfluxDB database to be consumed by external Charging/Billing System. The accounting artefact can be properly configured Telegraf instances or specific SW components instances. Example of useful accounting data to collect are:

- From the Inventory, number and type of NSP resources used, time stamps indicating beginning and end of slice activation.
- From the infrastructure: amount of used compute, resource and storage.

Unlike the monitoring workflow, in this case no analytics options are applied on available stored data in terms of rules definition to trigger events or data forwarding as in SliceNet there are no specific requirements for that. However, the actual FCAPS framework allows for future enhancements to accommodate, for example, support for real time charging features by defining specific rules to trigger charging events to external Billing/Charging entities.

Figure 21 shows the architecture of Accounting Management, evidencing the main modules responsible for the Accounting Management per a single NSP slice / sub-slice instance (NSSI).

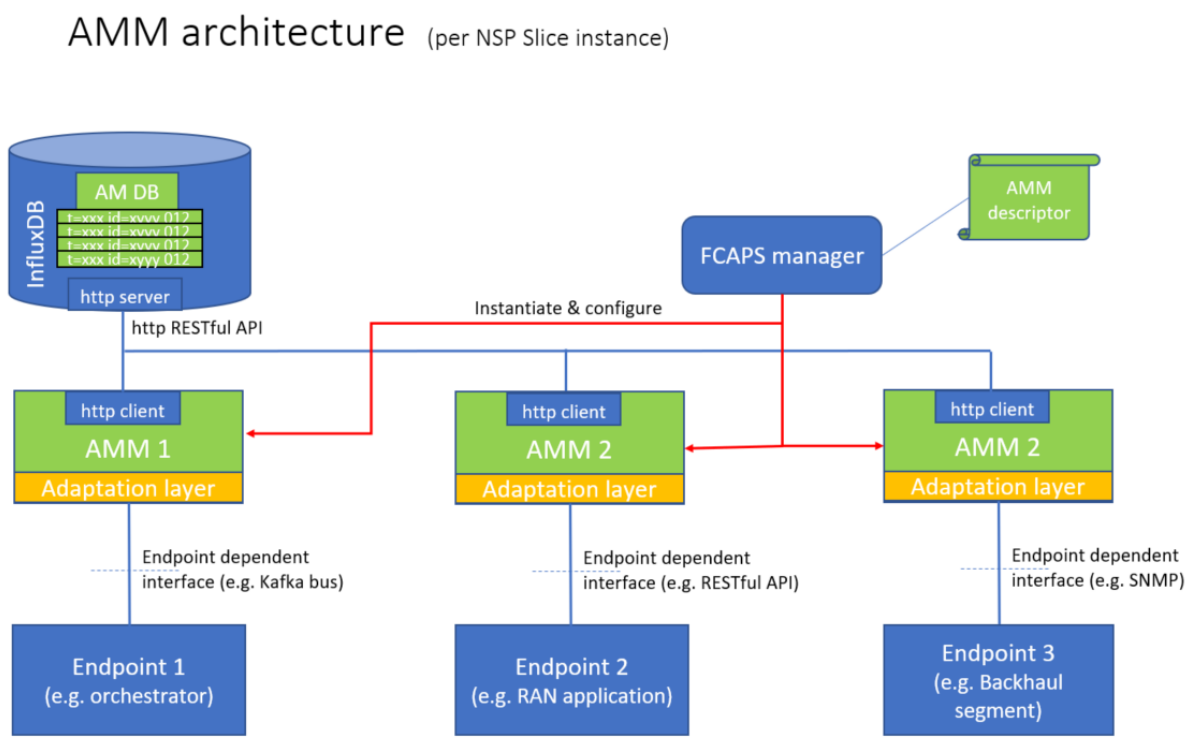

Figure 21: Accounting general architecture per NSSI

The lower blocks represent the endpoints to collect account data. There could be several endpoints involved in a single NSSI (e.g. RAN segment, Core segment, Backhaul segment, etc.).

4.6 Security

In NGMN (NGMN security guidelines [27]), the security concerns from architecture perspective are impersonation attacks against various network slice components, unintentional or malicious exhaustion of resources, and isolation of network slice resources. Besides these, other vulnerabilities can very well occur, and the present document assumes that they will be addressed as per the network-design and best practices known for security of virtualised infrastructures (such as, security of NFV, virtual network functions, etc.). SliceNet follows this approach but initiates it with an E2E service Blueprint filled in by the vertical/DSP, e.g., the security descriptor will be inserted into the E2E Service Blueprint/Network Slice Blueprint where the vertical/DSP specifies in the Blueprint the security options when requesting a service. The DSP/NSP orchestrator will translate this security descriptor into a set of virtual security network functions (vSNFs) to insert into the E2E service slice/network slice. To design the set of vSNFs that are offered to E2E service slices/network slices, SliceNet follows the KPIs for Security and Privacy that are defined in ETSI GS NGP 013 V1.1.1 (2018-09). Table 5 shows the categories and definition of these KPIs.

ID	Definition and rationale	Metric
SEC1	Security by default Security achieved without overlays.	Yes/No
SEC2	Crypto-agility for algorithms	Yes/No

	Key management independent of function invocation.	
SEC3	Reporting of security events to a recognized standard.	Yes/No

Table 5:Security and Privacy KPIs defined in ETSI GS NGP 013 V1.1.1 (2018-09)

Proposed security descriptor to be inserted in the blueprint.

```
"security" :
{
  "KPIs": [
    SEC1 : "Yes/No",
    SEC2 : "Yes/No",
    SEC3 : "Yes/No"
  ],
  "encapsulation": "L3VPN/L2VPN/MPLS/etc.",
  "segmented encapsulation": "Yes/No",
  "encryption": [
    application : "NONE/WebRTC/?"
    transport : "NONE/SSL"
    IP : "NONE/IPSec"
    ]
  "key" : "key management mechanism",
  "..." : "..."
}
```

Figure 22: SliceNet security descriptor

Following this guidance, SliceNet has studied these KPIs with different technologies to offer the performance indicators for tool/software/algorithms selection and evaluation. SliceNet will provide security by design for SEC1 and SEC2 and security at runtime for SEC3. When a network operator designs a new slice, which is composed of virtualized (VNFs) and non-virtualized resources (PNFs), he can add into the slice certain virtualized security network functions (vSNFs) such as vFW, vIDS, VDPS, etc. in different locations to have a certain level of security in his slice, depending on the characteristics of the designed slice. SliceNet intends to automate this process through the service blueprint, SLA management, framework policy and orchestration. When this slice is onboarded and instantiated, so it goes to the operation/runtime phase, a cognitive mechanism (autonomic security management) can be used for real-time analytics on the chained PNFs/VNFs to detect possible threats and prevent or minimize the impact of those detected threats on the system.

4.6.1 SEC1 – Security by default and Security achieved without overlays.

For **SEC1**, SliceNet targets to provide slice isolation by default, along with policies-based, rules-based security. To achieve security slice isolation, SliceNet provides the ***services isolation for slice instances*** with one-to-one relationship between the set of SliceNet services (CP services)/component instances (QoE, P&P) and the slice instances and supports ***encapsulation for E2E slice instances.*** The design of SliceNet Control Plane supports both E2E encapsulation and segmented encapsulation mode. Given the E2E slice requirements

(either specified in the blueprint or SLA agreement), this can translate into proper slice traffic tagging or encapsulation (e.g., L3VPN, L2VPN, MPLS, etc.) where applicable following SDN principles. Minimum set of information required as input to perform this operation includes the identification of the set of peering domains to interconnect to (e.g., a list of remote endpoints) and the traffic tagging/encapsulation to be adopted. Details of information model for E2E slice requirements including the encapsulation are in WP4 deliverables.

In addition, to achieve SEC1 with security by default and without overlays, policy-based security is also considered. The complexity of security mechanisms grows in the 5G networks not only due to virtualization of resources but also due to security requirements at different levels or domains such as network slice, network service, and network resource (physical & virtual) and RAN slice. Hence, a security management system, guided by a set of defined security policies, is essential to ensure that security mechanisms functions are enforced as planned.

- Rules and legal procedures to access the system and to modify entities characteristics.
- Access control on the system entities (slice instances, VNF instances, etc.).
- Role/Privilege based policies, such as identifying authorized and unauthorized services/processes any user can perform on the network.
- Policies & rules-based security (security policies applied to particular tenants, firewall rules applied to port, etc.)

In SliceNet, the first three items above can be abstracted under NSP orchestrator rules/policies and configuration while the last item will be identified by the FCAPS rule engine.

4.6.2 SEC2 – Crypto-agility for algorithms; Key management independent of function invocation

This section will explore the cryptographic algorithms, cryptographic key management and secure protocols, then it will discuss E2E encryption with options to have encrypted communication channels and E2E data encryption. Methodologies for cryptographic algorithms and key management will be covered in this section to provide guidance for the security option selections in designing slices toward verticals.

Crypto-agility for algorithms

Crypto-agility is simply the flexibility for moving from one implemented cipher to another without having to redo or rewrite everything. Today, almost all the systems/organisations are secured with digital cryptography but there is no guarantee that a secured system is unbreakable forever. All cryptographic algorithms will eventually fall over time and being replaced by new cryptographic algorithms that can patch the previous algorithms' weakness, e.g., SHA-1, MD4, MD5 are no longer considered as secure algorithms, RSA-2 has the same weakness but with increased length for harder breaking, RSA asymmetric cipher has been under constant attack since its introduction in 1977, e.g., ROCA attack, ROBOT attack in 2017, WPA2 KRACK attack in 2017, etc. This is the reason that any *system design needs to be crypto-agile, expecting to migrate quickly to alternative cryptographic primitives and algorithms.* The standard X.509

public key certificate is an example of crypto agility. It is defined by the International Telecommunications Unions' Standardization sector (ITU-T), and support different cryptographic parameters including key type, key length, and a hash algorithm where the system can specify any conforming cipher to create asymmetric keys and certificates and then indicate which one is being used on the certificate so that the other parties know how to read and use it appropriately. By providing the generic functionalities with a list of parameters, it can easily integrate the latest cryptographic technologies by implementing the new cryptographic algorithms and insert it into the list of parameters' options. The X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure web browsing protocol.

