A white and blue logo

Description automatically generated with medium confidence Text

Description automatically generated

**XGVela Developer Guide**

##### Document Version: 1.0

##### Published Date: March 15, 2021



XGVela | Mavenir

**Copyright & Notice**

Copyright 2020 Mavenir

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

**Contents**

1. [Revision History 10](#_bookmark0)
2. [Abbreviations 11](#_bookmark1)
3. [References 12](#_bookmark2)
4. [About this Publication 13](#_bookmark3)
   1. [XGVela Overview 13](#_bookmark4)
   2. [XGVela Features 13](#_bookmark5)
5. [XGVela Overview 14](#_bookmark6)
6. [XGVela Architecture 15](#_bookmark8)
   1. [CNF Deployment Model 15](#_bookmark9)
   2. [CNF Management Model 16](#_bookmark11)
   3. [Network Resource Model 16](#_bookmark13)
      1. [Definitions 16](#_bookmark14)
      2. [Class Diagrams 18](#_bookmark24)
      3. [Class Definitions 21](#_bookmark37)
      4. [Attribute Definitions 25](#_bookmark58)
7. [Packaging Network Functions 32](#_bookmark60)
   1. [Metadata Packaging 32](#_bookmark61)
   2. [Configuration Packaging 33](#_bookmark62)
   3. [CNF Event Definition Packaging 33](#_bookmark63)
   4. [CNF Dashboard Injection 34](#_bookmark64)
   5. [Defining Microservice Secret Definitions 34](#_bookmark65)
   6. [Variable Substitution 35](#_bookmark66)
8. [XGVela Release Artifacts 37](#_bookmark67)
9. [CIM 38](#_bookmark69)
   1. [CIM Beta Version Enhancements 40](#_bookmark71)
   2. [Additional CIM Settings 41](#_bookmark72)
   3. [XGVela Registration 42](#_bookmark74)
   4. [Shutdown 43](#_bookmark75)
   5. [CIM Liveliness and Readiness Settings 43](#_bookmark76)
10. [API Gateway 45](#_bookmark77)
    1. [Get NFService By Name 45](#_bookmark78)
    2. [Get NFServiceInstance By ID 45](#_bookmark79)
11. [TMaaS 47](#_bookmark80)
    1. [Kubec 48](#_bookmark82)
    2. [Topo 48](#_bookmark83)
       1. [Kubernetes Mapping 48](#_bookmark84)
       2. [States and Transitions 51](#_bookmark93)
    3. [TMaaS Chart Specification 55](#_bookmark98)
    4. [TMaaS Events 56](#_bookmark99)
    5. [TMaaS Interfaces 59](#_bookmark102)
    6. [Operation Specification Registration 59](#_bookmark103)
12. [LMaaS 61](#_bookmark105)
    1. [Packaging Model for LMaaS 61](#_bookmark107)
    2. [LMaaS Events 63](#_bookmark109)
    3. [LMaaS Metrics 63](#_bookmark111)
    4. [LMaaS Interface 64](#_bookmark113)
13. [CMaaS 65](#_bookmark114)
    1. [Packaging Model for Configuration Management 66](#_bookmark116)
       1. [CIM Configurations 66](#_bookmark117)
    2. [Update Policies 67](#_bookmark118)
    3. [Managing Dependencies 68](#_bookmark119)
    4. [Application Upgrade 68](#_bookmark120)
    5. [Day 1 Configuration 69](#_bookmark121)
       1. [Full Config On Restart 69](#_bookmark122)
    6. [CMaaS Interfaces 70](#_bookmark123)
       1. [updateConfig 70](#_bookmark124)
       2. [Load Full Config 71](#_bookmark126)
    7. [NBI Integration 72](#_bookmark128)
       1. [NF Onboarding and Configuration Management 72](#_bookmark129)
       2. [CMaaS NBI Details Exchange 73](#_bookmark130)
       3. [Configuration Update (Day-1/Day-2) 73](#_bookmark131)
    8. [CMaaS Metrics 75](#_bookmark132)
    9. [CMaaS Events 77](#_bookmark134)
14. [MMaaS 80](#_bookmark136)
    1. [Packaging Model for MMaaS 80](#_bookmark138)
    2. [Enabling Scraping of Metrics 81](#_bookmark139)
    3. [Custom Application Metrics 83](#_bookmark141)
    4. [Defining Metrics 84](#_bookmark143)
    5. [Sample Metrics 86](#_bookmark145)