Key management independent of function invocation

Cryptographic keys are used to secure/protect data-at-rest and data-in-motion. However, trying to keep them protected yet always available for use is one of the most difficult problems in practical cryptography.

There are two types of cryptographic keys and encryptions: asymmetric and symmetric. The asymmetric key is composed with a public and private key pair where public key is distributed to parties to encrypt/decrypt the data and the private key is for the owner to decrypt/encrypt it. Asymmetric encryption uses a pair of public key and a private key to encrypt and decrypt messages when communicating and is mostly used in today communication channels, especially over the Internet. Popular asymmetric key encryption algorithm includes ElGamal, RSA, DSA, Elliptic curve techniques, PKCS. The symmetric key on the other hand, is a single key that is used to encrypt and decrypt the data. Symmetric encryption uses a single key that needs to be shared among the people who need to receive the message. Blowfish, AES, RC4, DES, RC5, and RC6 are examples of symmetric encryption. The most widely used symmetric algorithm is AES-128, AES-192, and AES-256. Asymmetric and symmetric encryptions are complementing each other, e.g., asymmetric encryption is to eliminate the need to share the key in symmetric encryption by using a pair of public-private keys while symmetric encryption is to reduce the encryption time that relatively taken too long in asymmetric encryption. With symmetric key algorithms, data can be encrypted and decrypted at a much higher rate than with asymmetric algorithms offering equivalent security. Many secure communication protocols today make use of asymmetric cryptography to establish a shared symmetric key among all shared partners, and thus leveraging the convenient security of asymmetric cryptography and the performance benefits of symmetric cryptography.

The key management refers to all procedures related to generation, exchange, storage, safeguarding, use, crypto-shredding (destruction) and replacement of keys. This is one of the most critical components of a security system and also is one of the most difficult to find an optimal solution for, due to many challenges including:

- Key generation, complexity of the generator
- Key distribution, secure key delivery and recovery
- Secure key storage and recovery, loss or corruption of these keys can lead to loss of access to systems and data, as well as making a system completely unusable unless it is reformatted and reinstalled
- Verification and validation of the (shared) keys
- Surveillance of the keys
- Key lifecycle management and automation

The challenges grow with the size and complexity of the system environment. Improper key management can lead to key leakage, where an attacker obtains the key and recovers the sensitive messages from the encrypted data. One of the solutions for key generation and distribution is that the user generates the keypair itself and distribute its public key to other entitles for data encryption where only the user can decrypt the data with his own private key. The advantage of this solution is that it has never revealed the private key to external and thus guarantees the secrecy of the private key. However, the attacker could steal the user identify and pretend to be that user, with his own keypair for encrypting and decrypting data. To resolve this attack, alternative solution requires 3rd party, such as a Certificate Authority (CA), that issues digital certificates, which certify the ownership of the public keys by the named subject of the certificates, e.g., CA signs the public keys to verify the ownership of the keys. There is also information encoded in public key that verifies the private key and certificate validity. If the issuer (CA) is trusted and the signature is valid, the certificate's subject (owner of the key) is authenticated and the keypair is now can be used to communicate securely with the owner. Another option is that the CA will generate the keypair, sign the key, issue a digital certificate and then securely distribute to the user. After the key generation, the security of key distribution phase is critical to ensure that the private key remains protected.

A public key infrastructure (PKI) is a set of roles/policies/procedures to create, manage, distribute, use, store and revoke digital certificate and manage public-key encryption. A typical PKI consists of i) a certificate authority (CA) that stores, issues and signs the digital certificates; ii) a registration authority (RA) that verifies the identity of entities that own the digital certificates to be stored at the CA; iii) a central directory to securely store and index keys; iv) a certificate management system to manage the accessibility of the stored certificates and certificate delivery; and v) a certificate policy to analyse the PKI's trustworthiness.

SliceNet design goal in this SEC2 is to provide an E2E channel/data encryption by exploring the options for secure communication protocols with VPN and E2E data encryption with OpenSSL and Vault, taking into account the above concepts. PKI providers (with CA, RA) should be the entities/organisations that already have trusted relationships or implemented in a company for managing internal security aspects, and thus it is out of scope in SliceNet project.

E2E channel encryption

A VPN (Virtual private network) is an extension of a private network across a public network, allowing users to connect to it, send and receive data across shared or public networks as if they are in the private network. To ensure security, the private network connection is established using an encrypted layered tunnelling protocol, and VPN clients can connect to the tunnel with authentication methods, e.g., passwords/certificates, to gain access to the private network. VPN can support remote access for users and also site-to-site connection, connecting two or more private networks as a single logical private network. By supporting VPN, SliceNet can provide an E2E encrypted channel for particular use case that requires security and privacy such as eHealth UC.

There are three main VPN networking protocols spanning across Layer 2, Layer 3 and Layer 4 in the OSI model:

- Point-to-Point Tunnelling Protocol (PPTP) and its extension, the Layer 2 Tunnelling Protocol (L2TP). These protocols are not robust and have many well-known security issues, e.g., the PPP authentication, MPPE protocol design, etc. Examples are the

Microsoft Point-to-Point Encryption (MPPE) and Microsoft Secure Socket Tunnelling Protocol (SSTP).

- Internet Protocol Security (IPSec) which was initially developed by IETF for IPv6 but the protocol is also widely used with IPv4 and Layer 2 Tunnelling Protocol. IPSec can be used for both site to site and remote user connectivity and provides flexibility and strength in depth, however, it is a complex framework to work with.
- Secure Socket Layer/Transport Layer Security (SSL/TLS) that can secure individual connections or tunnel an entire network's traffic, currently used in OpenVPN and SoftEther VPN. This protocol provides excellent security for remote access users as well as ease of use and is heavily used for secure web browsing today.

For security purpose, open-access and ease of deployment, *SliceNet promotes the SSL/TLS to be exploited further to secure the communication channel*.

E2E data encryption

As discussed in the topic of cypto-agility for algorithms and key management independent of function invocation, SliceNet also supports user data encryption where it provides encryption service to the users. In addition, the encryption service can also be implemented as a VNF which can be inserted into specific slice as indicated in the security descriptor, e.g., vEaaS VNF that can be orchestrated through the slice creation and instantiation process. This is to demonstrate the flexibility of the SliceNet framework that can supports different alternatives. The idea is that this encryption service, either implemented as an independent service for any use cases or as a VNF inserted into a slice that requires data encryption, provides API function calls to users/applications for key generations (agile-approach), data encryption and data decryption. Depending on the security descriptor and use cases, channel encryption and data encryption can be used together but then it will slow down the performance significantly (due to huge overheads). Also, current SSL/TLS secure channel implementations (e.g., OpenVPN) already include cryptography to protect the user data. For this reason, it is recommended to use only one of those for E2E encryption.

4.6.3 SEC3 – Reporting of security events to a recognized standard.

The SEC1 and SEC2 are covered by a set of SNFs that will be inserted/orchestrated by the orchestrator based on the security descriptor and requirement from the vertical. The work is done by design and onboarding time. In this SEC3, SliceNet provides a mechanism for security events (e.g., threats, anomalies) sensing and monitoring, detection and response during the runtime of the slice instances, the generic framework and workflow are as in Figure 23

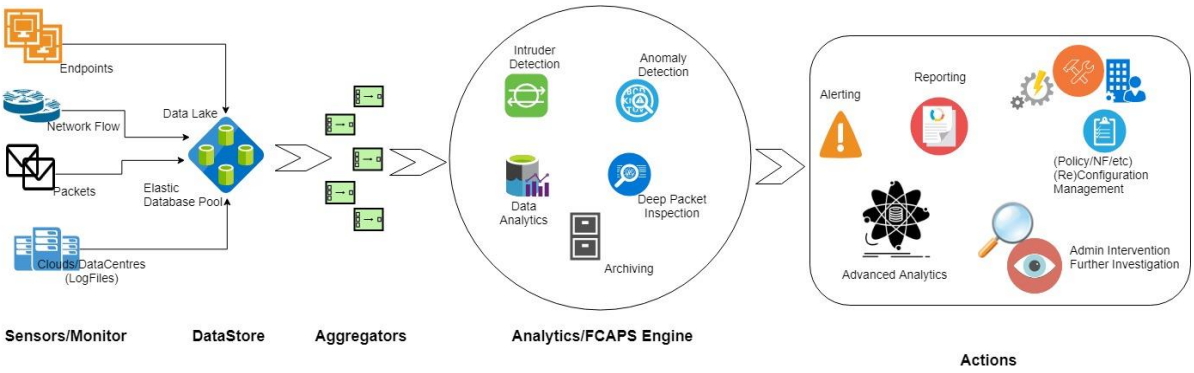

Figure 23: SLICENET Security Events Monitoring, Detection & Response Framework

This section explores the security principles and how SliceNet can support these. Later section 5.4 will explore the options for SNFs that the project will implement and deploy for UC demonstration that cover the above recommended KPIs for security and data privacy.

5 Description of Single-Domain Management Plane FCAPS components

5.1 TICK stack

TICK Stack is a platform composed of open source components which supports collection, storage, graphing and alerting on-time series data (such as metrics and events). The components of the TICK Stack are: Telegraf, a server agent for collecting and reporting metrics; InfluxDB, a high-performance time series database; Chronograf, a user interface for the platform; and Kapacitor, a data-processing engine that can process, stream and batch data from InfluxDB. TICK Stack is based on the Push model of collecting data. The heart of the system is the InfluxDB component which is a time series database.

FCAPS framework includes InfluxDB to store persistent data collected by different endpoints. Telegraf is conveniently configured to collect data from some endpoints, and Kapacitor is used for data processing.

Prior to adopting the utilisation of the components of the TICK stack, other monitoring frameworks were evaluated. The evaluation indicated that the TICK stack combines performance with granularity and high customisation while being less demanding with respect to resources, which fits in the context of per slice and on demand monitoring. Other cases are proved to be either resource demanding like ElasticSearch or technology specific like Monasca.

5.2 FCAPS Automation Framework

The FCAPS automation framework handles all the workflows for the instantiation of the required logic that aims at supporting the runtime phase of a slice. This logic involves collection and maintenance of monitor-able information and performance counters, harmonisation of this information so that it can be used in a transparent and technology agnostic manner by higher level workflows aiming at ensuring the operation of a slice within the borders posed by an SLA, activation of rule based management that support autonomous operation of the slice as far as the SLA details are concerned.

The operation of the FCAPS automation framework (Figure 9) relates with slice catalogue and inventory information (Figure 24). This information is annotating the slice NF topology with options regarding monitoring and actuation capabilities. Those capabilities are slice specific and reflect the features that have been clarified between NSP and DSP. There are two running tasks launched by the FCAPS Manager for this purpose. One task processing the entries in the FCAPS Descriptors Catalogue and another one scanning the Slice Inventory for detecting changes (e.g. association of an NF with a monitoring descriptor) that have to trigger the instantiation of monitoring artefacts or updates to the NF-CONFIG CPS for actuation workflows.

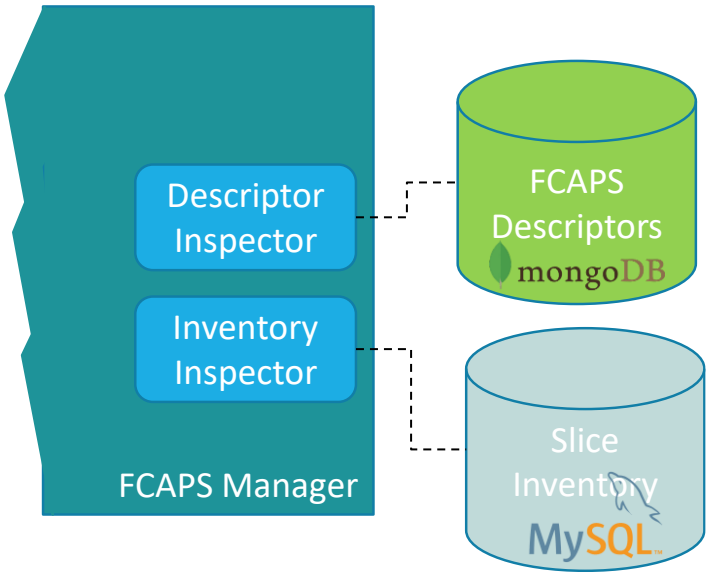

Figure 24: Slice Instance Annotation Processing

In the cases where an FCAPS descriptor references a custom artefact that has to be involved in the process of monitoring, the Type Management Logic is triggered to prepare a Docker image which contains the binary package indicated (Figure 25). The image is not instantiated directly but it remains available to be instantiated when the related descriptor is associated with an NF.

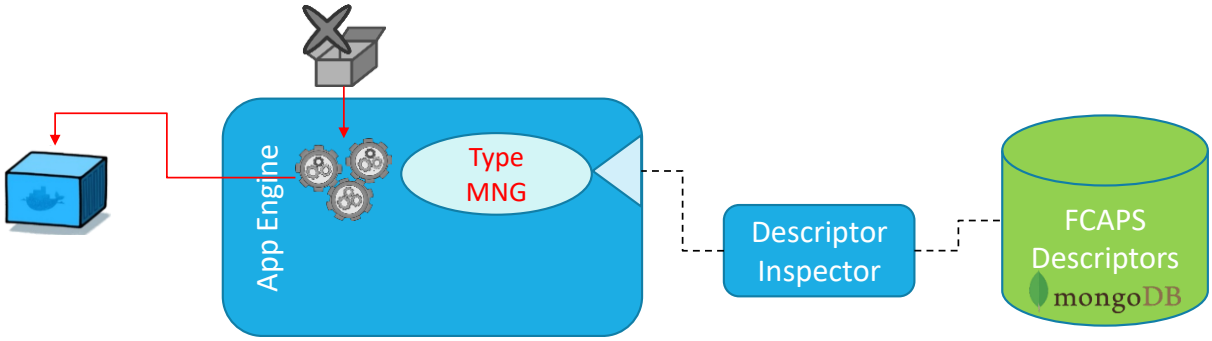

Figure 25: Custom Artifact Management

The Inventory Inspector task checks for association between descriptors and NF instances (Figure 26). From the descriptor details a variable set is extracted. The variables regard all the configurations that have to be used for activating the monitoring artefacts for an NF. Once the values for this variable set can be retrieved from the Slice Inventory, the monitoring artefacts can be activated. If a custom artefact is defined in the descriptor, the docker image that has been built in the previous step is bootstrapped with the configuration set and the launch command is invoked upon container creation. Similarly, the protocol section of the descriptor leads to the activation of a generic Telegraf based container image. The Telegraf configuration is composed on the fly according to the variables values that have been collected. In case the Telegraf instance has to communicate with the custom artefact instance that applies a particular monitoring data pre-processing, both containers are attached to the same docker network.

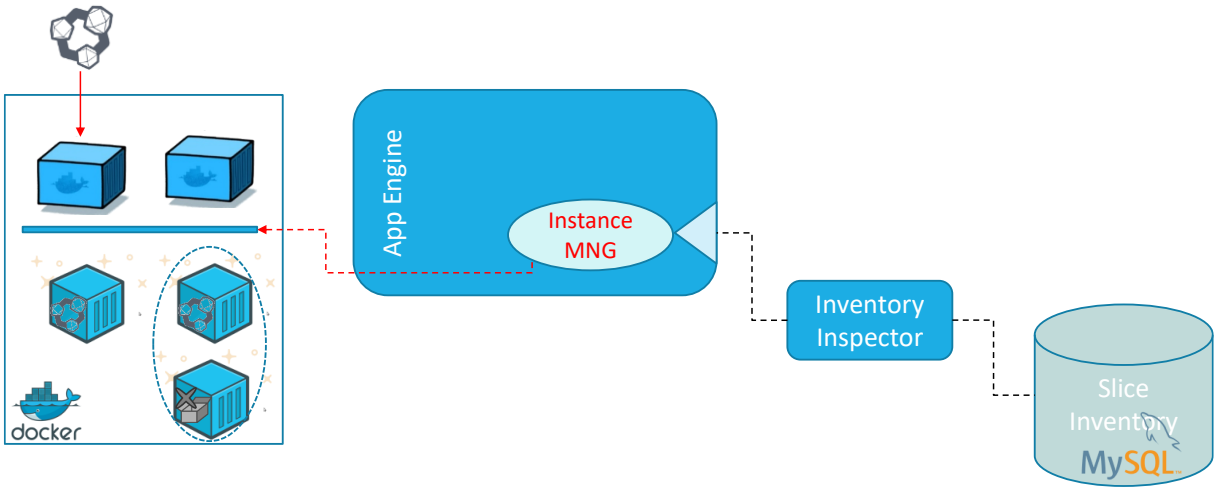

Figure 26: Monitoring Artifact Management

Upon activation, these containers are storing data to an InfluxDB by creating slice specific databases and measurement related tables (Figure 27). The Slice identifier is used as database name and the descriptor name type is used to identify the measurements table within that database. Up to this point in the operation of the FCAPS subsystem, the slice counters are made available according to the requirements that have been identified through a higher level procedure that takes place in the interaction between DSP and NSP. This interaction is performed in the context of customisation of the Vertical request and the requirements are resolved by the selection of the offerings an NSP is supporting for the set of NFs it is making available as federated items.

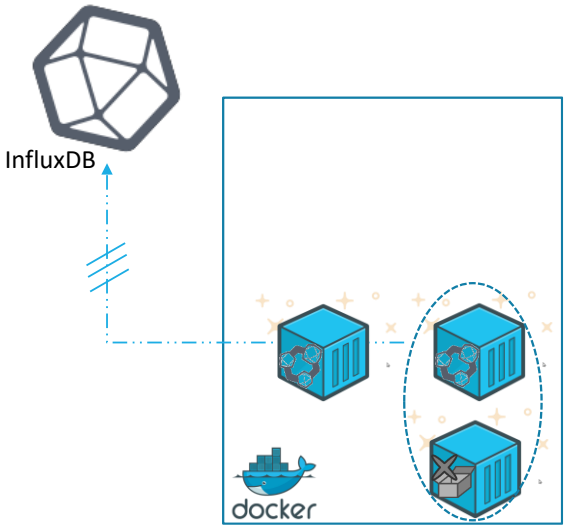

Figure 27: Slice Time Series Storage

The use of the produced time series is dictated by the autonomic rules distributed as policies that have been identified for a slice (4.3). These rules are affecting the Rule Engine that is the other major component of the FCAPS automation framework. In the context of these rules, a number of reactions are defined. The reactions are pushed to the Rule Engine via a REST based interface per slice, according to the required autonomic behaviour that has to be applied.

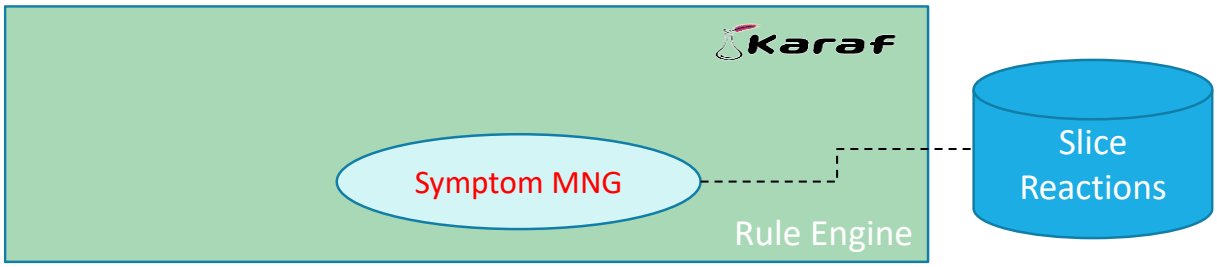

Figure 28: Rule Engine Configuration

The reactions are defined in the context of the filtering and aggregation of the stored counters, alarms, faults for a given slice. This aggregation and filtering is applied via dynamically composed Kapacitor scripts (Figure 2928) that identify the database from the slice identifier, the measurement table from the sensor name and the metric output from the related TAL definition. The threshold is used to create an alarm definition in the script. Kapacitor is configured to propagate this alarm via Kafka bus on a topic named under the name assigned to the symptom or condition that has to be detected. The rule engine apart from composing and activating the aggregation script launches a symptom instance object to collect the produced alarm. Each symptom detection object is able to apply condition evaluation and resolution of the tactic details that have been identified in the tactic branch of the rule/policy (Figure 29).