       1. [Metrics Exposed by DU Microservices 86](#_bookmark146)
       2. [VES Gateway Measurement Event Mapping for DU 86](#_bookmark147)
       3. [Generated Measurement VES Event for DU 87](#_bookmark148)
    6. [Prometheus Configuration 88](#_bookmark149)
    7. [Metrics Exposition Format 88](#_bookmark150)
    8. [Metric Types 89](#_bookmark151)
    9. [Clock-Aligned Metrics 91](#_bookmark152)
    10. [Metrics with Prometheus Operator 91](#_bookmark153)
    11. [MMaaS Deployment Models 92](#_bookmark154)
    12. [Prerequisites for Helm Charts 94](#_bookmark157)
        1. [Non Clock Aligned Metrics with XGVela Prometheus 94](#_bookmark158)
        2. [Clock Aligned Metrics with XGVela Prometheus 95](#_bookmark160)
        3. [Prometheus Operator and Non Clock Aligned Metrics 96](#_bookmark162)
        4. [Prometheus Operator and Clock Aligned Metrics 97](#_bookmark164)
    13. [Multiple Metrics Remote Storage 97](#_bookmark166)
15. [VES Gateway 99](#_bookmark167)
    1. [Packaging Model for VES Gateway 99](#_bookmark169)
    2. [Northbound Interface Configuration 100](#_bookmark170)
       1. [Dynamic Configuration Support for VES Gateway 102](#_bookmark171)
    3. [Discovery of XGVela Management Connection Details - Management IP and Ports 102](#_bookmark172)
    4. [VES Gateway Events 102](#_bookmark173)
    5. [VES - General Introduction and Events 103](#_bookmark175)
       1. [VES - Common Event Header 105](#_bookmark177)
    6. [VES Heartbeat Domain 107](#_bookmark179)
       1. [Common Event Header Fields for Heartbeat Domain 109](#_bookmark181)
    7. [VES Measurement Domain 110](#_bookmark183)
       1. [Common Event Header Fields for the Measurement Domain 113](#_bookmark185)
    8. [VES Fault Domain 114](#_bookmark187)
       1. [Common Event Header Fields for Fault Domain 115](#_bookmark189)
    9. [VES TCA Domain 117](#_bookmark191)
       1. [Common Event Header Fields for TCA Domain 118](#_bookmark193)
    10. [VES Notifications Domain 120](#_bookmark195)
        1. [Common Event Header Fields for Notifications Domain 121](#_bookmark197)
16. [FMaaS 124](#_bookmark199)
    1. [Packaging Model for Fault Management 125](#_bookmark201)
    2. [Defining Events 125](#_bookmark202)
    3. [Defining TCAs 126](#_bookmark204)
    4. [System TCAs 127](#_bookmark205)
    5. [Event Subscriptions 128](#_bookmark206)
       1. [Event Subscription Interfaces 128](#_bookmark207)
    6. [FMaaS Events 130](#_bookmark211)
    7. [FMaaS Metrics 131](#_bookmark213)
    8. [FMaaS Interfaces 131](#_bookmark215)
       1. [Application Interface 131](#_bookmark216)
       2. [Management Interfaces 136](#_bookmark218)
17. [XGVela Events 144](#_bookmark228)
18. [Annexure A 152](#_bookmark232)
    1. [Fault\_XGVela-Mavenir\_FmaasConfigUpdateFailed 152](#_bookmark233)
    2. [Notification\_XGVela-Mavenir\_NFServiceInstanceCreated 153](#_bookmark234)
    3. [Notification\_XGVela-Mavenir\_NFServiceInstanceDeleted 154](#_bookmark235)
    4. [Notification\_XGVela-Mavenir\_NetworkFunctionStateChanged 156](#_bookmark236)
    5. [Mvfs\_xgvela1\_Measurements 157](#_bookmark237)
    6. [heartbeat\_xgvela1 158](#_bookmark238)
    7. [NFMgmtIntfReady 158](#_bookmark239)
19. [Annexure B 160](#_bookmark240)
    1. [NFService By Name 160](#_bookmark241)
    2. [NFServiceInstance by ID 166](#_bookmark242)
20. [Annexure C 168](#_bookmark243)
21. [Annexure D 171](#_bookmark244)
22. [Annexure F 172](#_bookmark245)
23. [Document Quality and Content Feedback 173](#_bookmark246)