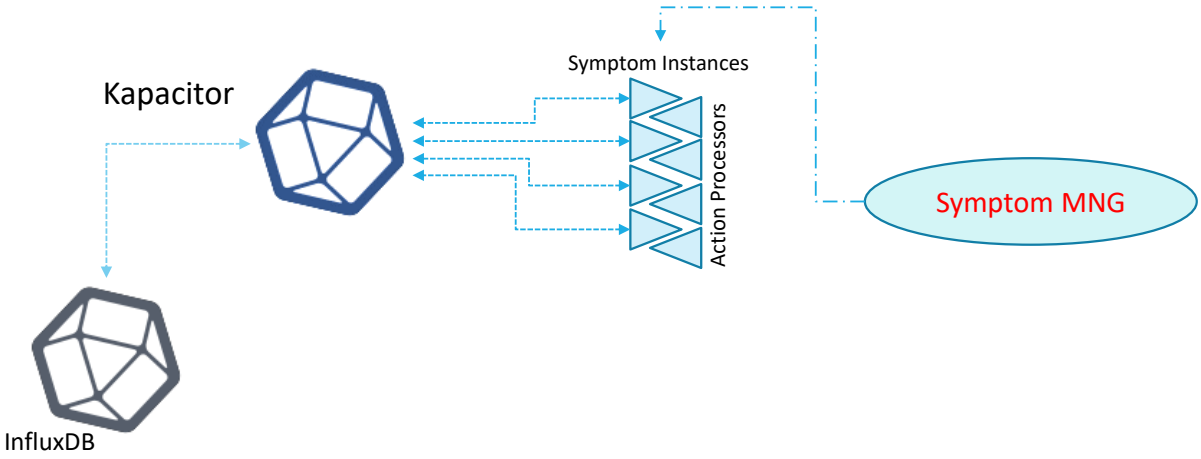

Figure 29: Rule Activation

The FCAPS Manager and the Rule Engine are deployed as Karaf Blueprints inside the Karaf runtime platform.

5.3 Fault Mediation Manager

The FMM can be realized either by properly configured Telegraf instances or by custom artefacts that provide the requested adaptation to the specific endpoint. Custom artefacts can be either coupled with a Telegraf instance for information persistence or directly connect and populate the InfluxDB. For both Telegraf and custom artefacts the FCAPS APP manager creates, configures and starts Docker containers as described in previous section 5.2 in the context of FCAPS automation framework.

Upon activation, the FMM container creates a new measurement table for the slice database where the alarms can be permanently logged. The FMM module has the task to collect the raw events data generated from the different alarm end points and to process them to normalize, flatten and map the events in a defined semantic, achieving a general alarm format agnostic respect to the specific alarm provider. The structure in Table 6 is proposed as agnostic format to be stored in InfluxDB measurement table for all the alarms generated by the different alarm end points relevant for a specific Slice id:

attribute name (*) are key tag	data type	cardinality	description
Alarm id	Alarm identifier	1	uuid of the alarm
NSI id (*)	NSI identifier	0..1	uuid of the network service instance (OSM - ETSI NFVI)
timestamp	Date time	1	Time stamp
geo_agg_code (*)	identifier	0..1	Geografic Location for alarm
Description (problem class)	string	0..N	alarm text description
Resource id (*) (entity id)	identifier	1	id of the faulty resource
Originating subsystem (*)	string	0..N	Subsystem (provider) generating the alarm (e.g. OSM, AWS, etc..)
Severity (*)	Enumerated (Critical, Major, Minor, Warning, Indeterminate, Cleared)	1	perceived severity of the alarm
Metric Name (*)	Enumerated (Compute, Storage, Network)	0..1	faulty resource type generating the alarm
Alarm raised time	date time	1	time stamp indicating when the alarm has been raised
Alarm changed time	date time	0..1	time stamp indicating when the alarm has been updated
Alarm cleared time (close time)	date time	0..1	time stamp indicating when the alarm has been ceased

Table 6: Agnostic format for alarms in InfluxDB database per NSSI

5.4 Security Management

This section will elaborate the security principles discussed above to further explore how SliceNet architecture and system framework can support those principles in practice. At the design phase, a network operator can add into the slice certain virtualised security network

functions (vSNFs) such as vFW, vIDS, etc. in different locations to have a certain level of security in the slice, depending on the characteristics of the designed slice. SliceNet intends to automate this process through the SLA management and orchestration (Figure 30).

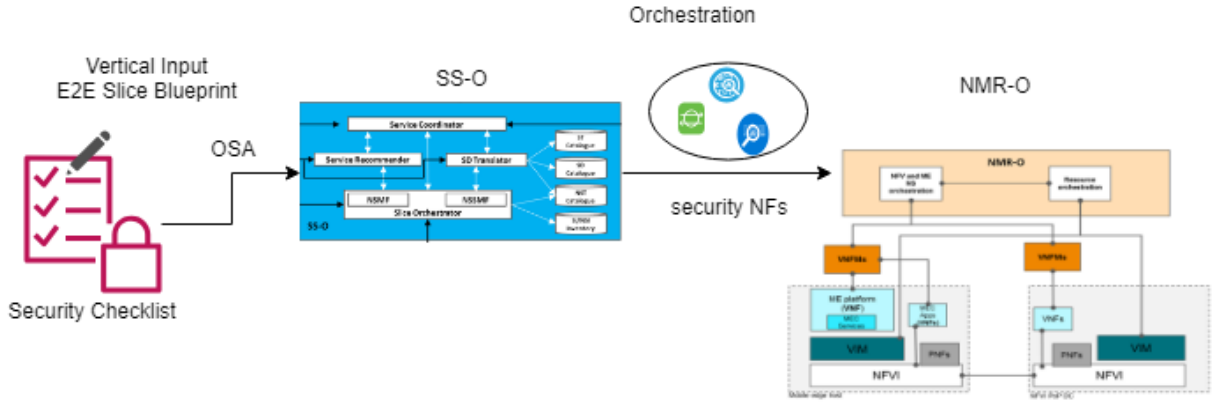

Figure 30: Orchestration of the security network functions

When the slice is onboarded and instantiated, the associated SNFs components are also being instantiated and allocated in the appropriate locations (NFVI, Switch/Router, Endpoints, etc.). The allocation is based on the design of the specific SNF and its configuration. Any actions that are associated in the SNF instance's INSTANTIATION/CONFIGURE event (in its lifecycle) will be triggered. After the event, the SNF instances will go to the operation/runtime phase where security events being monitored and analysed based on its functionalities. For study case, SliceNet will investigate different SNFs options including, but not limited to, firewall, encryption, IDS.

Firewall

In SliceNet, Openstack is selected for the functional block of VIM (Virtual Infrastructure Manger). Openstack offers a range of security aspects including policies-based security group and rule-based firewall group. The VIM administrator can identify a set of rules in a security group that he wants to apply for a tenant. The firewall group (e.g., neutron-fwaas service) is to indicate that a firewall consists of two policies: an ingress policy and an egress policy and it is applied at the port level (router ports, VM ports). Security group and firewall group are coexisted in Openstack VIM. The Opensource MANO OSM also supports to enable security group and security port in the Openstack VIM by setting the field name (e.g., port-security-enabled) in the VNF descriptor (VNFD) to true when onboarding a VNF.

Besides those existing services, SliceNet also has the option to have a stand-alone vFirewall VNF such as ufw or pfSense.

> **ufw - Uncomplicated Firewall -** ufw is a default firewall configuration tool for Ubuntu which is developed to ease iptables firewall configuration, *ufw* provides a user-friendly way to create an IPv4 or IPv6 host-based firewall.

pfSense - pfSense is developed by Netgate, and it is the world's most trusted open source firewall. pfSense offers many different security services including firewall, traffic rule, VPN tunnelling,etc. In addition, pfSense SNF also provides functionalities to capture the traffic coming in and out the server for further anomaly analytics , and provides L7 deep packet inspection. Security study case in eHealth UC is built with pfSense on the top of the infrastructure to secure the system (D7.2 - eHealth UC prototyping) and perform basic packet monitoring.

Encryption

OpenSSL [28] contains cryptographic libraries providing functions to generate keys and certificates, distribute keys, functions for encryption and decryption. OpenSSL contains an open-source implementation of the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. Applications can invoke OpenSSL functions for its own data encryption purposes.

SSL/TLS channel encryption [29], OpenVPN implements SSL/TLS protocol to provide flexible VPN solutions for businesses to secure all data communications and extend private network services from site to site or connect one endpoint to a private network while maintaining security. OpenVPN is selected to secure the communication channel, D7.2 shows a demonstration of this methodology in eHealth testbed.

Vault [30] - Secure, store and tightly control access to tokens, passwords, certificates, encryption keys for protecting secrets and other sensitive data using a UI, CLI, or HTTP API. Vault also provides Secret Engine/ Encryption as a service for developer not to worry about the implementation of the cryptography feature, only need to call a service for encryption and decryption, also apply for data at rest. As Vault manages secrets (tokens, passwords, certificates, keys, confidential data) based on policies, any user of the API needs to authenticate and only sees the secrets for which he is authorised. Vault encrypts data using 256-bit AES with GCM and store data in various backends (files, Amazon DynamoDB, Consul, etcd, etc.) for extra security reason. Integrating Vault with the Federal Information Processing Standard FIPS 140-2 certified HSM and enable the Seal Wrap feature in Vault will wrap the secrets with an extra layer of encryption leveraging the HSM encryption and decryption. For all those capabilities, Vault is a good selection in SliceNet for managing secrets and providing encryption service for any applications or services and it will be further explored in eHealth UC.

Security Events (Threat, Intruder, Packet Anomalies) Monitoring and Detection

In this topic, RSA Netwitness and Moloch are discussed to enhance the security level in the system. Moloch alone can only capture the network traffic where SliceNet Analyser component in WP5 can use the captured data for further analytics or FCAPS component can perform rule checking on certain thresholds for further actuation plan for example, the captured traffic data shows that a source is sending packets that has reached a limit per second, FCAPS will send a request to a firewall SNF to block this source.

RSA Netwitness - developed by RSA, a family member of Dell Technologies, Netwitness is fully supported in VMware, Hyper-V and Openstack, and it is under testing in Docker and Kubernetes environments. Netwitness comes with many functionalities, including

- Endpoint - RSA Netwitness Endpoint is an endpoint detection and response tool that continuously monitors assets, e.g., laptops, desktops, servers, and VMs, to provide deep visibility into and powerful analysis of all behaviour and processes on an organization's endpoints. Endpoint data can be either analysed at its own free-standing analytics server, or integrated with RSA NetWitness Logs & Packets for visibility across the infrastructure.
- Logs and Packet - this part collects and analyses logs, network packets, NetFlow, and endpoint data. At the time of collection, RSA NetWitness Logs and Packets uses Capture Time Data Enrichment to inspect every piece of data collected for threat indicators. Two use cases are covered in this part: i) SIEM with log and NetFlow collection, correlation, archiving, incident management, compliance and security reporting; and ii) Network forensics and monitoring with full packet capture, correlation, Big Data analytics, deep dive investigations, incident management and response.
- UEBA Machine Learning (ML) - this feature is a purpose-built, big data-driven, user and entity behaviour analytics solution. By leveraging unsupervised ML algorithms, across a large breadth of use cases, RSA NetWitness UEBA provides comprehensive detection for unknown threats based on behaviour, without the need for analyst tuning. It can also augment the existing security system to provide rapid detection and actionable insights at every step of the attack lifecycle. RSA NetWitness UEBA is core to the RSA NetWitness Platform to help with full attack investigation lifecycle and breach resolution. The use cases this RSA part covers includes Insider threat, Brute force, Account takeover, Compromised account, Privilege account abuse and misuse, Elevated privileges, Snooping user, Data exfiltration, Abnormal system access, Lateral Movement, Malware activity and Suspicious behaviours. Details of the use cases can be found in [31].

  Event Stream Analysis (ESA) - this part provides two main services: i) ESA Correlation service that provides near real-time correlation and complex event (a series of events that when occurring in a particular order create a particular result) processing of log and packet meta; and ii) ESA Analytics service which is used for Automated Threat Detection.

RSA Netwitness (NW) has been deployed in eHealth testbed with core components: NW Server, NW Endpoint, NW ESA, NW UEBA, NW Packet. The tool can capture traffic from different sources and analyse the behaviour, RSA Netwitness will coordinate with SliceNet components, e.g., FCAPS, to detect and mitigate detected attacks or anomalies.

Moloch [32]is a large scale, open source, indexed packet capture and search system. Moloch captures network traffic in standard PCAP format with fast indexed access and fast analysis (time reduction) for analysing suspected incidents. Moloch also exposes APIs that allow PCAP data and JSON-formatted session data to be downloaded directly. Moloch comes with three components: i) **capture** that monitors network traffic, writes PCAP formatted files to disk,

parses the captured packets, and sends metadata (SPI data) to elasticsearch; ii) **viewer** that runs per capture node and handles the web interface and transfer of PCAP files; and iii) **elasticsearch** that provides search database technology powering Moloch. Moloch can be deployed in SliceNet system as a network traffic monitoring system for further analysis that might be required in FCAPS Engine or Analyzer component.

Skydive [23]FCAPS Security can leverage Skydive for anomaly detection. Skydive data can be used to validate that security rules in the network policy are working as expected. To do the anomaly detection, machine learning models would be trained with data collected by Skydive exemplifying what the network looks like when there is no threat and when it is under various threats. The models would then be applied at run time to identify the threats.

The list below shows some examples:

- Outbound suspicious access
    - Skydive would collect data on outbound access in training phase and at runtime it would be used to detect unusual outbound access.
- Data leakage detection
    - Skydive would collect data on the egress services in the training phase and at runtime it would be used to detect egress services with higher than typical outbound volumes.
- Reconnaissance/port scan attack detection
    - Skydive would collect data on services and their interfaces in the training phase and at runtime used to detect services with higher than typical number of connection requests for different IP ports.
- Insider threat detection
    - Skydive would collect data on how services interact within network cluster in training phase and at runtime it would be used to detect unusual interactions between services.

5.5 Monitoring sensors / Endpoints

This section describes some or the possible sensors / endpoints studied during the project. The list is not exhaustive and FCAPS framework can be adapted to be fed by other potential endpoints.

5.5.1 Flow Monitoring Agent Sensor

Traditional network flow monitoring sensors such as IPFIX, NetFlow and sFlow are primarily designed for pure IP networks. Therefore, they do not fit 5G networks where traffic changes its shape dynamically across the different interfaces of the data path, which is built upon an overlay structure featured with encapsulation for multi-tenancy and user mobility support. To address this, SliceNet has designed and prototyped the Flow Monitoring Agent (FMA), a network probe that is able to dissect the 5G traffic in order to understand the inner traffic inside of the overlay network and thus able to provide network metrics not only over traditional IP flows but also over tenant flows and also over 5G users flows. This probe provides key

information, together with the network flow metrics, that allows associating flows with different shapes detected in other parts of the network with these ones to set up a direct relationship between flows with different autonomy. Another differentiating point of this network probe is that it provides a complete set of metadata associated to each of the flows being reported. It is noted that a preliminary version of FMA was designed and prototyped in the 5G-PPP Phase 1 project SELFNET and further enhancements have been made in SliceNet to support slice monitoring. In particular, the metadata set has been aligned with the data model related to the slice definition at network flows level, thereby allowing producing slice-level metrics when different metrics associated to the network flows belonging to the same slice are aggregated together using the information provided by this probe. FMA is also in charge of dealing with the network flow discovery and inventory in order to allow the management of the life cycle associated to the network flows. The software architecture of FMA is shown in Figure 31. As illustrated, FMA is directly hooked into the different network interface cards (NICs) available in a given computer and it calculates metrics over the network flows that are passing such NICs by inspecting the new metadata available. As a result, FMA provides periodic reporting of both network flow metrics and network flow metadata. This reporting interface is currently implemented using both RabbitMQ and Kafka as a way to publish the information out of the probe.

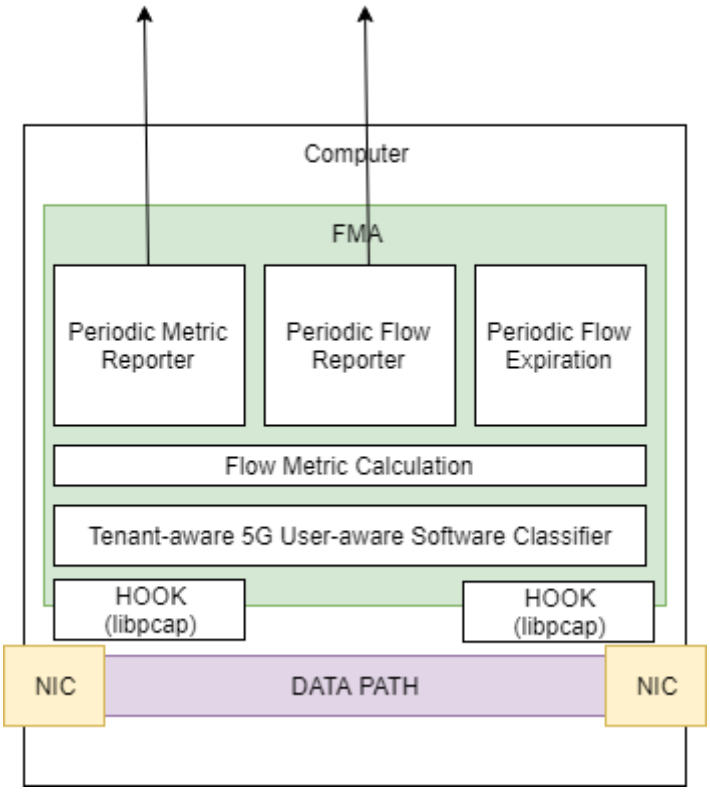

Figure 31: Flow Monitoring Agent

5.5.2 Skydive as network sensor

Overview

Skydive is an open source real-time network topology and protocols analyzer. It aims to provide a comprehensive way of understanding what is happening in the network infrastructure. Skydive agents act as data collectors employing efficient mechanisms to control the granularity of data collected and collection intrusiveness, in terms of CPU, memory and network overheads. These mechanisms allow for extra flexibility in capturing network topology, interface metrics, and network flows data, as compared to other existing methods. At the same time, they enable capturing rich sets of metadata attributes from network entities, both in the physical and in the virtual domains.

Data from Agents is reported to a centralized server component known as the Skydive Analyzer. The Analyzer is responsible for persistence, UI, and analytics of the data collected by the agents. The collected data is stored in a time series database, elasticsearch, and made available through its API and GUI; a Grafana plugin is also available. The elasticsearch database can be exported to SliceNet's Data Lake for analysis by other SliceNet components.

Skydive architecture allows for capturing information in various locations of the infrastructure, both horizontally (network entities, e.g. interfaces, switches etc., on the same layer of the stack) and vertically (network entities on multiple layers of the stack, e.g. application layer, virtual networking layer, physical networking layer, etc.). Having data collected with flexible granularity on the one hand and with high redundancy on the other allows for correlating information between locations and layers and to use various algorithms to produce insights.