**List of Tables**

#### [Table 1: ManagedElement Attributes 22](#_bookmark41)

#### [Table 2: ManagedFunction Attributes 23](#_bookmark47)

#### [Table 3: ManagedNFService Attributes 24](#_bookmark52)

#### [Table 4: Top Attributes 25](#_bookmark57)

#### [Table 5: Attribute Definitions 25](#_bookmark59)

#### [Table 6: XGVela Release Artifacts 37](#_bookmark68)

#### [Table 7: cim.json Attribute Descriptions 39](#_bookmark70)

#### [Table 8: cim.json Attribute Descriptions 41](#_bookmark73)

#### [Table 9: ManagedElement Mapping 49](#_bookmark86)

#### [Table 10: NetworkFunction Mapping 50](#_bookmark88)

#### [Table 11: NFService Mapping 50](#_bookmark90)

#### [Table 12: NFServiceInstance Mapping 51](#_bookmark92)

#### [Table 13: NF States 51](#_bookmark94)

#### [Table 14: State Transitions 52](#_bookmark95)

#### [Table 15: NF Service Instance States and Actions 54](#_bookmark97)

#### [Table 16: TMaaS Events 56](#_bookmark100)

#### [Table 17: TMaaS Operation Related Events 59](#_bookmark101)

#### [Table 18: Registration Operation Parameters 60](#_bookmark104)

#### [Table 19: LMaaS Parameter Details 62](#_bookmark108)

#### [Table 20: LMaaS Events 63](#_bookmark110)

#### [Table 21: LMaaS Metrics 64](#_bookmark112)

#### [Table 22: updateConfig API Parameters 70](#_bookmark125)

#### [Table 23: NF Events 72](#_bookmark127)

#### [Table 24: CMaaS Metrics 75](#_bookmark133)

#### [Table 25: CMaaS Events 77](#_bookmark135)

#### [Table 26: MMaaS Annotation Structure for Clock Aligned Metrics 83](#_bookmark140)

#### [Table 27: Metric Fields 83](#_bookmark142)

#### [Table 28: Field Descriptions 85](#_bookmark144)

#### [Table 29: MMaaS Deployment Models 93](#_bookmark155)

#### [Table 30: Number of Running Pods for MMaaS Deployment Models 94](#_bookmark156)

#### [Table 31: Prerequisites for Helm Charts for Non Clock Aligned Metrics with XGVela](#_bookmark159) [Prometheus 94](#_bookmark159)

#### [Table 32: Prerequisites for Helm Charts for Clock Aligned Metrics with XGVela](#_bookmark161) [Prometheus 95](#_bookmark161)

#### [Table 33: Prerequisites for Helm Charts for Prometheus Operator and Non Clock](#_bookmark163) [Aligned Metrics 96](#_bookmark163)

#### [Table 34: Prerequisites for Helm Charts for Prometheus Operator and Clock Aligned](#_bookmark165) [Metrics 97](#_bookmark165)