Skydive is Software Defined Network (SDN) agnostic but provides various SDN drivers in order to enhance the topology and flows information, so it can be easily configured to accommodate to different SliceNet infrastructure deployments, i.e. OpenStack, Kubernetes, etc.. Skydive even supports SDN implements of virtual network with nested environment, e.g. containers running on top of an OpenStack deployment.

Skydive can be leveraged as a network sensor to provide feedback from network resources to the SliceNet Traffic and Topology Monitors to be used for FCAPS Configuration Management, Performance Management, Fault Management, and Security.

FCAPS Configuration Management with Skydive Topology Support

FCAPS Configuration can use Skydive to better understand the network topology. Skydive exposes the network topology, connectivity between network nodes, and metrics related to these network nodes and edges. Management can also start traffic capture allowing it to monitor metrics for specific protocols or according to topology information. Insights can be drawn on whether the network is actually acting according to network policies, i.e. whether slices are isolated from each other. It can even be instructed to listen for interfaces that currently do not exist but will come online in the future, e.g. new slice.

Many topology probes are available. These probes extract topological information from the host about entities residing on the host:

- Docker (docker)
- Ethtool (ethtool)
- LibVirt (libvirt)

- LLDP (lldp)
- Lxd (lxd)
- NetLINK (netlink)
- NetNS (netns)
- Neutron (neutron)
- OVSDB (ovsdb)
- Opencontrail (opencontrail)
- runC (runc)
- Socket Information (socketinfo)
- VPP (vpp)
- Kubernetes

FCAPS Performance Management with Skydive Flow Capture

FCAPS Performance management can use Skydive to gain insights into network traffic, i.e. the flow of data and bottlenecks in the network. Skydive keeps track of packets captured in flow tables. It allows Skydive to keep metrics for each flow. At a given frequency or when the flow expires flows are forwarded from agents to analysers and then to the datastore. Each time a new flow is received by the analyser the flow is enhanced with topology information like where it has been captured, where it originates from, where the packet is going to.

Flow probes capture packets and fill agent flow tables. There are different ways to capture packets like sFlow, afpacket, PCAP, etc. Flows can be captured at the level of a single interface across multiple interfaces.

Below is the list of the currently supported Flow capture probes:

- AFPACKET: MMap'd AF_PACKET socket (default).
- PCAP: Packet Capture library based.
- PCAP Socket: open a TCP port accepting PCAP file format. Useful to inject already captured traffic to Skydive.
- sFlow: implement a sFlow agent. It opens a UDP port reading `sFlow` frames. Useful to send `sFlow` traffic from external resources to Skydive.
- eBPF: in Kernel lightweight capture solution.
- OvsMirror: leverages Open vSwitch port mirroring.
- DPDK

FCAPS Fault Management with Skydive Alerts and Packet Injection

Fault Management can use Skydive to identify faults in the network. Skydive allows the user to create alerts, based on queries on both topology graph and flows. Thus, Fault Management can control and set Skydive alerts according to what is expected. Skydive keeps track of each modification allowing for troubleshooting past events.

Skydive also provides a Packet Generator/Injector feature. The goal of the Packet Generator is to forge packets and to inject them into an interface of the topology. Using this feature with the Flow capture would allow Fault Management to verify if the traffic is forwarded as expected.

5.5.3 Open Source Mano (OSM) as endpoint

Open Source Mano (OSM) is an ETSI-hosted initiative developing an Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV. Two of the key components of the ETSI NFV architectural framework are the NFV Orchestrator and VNF Manager, known as NFV MANO. Additional layers, such as service orchestration are also required for operators to enable true NFV services. ETSI OSM provides an opportunity to capitalize on the synergy between standardization and open source approaches by accessing a greater and more diverse set of contributors and developers than would normally be possible. In OSM release FOUR and FIVE monitoring capabilities have been extended. On-demand and (in an upcoming point release) descriptor-driven setting of alarms and metrics are now much simpler and convenient to configure and consume. Likewise, the support of push notifications and configuration of reactive policies, via the new Policy Manager, opens the door to closed-loop operations [24]

The following Figure 32 is a sketch of the fault and performance management modules available in OSM:

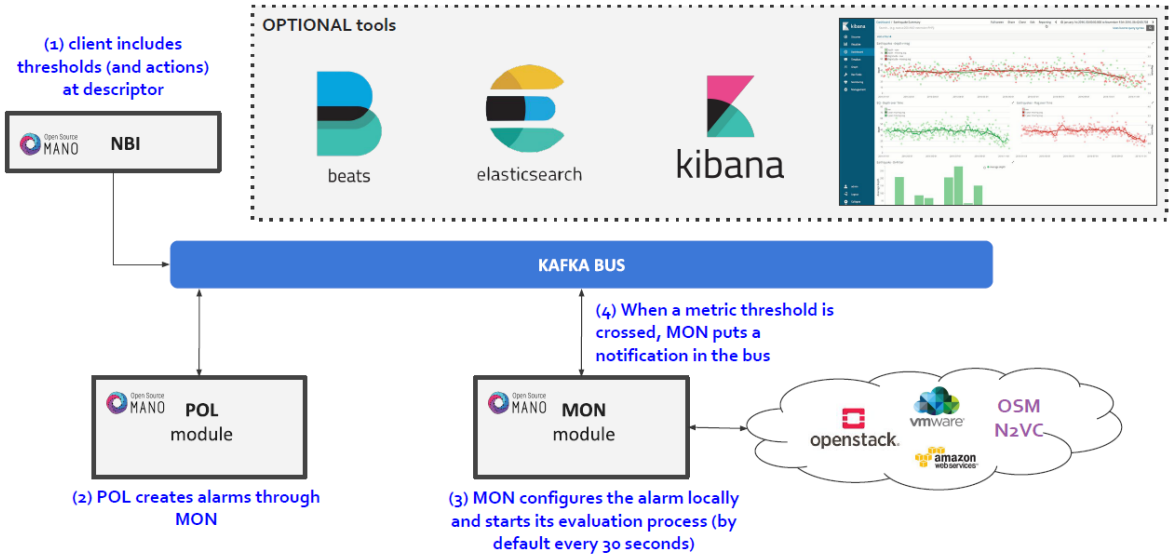

Figure 32: OSM Fault & Performance Management

OSM is composed by several modules (NBI, POL, MON) all connected to the Kafka bus. A set of optional tools can be connected to the kafka bus to aggregate, store and visualize the data published on kafka bus. FM functionality can be configured via specific instructions inserted into the descriptor of the virtual function and actuated at instantiation phase. By these instructions OSM is instructed to produce alarms every time the event described in the

subscription happen. The subscribed events will produce alarms on a specific topic (ALARM) on the Kafka bus. Another topic will be similarly populated with monitoring data (MON).

5.6 FMM Prototype

The prototype demonstrates the concept of the SliceNet Management Architecture for the Fault Management (FM) functionality in the single domain scenario.

The Prototype serves wider circumstances, e.g. in connection to some vertical use case, providing FM functionality and it is expected to be integrated with other SliceNet components and subsystems at a later stage of the project.

The Prototype is focused on Open Source Mano (OSM) end point, however it could be easily adapted to support different type of end points / sensors.

Figure 33 is a general sketch of the FM prototype in a scenario where two slides are defined on the same OSM orchestrated network:

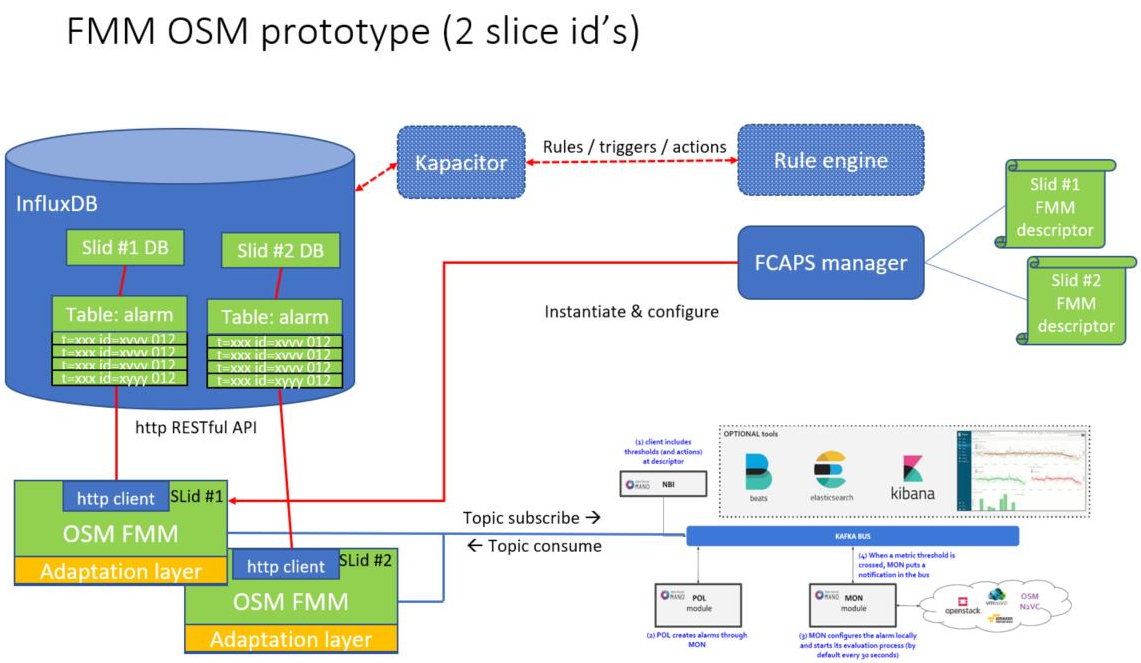

Figure 33: FMM for OSM prototype

The purpose of the prototype in this phase is to collect alarms generated by OSM and published on Kafka bus, and to aggregate and permanently store them into a database (InfuxDB) in a format that can be consumed and digested by other modules. Some examples are modules implementing rule enforcers or cognition algorithm, able to generate actions from the series of alarm data in the database.