#### [Table 35: VES Gateway Events 103](#_bookmark174)

#### [Table 36: Event Datatype Field Descriptions 104](#_bookmark176)

#### [Table 37: Event Datatype Field Descriptions 105](#_bookmark178)

#### [Table 38: Heartbeat Domain Fields Descriptions 108](#_bookmark180)

#### [Table 39: Common Event Header Fields for Heartbeat Domain 110](#_bookmark182)

#### [Table 40: Measurement Domain Field Descriptions 111](#_bookmark184)

#### [Table 41: Common Event Header Fields for the Measurement Domain 113](#_bookmark186)

#### [Table 42: VES Fault Domain Fields Descriptions 115](#_bookmark188)

#### [Table 43: VES Fault Domain Common Event Header Fields Descriptions 116](#_bookmark190)

#### [Table 44: TCA Domain Fields Descriptions 117](#_bookmark192)

#### [Table 45: TCA Domain Fields Descriptions 118](#_bookmark194)

#### [Table 46: TCA Domain Fields Descriptions 120](#_bookmark196)

#### [Table 47: TCA Domain Fields Descriptions 121](#_bookmark198)

#### [Table 48: FMAaaS Domains and Severities 126](#_bookmark203)

#### [Table 49: FMaaS Events 131](#_bookmark212)

#### [Table 50: FMaaS Metrics 131](#_bookmark214)

#### [Table 51: Event Fields 132](#_bookmark217)

#### [Table 52: Notification Domain Events 144](#_bookmark229)

#### [Table 53: Fault Domain Events 148](#_bookmark230)

#### [Table 54: TCA Domain Events 150](#_bookmark231)

**List of Figures**

#### [Figure 1: XGVela Architecture 14](#_bookmark7)

#### [Figure 2: CNF Deployment Model 15](#_bookmark10)

#### [Figure 3: CNF Management Model 16](#_bookmark12)

#### [Figure 4: Relationships Between a Name Space and Participating MOs 17](#_bookmark18)

#### [Figure 5: UIM Class Diagram 18](#_bookmark26)

#### [Figure 6: Performance Measurement Control NRM Fragment 19](#_bookmark27)

#### [Figure 7: Measurement Threshold Monitoring Control Fragment 19](#_bookmark28)

#### [Figure 8: Heartbeat Notification Control Fragment 20](#_bookmark29)

#### [Figure 9: Notification Subscription Control Fragment 20](#_bookmark30)

#### [Figure 10: Inheritance Hierarchy NRM Fragment 20](#_bookmark32)

#### [Figure 11: Performance Measurement Control NRM Fragment 20](#_bookmark33)

#### [Figure 12: Measurement Threshold Monitoring NRM Fragment 20](#_bookmark34)

#### [Figure 13: Heartbeat Notification Control Fragment 21](#_bookmark35)

#### [Figure 14: Notification Subscription Fragment 21](#_bookmark36)

#### [Figure 15: TMaaS Design 47](#_bookmark81)

#### [Figure 16: NF States and State Transitions 54](#_bookmark96)

#### [Figure 17: LMaaS Design 61](#_bookmark106)

#### [Figure 18: CMaaS Design 65](#_bookmark115)

#### [Figure 19: MMaaS Design 80](#_bookmark137)

#### [Figure 20: VES Gateway Architecture 99](#_bookmark168)

#### [Figure 21: FMaaS Design 124](#_bookmark200)

# Revision History

|  |  |  |
| --- | --- | --- |
| **Release** | **Date** | **Comments** |
| 1.0 | 15/03/2021 | Initial version |

# Abbreviations

|  |  |
| --- | --- |
| Acronym | Description |
| CaaS | Container as a Service |
| CIM | Container Interface Module |
| CMaaS | Configuration Management as a Service |
| DN | Distinguished Name |
| FMaaS | Fault Management as a Service |
| IOC | Information Object Class |
| LMaaS | Log Management as a Service |
| MIB | Management Information Base |
| MMaaS | Metrics Management as a Service |
| MO | Managed Object |
| MOC | Managed Object Class |
| MOI | Managed Object Instance |
| MWP | Mavenir Webscale Platform |
| NRM | Network Resource Model |
| ONAP | Open Networking Automation Platform |
| RDN | Relative Distinguished Name |
| TMaaS | Topology Management as a Service |
| MTCIL | Mavenir Telco Cloud Integration Layer (Seed Code) |

# References

## 3GPP SPECIFICATIONS

1. 3GPP TS 28.532: Management and orchestration; Generic management services
2. 3GPP TR 28.890: Management and orchestration; Study on integration of Open Network Automation Platform (ONAP) and 3GPP management for 5G networks
3. 3GPP TS 32.102: "Telecommunication management; Architecture"
4. 3GPP TS 32.300: "Telecommunication management; Configuration Management (CM); Name convention for Managed Objects"
5. 3GPP TS 32.352: Telecommunication management; Communication Surveillance (CS) Integration Reference Point (IRP); Information Service (IS)
6. 3GPP TS 28.533: Management and orchestration; Architecture framework
7. 3GPP TS 28.620: Telecommunication management; Fixed Mobile Convergence (FMC) Federated Network Information Model (FNIM) Umbrella Information Model (UIM) (V15.1.0)

## ETSI SPECIFICATIONS

1. ETSI GS NFV-SWA 001 Network Functions Virtualization (NFV); Virtual Network Functions Architecture

## ITU SPECIFICATIONS

1. ITU-T Recommendation X.710 (1991): "Common Management Information Service Definition for CCITT Applications".
2. T-REC-X.731-199201-I!!PDF-E: State Management Function

## OTHER REFERENCES

1. VES: <https://wiki.onap.org/display/DW/5G+-+Real+Time+PM+and+High+Volume+Stream+Data+Collection>
2. VES:<https://wiki.onap.org/display/DW/VES+7.1>
3. Kubernetes:<https://kubernetes.io/>

# About this Publication

Refer to the following sections for more information about XGVela and its features:

###### [XGVela Overview](#_bookmark4)

###### [XGVela Features](#_bookmark5)

# XGVela Overview

Modern systems must be distributed and cloud-native to deliver expected levels of reliability, agility, and scale. Containers are fundamental building blocks of a cloud-native NFs or microservices. In Kubernetes, the embodiment of a modular container service is a pod. A pod is a group of containers that share resources like file systems, kernel namespaces and an IP address. To build an NF or microservice from modular containers, symbiotic groups of containers must cooperate to provide a service, not one container per service.

XGVela is a cloud-native platform and a framework for Cloud-Native NFs (CNFs).

[Figure 1: XGVela Architecture](#_bookmark7) describes the XGVela architecture and shows the interactions between XGVela and the other layers, such as PaaS, CaaS, CMS, CI/CD, and NFs:

Graphical user interface, text, application

Description automatically generated

Figure 1: XGVela Architecture

XGVela provides the following features:

* Distributed logging, tracing, telemetry, configuration interfaces, and infrastructure
* Deployable on any Kubernetes based PaaS that is instrumented on Cloud or BareMetal.
* Interface towards OSS, MANO and Analytics.

Telecom Network Functions (NF) are deployed on XGVela, that employs a pluggable service interface and infrastructure towards PaaS. This allows network functions to easily adapt and cope with different vendor PaaS environments and operator SLAs.

Following are the capabilities of CNF Framework:

* Provides a framework to develop, package, deploy, and manage network functions.
* Implements a set of management and infrastructure microservices to support and manage NFs or µServices.