5.6.1 OSM_FM Simulator

At this stage of the SliceNet design OSM has not yet been configured to deploy virtual networks and to generate alarms and monitoring data. This will happen in the coming months and is part of WP7.

In the meantime an OSM_FM simulator has been developed to populate Kafka bus with alarm data.

OSM_FM Simulator replaces OSM and connects directly to Kafka bus to produce random alarm data set. The following is an example of the data structure of the generated alarms:

```
alarm template = {
    'schema_version': '1.0',
    'schema_type': 'notify_alarm',
    'notify_details':
    {
        'alarm_uuid': 'XXX',
        'description': 'my_alarm',
        'resource_id': 'XXX',
        'severity': 'XXX',
        'status': 'my_status',
        'operation': 'GE',
        'threshold_value': '1.5',
        'metric_name': 'XXX',
        'ns_id': 'XXX',
        'vnf_member_index': 9999,
        'vdu_name': 'my_vdu',
        'start_date': 'XXX',
        'update_date': '',
        'cancel_date': ''
    }
}
```

Fields marked XXX are those that are variable.

An alarm event is identified by the alarm uuid and each alarm will be initiated, optionally updated, and optionally ceased in a sequence like the following (only some field are shown):

```
Notify { alarm uuid: 12345, start date: 12/05/2019 23:08:21, update date: -, cancel date:- }
/alarm start. This is always present


Notify { alarm uuid: 12345, start date: 12/05/2019 23:08:21, update date: 12/05/2019 23:13:55,
cancel date:- }  /alarm update. This is optional, it could happen because something is changed
in the alarm condition (e.g. severity level) or because the element under alarm has lost
connection and the alarm situation needs to be updated.


Notify { alarm uuid: 12345, start date: 12/05/2019 23:08:21, update date: -, cancel
date:12/05/2019 23:28:00 }  /alarm ceased. This is generated when the alarm condition is
canceled.
```

OSM_FM Simulator shall be able to generate a consistent sequence of alarm start, alarm ceased (optionally some alarm update). Examples of complete alarm data generated by simulator:

```
--------------------1 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'1c26ffa8-ccce-240c-81c1-e7ff6efacd5b', 'description': 'my_alarm', 'resource_id': 'bc991c53-
```

```
1484-f407-1f0e-4b4a886c3a30', 'severity': 'warning', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'storage', 'ns_id': '3908ec20-a2b3-41ea-8ccf-
2422d8c3a3a1', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.858863'}}
--------------------2 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'a5c2948b-eed6-9fbb-a8a8-2a070c425b3f', 'description': 'my_alarm', 'resource_id': '3acdf616-
55d9-8c6e-3459-294ef273b44c', 'severity': 'minor', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'compute', 'ns_id': 'a92298fb-00f1-48a4-b4ed-
bcad8335cbe6', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.859866'}}
--------------------3 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'3047c198-240d-84d6-2daf-9ac7f5faf207', 'description': 'my_alarm', 'resource_id': 'a0569d76-
5ebc-64cb-58d5-48aae4921bf7', 'severity': 'warning', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'compute', 'ns_id': '688d5ab3-eb21-4604-afc8-
d7bbbe1bc8f4', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.862371'}}
--------------------4 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'1c26ffa8-ccce-240c-81c1-e7ff6efacd5b', 'description': 'my_alarm', 'resource_id': 'bc991c53-
1484-f407-1f0e-4b4a886c3a30', 'severity': 'warning', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'storage', 'ns_id': '3908ec20-a2b3-41ea-8ccf-
2422d8c3a3a1', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.858863', 'update_date': '2019-05-14 10:28:40.867386'}}
--------------------5 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'a5c2948b-eed6-9fbb-a8a8-2a070c425b3f', 'description': 'my_alarm', 'resource_id': '3acdf616-
55d9-8c6e-3459-294ef273b44c', 'severity': 'minor', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'compute', 'ns_id': 'a92298fb-00f1-48a4-b4ed-
bcad8335cbe6', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.859866', 'update_date': '2019-05-14 10:28:40.868388'}}
--------------------6 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'3047c198-240d-84d6-2daf-9ac7f5faf207', 'description': 'my_alarm', 'resource_id': 'a0569d76-
5ebc-64cb-58d5-48aae4921bf7', 'severity': 'warning', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'compute', 'ns_id': '688d5ab3-eb21-4604-afc8-
d7bbbe1bc8f4', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.862371', 'update_date': '2019-05-14 10:28:40.868888'}}
--------------------7 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'1c26ffa8-ccce-240c-81c1-e7ff6efacd5b', 'description': 'my_alarm', 'resource_id': 'bc991c53-
1484-f407-1f0e-4b4a886c3a30', 'severity': 'warning', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'storage', 'ns_id': '3908ec20-a2b3-41ea-8ccf-
2422d8c3a3a1', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.858863', 'cancel_date': '2019-05-14 10:28:40.869390'}}
--------------------8 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'a5c2948b-eed6-9fbb-a8a8-2a070c425b3f', 'description': 'my_alarm', 'resource_id': '3acdf616-
55d9-8c6e-3459-294ef273b44c', 'severity': 'minor', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'compute', 'ns_id': 'a92298fb-00f1-48a4-b4ed-
bcad8335cbe6', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.859866', 'cancel_date': '2019-05-14 10:28:40.869891'}}
--------------------9 PRODUCED--------------------
{'schema_version': '1.0', 'schema_type': 'notify_alarm', 'notify_details': {'alarm_uuid':
'3047c198-240d-84d6-2daf-9ac7f5faf207', 'description': 'my_alarm', 'resource_id': 'a0569d76-
5ebc-64cb-58d5-48aae4921bf7', 'severity': 'warning', 'status': 'my_status', 'operation': 'GE',
'threshold_value': '1.5', 'metric_name': 'compute', 'ns_id': '688d5ab3-eb21-4604-afc8-
d7bbbe1bc8f4', 'vnf_member_index': 9999, 'vdu_name': 'my_vdu', 'start_date': '2019-05-14
10:28:40.862371', 'cancel_date': '2019-05-14 10:28:40.870393'}}
```


5.6.2 OSM FMM Instance

For each slice who needs to get alarms from OSM an OSM FMM instance is instantiated in a Docker. The OSM FMM instance is launched and configured by FCAPS Application Manager. FCAPS manager passes the needed configuration to each OSM FMM instance containing data such as: OSM endpoint address, Slice Id, corresponding NSIid in OSM, etc. Each Telegraf instance is responsible to consume alarm data for the relevant Network Slice and to store them into InfluxDB (Figure 34).

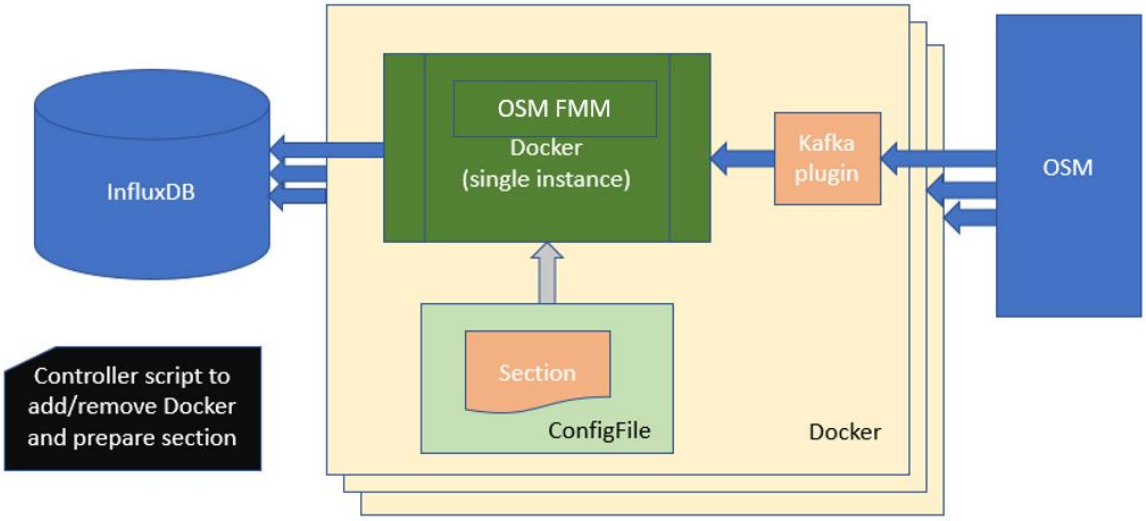

Figure 34: OSM FMM

In the example data produced in Kafka bus by OSM-FM simulator described in section 5.6.1 are normalized, flatten and mapped in the a general alarm agnostic format described in 5.3.The following is a snapshot of the InfluxDB data base while the prototype is running.

```
 > select * from slice_ciriaco
name: slice_ciriaco
-------------------
time           alarm_uuid            cancel_date           description           geo_agg_code
               metric_name           ns_id          originating_subsystem        resource_id
       severity              start_date


1559154712237313365           59885afc-bb61-a9cd-649d-da6eb49c83dc
       my_alarm              lat-long              compute        ns_ciriaco             OSM
               1c26ffa8-ccce-240c-81c1-e7ff6efacd5b         indeterminate         2019-05-29
19:28:56.818125


1559154712264694793           59885afc-bb61-a9cd-649d-da6eb49c83dc         2019-05-29
19:28:56.825143               my_alarm              lat-long              compute
       ns_ciriaco            OSM           1c26ffa8-ccce-240c-81c1-e7ff6efacd5b
       indeterminate         2019-05-29 19:28:56.818125


> show field keys on telegraf_db from slice_ciriaco
name: slice_ciriaco
-------------------
fieldKey        fieldType
alarm_uuid      string
cancel_date     string
description     string
start_date      string


> show tag keys on telegraf_db from slice_ciriaco
name: slice_ciriaco
-------------------
tagKey
geo_agg_code
metric_name
ns_id
```

```
originating_subsystem
resource_id
severity
```


6 Conclusions

The FCAPS framework has been designed to cater as an enabling component in the context of single domain that can allow an NSP to federate its operation phase offerings that can be further exploited in higher level (DSP and Vertical) workflows. The approach is currently being integrated in Use Case scenarios and it is expected that useful feedback will be collected to allow for further improvements as well as consolidation of aspects. It is worth mentioning that several aspects have been tested with emulated sources of information. Therefore, alignment with overall orchestration and service delivery workflows is necessary and expected to take place in the following period.