# XGVela Architecture

XGVela follows an architecture where NFs are a collection of loosely coupled µServices. Helm is used as the package manager for CNF. An NF package includes the following:

* Templates
* Specifications of microservices
* Configuration YANG file and initial configuration data
* Event definitions
* Metrics and alert rules
* Dashboard

XGVela provides the following functionalities:

* TMaaS
* LMaaS
* CMaaS
* FMaaS
* MMaaS

Refer to the following sections for information about CNF deployment and management:

###### [CNF Deployment Model](#_bookmark9)

###### [CNF Management Model](#_bookmark11)

###### [Network Resource Model](#_bookmark13)

## CNF Deployment Model

All network functions are componentized, virtualized, and grouped into CNF. The management system manages functions based on the CNF view as follows:

* Each NF is disaggregated into microservices of various types.
* One or more network functions can be deployed on XGVela.
* Each XGVela instance is deployed on CaaS/PaaS and runs in its own namespace
* Each NF instance deployed on XGVela runs in its own namespace, which is derived from the XGVela namespace it is running on.

[Figure 2: CNF Deployment Model](#_bookmark10) describes the CNF deployment model:



Figure 2: CNF Deployment Model

## CNF Management Model

TMaaS defines and implements a generic ManagedObject model schema for describing the exporting the cluster and NF topology. The topology is rendered as a hierarchical structure of ManagedObjects. Topology discovery and model generation is described in discovery section.

A ManagedObject models basic properties and relationships. These are updated with actual NF, microservice resources (containers, pod, volumes, and configuration), properties and relationships upon discovery of the same using Kebernetes APIs.

TMaaS model is based on and in most part derived from 3GPP NRM (Release 16). [Figure 3: CNF Management Model](#_bookmark12) describes the CNF management model:

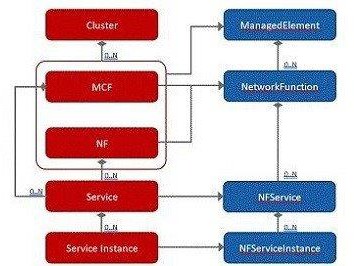


Figure 3: CNF Management Model

## Network Resource Model

Refer to the following sections:

* [Definitions](#_bookmark14)
* [Class Diagrams](#_bookmark24)
* [Class Definitions](#_bookmark37)
* [Attribute Definitions](#_bookmark58)

### Definitions

This section provides definitions of commonly used terms. Refer to the following sections:

* [Information Object Class](#_bookmark15)
* [Managed Object](#_bookmark16)
* [Management Information Base](#_bookmark17)
* [Name Space](#_bookmark19)
* [Network Resource](#_bookmark20)
* [Network Resource Model](#_bookmark21)
* [Global Root and Local Root](#_bookmark22)
* [Distinguished Name and Relative Distinguished Name](#_bookmark23)
  + - 1. Information Object Class

An Information Object Class (IOC) represents the management aspect of a network resource. It describes the information that can be passed or used in management interfaces. IOC representations are technology agnostic software objects. It has attributes that represents the various properties of the class of objects. An IOC can support operations providing network management services invocable on demand for that class of objects. An IOC may support notifications that report event occurrences relevant for that class of objects. It is modelled using the stereotype Class in the UML meta-model.

* + - 1. Managed Object

A managed object (MO) is an instance of a Managed Object Class (MOC) representing the management aspects of a network resource. Its representation is a technology specific software object. It is sometimes called MO instance (MOI). The MOC is a class of such technology specific software objects. An MOC is the same as an IOC except that the former is defined in technology specific terms and the latter is defined in technology agnostic terms. MOCs are used or defined in SS level specifications. IOCs are used or defined in IS level specifications.

* + - 1. Management Information Base

A Management Information Base (MIB) is an instance of an NRM and has some values on the defined attributes and associations specific for that instance. A MIB consists of the following:

* Name space (describing the MO containment hierarchy in the MIB through Distinguished Names)
* MOs and its attributes
* Associations between MOs.

TMN (ITU-T Recommendation X.710 [7]) defines a concept of a Management Information Tree, also known as a Naming Tree, that corresponds to the name space (containment hierarchy) portion of the MIB definition.