FCAPS components are under integration in the context of the current use cases scenarios. The attempt aims to address as much aspects as possible so that a hand on feedback can be collected in the following period to drive optimizations and adaptations towards the final phase of the project. Once adequacy of the FCAPS workflows is verified, other aspects relating to integration with One Stop API and Orchestration sub-plane will be actively evaluated and any additional iterations will be performed in the context of integration tasks.

As standardization for FCAPS management for network slices is still underway in the standardization bodies with technical and vertical service requirements defined and basic interfaces emerged, the design and prototyping of SliceNet FCAPS management framework has been informed by these ongoing and emerging standardization activities, and compliance with the principles and practices has been attempted. The overall aim of this work is to evaluate the solution in the foreseen Use Cases and consolidate certain parts that can be potentially used for contribution to the standardization activities.

7 References

[1] SliceNet Deliverable D2.2, “Overall Architecture and Interfaces Definition”, SliceNet Consortium, January 2018

[2] SliceNet Deliverable D2.3, “Control Plane System Definition, APIs and Interfaces”, SliceNet Consortium, April 2018

[3] SliceNet Deliverable D2.4, “Management Plane System Definition, APIs and Interfaces”, SliceNet Consortium, May 2018

[4] SliceNet Deliverable D4.1, “Plug & Play Control Plane for Sliced Networks”, SliceNet Consortium, September 2018.

[5] SliceNet Deliverable D4.2, “Network Slicing in 5G RAN-Core”, SliceNet Consortium, October 2018

[6] SliceNet Deliverable D4.4, “Multi-Domain Network Slicing Control and Negotiation”, SliceNet consortium, planned for February 2019.

[7] SliceNet Deliverable D6.3, “Management for the Plug & Play Control Plane”, SliceNet consortium, planned for May 2019

[8] ONF TR-526, “Applying SDN Architecture to 5G Slicing”, Apr. 2016.

[9] Floodlight Project web site. Available at http://www.projectfloodlight.org/floodlight/.

[10] ODL OpenDayLight web site. Available at https://www.opendaylight.org/.

[11] ONOS web site. Available at https://onosproject.org/.

[12] RyU web site. Available at https://osrg.github.io/ryu/.

[13] Open Networking Foundation (ONF) web site. Available at https://www.opennetworking.org/.

[14] Network Configuration Protocol RFC. Available at https://tools.ietf.org/html/rfc6241.

[15] 5G PPP Phase 1 project SELFNET Deliverable D3.4 Report and Prototype Implementation of the NFV & SDN Sensors and Actuators related to the Self-Optimizing Use Case, DOI: 10.18153/SLF-671672-D3_4, available at: https://bscw.selfnet-5g.eu/pub/bscw.cgi/d74902-5/*/*/*/*/*/DOI-D3.4.html.

[16] 5G PPP Phase 1 project SELFNET D3.1: Report and Prototype Implementation of the NFV & SDN Repository, https://bscw.selfnet-5g.eu/pub/bscw.cgi/d74922-5/*/*/*/*/*/DOI-D3.1.html

[17] 3GPP TS 23.251 Multi-Operator Core Network (MOCN)

[18] 3GPP TR 23.707 Dedicated Core Networks (DECOR)

[19] 3GPP TS 23.501 SBA System Architecture for the 5G System

[20] 3GPP TS 28.530 Aspects; Management and orchestration; Concepts, use cases and requirements

[21] 3GPP TS 28.550 Management and orchestration; Performance assurance

[22] 3GPP TR 28.801 Telecommunication management;Study on management and orchestration of network slicing for next generation network

[23] Skydive, Real-time network analyzer, http://skydive.network

[24] OSM Release FOUR Wiki: https://osm.etsi.org/wikipub/index.php/OSM_Release_FOUR

[25] 5G PPP Phase 1 project Selfnet D6.2 Report and Prototypical Implementation of the NFV & SDN Application Manager, https://bscw.selfnet-5g.eu/pub/bscw.cgi/d99388-5/*/*/*/*/*/DOI-D6.2.html

[26] 5G PPP Phase 1 project Selfnet D5.1 Report and Software Libraries to deal with the Tactical Autonomic Language, https://bscw.selfnet-5g.eu/pub/bscw.cgi/d99388-5/*/*/*/*/*/DOI-D5.1.html
[27] Security Aspects of Network Capabilities Exposure in 5G https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180921_NGMN-NCEsec_white_paper_v1.0.pdf
[28] OpenSSL - https://www.openssl.org/
[29] OpenVPN, https://openvpn.net/
[30] Vault, https://www.hashicorp.com/products/vault/
[31] RSA Netwitness UEBA Use Cases, https://www.rsa.com/content/dam/en/use-case/rsa-netwitness-ueba-use-cases.pdf
[32] Moloch, https://github.com/aol/moloch

Annex A

A.1 Complete TAL Policy Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:tal  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
          xmlns:ns0='tal.cse.com'
          xsi:schemaLocation='tal.cse.com tal.xsd' OID="">
    <ns0:reaction>
        <ns0:diagnosis>
            <ns0:symptom OID="ran_failure">
                <ns0:analysis>
                    <ns0:aggregation>
                        <ns0:aggregationItem>
                            <ns0:aggregationData>
                                <ns0:sensor OID="meo-dataset">
                                    <ns0:output>
                                        <ns0:parameter>
                                            <ns0:name>id</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>open</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>event_count</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>correlation_rule</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>close_time</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>geo_agg_code</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>entity_class</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>entity_id</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>severity</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>problem_class</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>open_time</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>origin_subsystem</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>entity_technology</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>affected_services</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>ticket_id</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>site_owner</ns0:name>
                                        </ns0:parameter>
                                        <ns0:parameter>
                                            <ns0:name>site_services</ns0:name>
                                        </ns0:parameter>

                                    </ns0:output>
                                </ns0:sensor>
                            </ns0:aggregationData>
                            <ns0:aggregationRule>
                                <ns0:ruleItem>
                                    <ns0:sliding_window>
```



```
                                <ns0:parameter>
                                    <ns0:name>geo_agg_code</ns0:name>
                                </ns0:parameter>
                                <ns0:period duration="00:00:10"/>
                            </ns0:sliding_window>
                        </ns0:ruleItem>
                    </ns0:aggregationRule>
                </ns0:aggregationItem>
            </ns0:aggregation>
            <ns0:analysisRule>
                <ns0:plugin OID="ML-Module">
                    <ns0:output>
                        <ns0:parameter>
                            <ns0:name>prediction.probability</ns0:name>
                        </ns0:parameter>
                        <ns0:parameter>
                            <ns0:name>prediction.time</ns0:name>
                        </ns0:parameter>
                        <ns0:parameter>
                            <ns0:name>context_data.geo_agg_code</ns0:name>
                        </ns0:parameter>
                    </ns0:output>
                </ns0:plugin>
            </ns0:analysisRule>
        </ns0:analysis>
    </ns0:symptom>
    <ns0:causes>
        <ns0:cause  causeOID="onTimePrediction">
            <ns0:threshold comparison="moreEqual">
                <ns0:level>
                    <ns0:parameter>
                        <ns0:OID>time</ns0:OID>
                        <ns0:name>prediction.time</ns0:name>
                        <ns0:value>00:00:20</ns0:value>
                    </ns0:parameter>
                </ns0:level>
            </ns0:threshold>
        </ns0:cause>
        <ns0:cause  causeOID="latePrediction">
            <ns0:threshold comparison="lessOrEqual">
                <ns0:level>
                    <ns0:parameter>
                        <ns0:OID>time</ns0:OID>
                        <ns0:name>prediction.time</ns0:name>
                        <ns0:value>00:00:20</ns0:value>
                    </ns0:parameter>
                </ns0:level>
            </ns0:threshold>
        </ns0:cause>
    </ns0:causes>
</ns0:diagnosis>
<ns0:tactic>
    <ns0:tacticItem>
        <ns0:cause causeOID="onTimePrediction"/>
        <ns0:action>
            <ns0:actionOption  operation="IncreaseBW">
                <ns0:actuator OID="CP.QoS">
                    <ns0:configuration>
                        <ns0:parameter>
                            <ns0:name>eNBId</ns0:name>
                            <ns0:value>{context_data.geo_agg_code}</ns0:value>
                        </ns0:parameter>
                        <ns0:parameter>
                            <ns0:name>increase</ns0:name>
                            <ns0:value>0.25</ns0:value>
                        </ns0:parameter>
                    </ns0:configuration>
                </ns0:actuator>
            </ns0:actionOption>
        </ns0:action>
    </ns0:tacticItem>
    <ns0:tacticItem>
        <ns0:cause causeOID="latePrediction"/>
        <ns0:action>
            <ns0:actionOption typeOfOption="add" operation="export">
                <ns0:resourceAction OID="NSP_Alarm">
                    <ns0:details>
```

```
                                <ns0:parameter>
                                    <ns0:name>NSP_ID</ns0:name>
                                    <ns0:value>{NSP_ID}</ns0:value>
                                </ns0:parameter>
                                <ns0:parameter>
                                    <ns0:name>NSS_ID</ns0:name>
                                    <ns0:value>{NSS_ID}</ns0:value>
                                </ns0:parameter>
                                <ns0:parameter>
                                    <ns0:name>AlarmType</ns0:name>
                                    <ns0:value>RAN_FAILURE</ns0:value>
                                </ns0:parameter>
                            </ns0:details>
                        </ns0:resourceAction>
                    </ns0:actionOption>
                </ns0:action>
            </ns0:tacticItem>
        </ns0:tactic>
    </ns0:reaction>
</ns0:tal>
```