[Figure 4: Relationships Between a Name Space and Participating MOs](#_bookmark18) depicts the relationships between a name space and participating MOs:

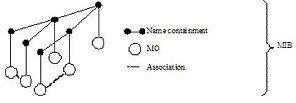


Figure 4: Relationships Between a Name Space and Participating MOs

* + - 1. Name Space

A name space is a collection of names. The IRP name convention restricts the name space to a hierarchical containment structure, including its simplest form the one-level, flat name space. Refer 3GPP TS

32.300 for details.

All MOs in a MIB are included in the corresponding name space and the MIB or name space supports a strict hierarchical containment structure with one root object. An MO that contains another MO is the superior (parent), and the contained MO is referred to as the subordinate (child). The parent of all MOs in a single name space is called a Local Root. The ultimate parent of all MOs of all managed systems is called the Global Root.

* + - 1. Network Resource

A network resource is a discrete entity represented by an IOC for the purpose of network and service management.

**Note:** A network resource may represent intelligence, information, hardware, and software of a telecommunication network.

* + - 1. Network Resource Model

A Network Resource Model (NRM) is a collection of IOCs, inclusive of their associations, attributes and operations, representing a set of network resources under management.

* + - 1. Global Root and Local Root

Names in a name space are organized in a hierarchy. An MO instance that contains another MO is referred to as the superior (parent), and the contained MO instance is referred to as the subordinate (child).

In modern network management, it is expected that the Enterprise name space is partitioned for implementations in multiple managed systems. The parent of all MO instances in a single managed system is called the Local Root. The ultimate parent of all MO instances of all managed systems is called the Global Root.

* + - 1. Distinguished Name and Relative Distinguished Name

A Distinguished Name (DN) is used to uniquely identify a MO within a name space. A DN is built from a series of name components, referred to as Relative Distinguished Names (RDNs). ITU-T Recommendation X.500 defines the concepts of DN and RDN in detail, using ASN.1, in the following way:

DistinguishedName ::= RDNSequence

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName RelativeDistinguishedName ::= SET SIZE (1..MAX) OF AttributeTypeAndValue AttributeTypeAndValue ::= SEQUENCE {type AttributeType,

value AttributeValue}

From a DN of a MO, one can derive the DN of its containing MO, if any. This containment relation is the only relation carried by the DN. No other relation can be carried or implied by the DN.

### Class Diagrams

This section provides class diagrams. Refer to the following sections:

* [Relationships](#_bookmark25)
* [Inheritance](#_bookmark31)
  + - 1. Relationships

This section provides an overview of relevant classes in UIM.

[Figure 5: UIM Class Diagram](#_bookmark26) shows the containment or naming hierarchy and the associations of the classes defined in UIM.

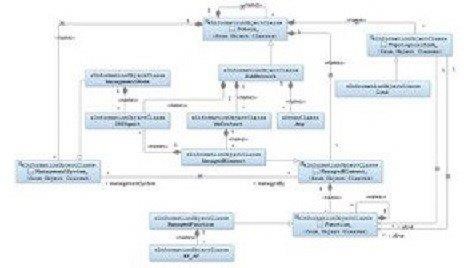


Figure 5: UIM Class Diagram

ManagedElement may be contained in the following:

* SubNetwork (since SubNetwork inherits from Domain\_ and ManagedElement inherits from ManagedElement\_ and Domain\_ name-contained ManagedElement\_ as seen in [Figure 5: UIM Class Dia-](#_bookmark26) [gram](#_bookmark26)
* MeContect instance as seen in [Figure 5: UIM Class Diagram](#_bookmark26)
* ManagedElement may also have no parent instance at all.

If the configuration contains several instances of SubNetwork, exactly one SubNetwork instance directly or indirectly contains all the other SubNetwork instances. The SubNetwork instance not contained in any other instance of SubNetwork is referred to as the root SubNetwork instance.

The root SubNetwork instance contains the ManagementNode. If contained in a SubNetwork instance, IRPAgent shall be contained in the root SubNetwork instance.

Each ManagedObject is identified with a DN that expresses its containment hierarchy according to 3GPP TS 32.300 [7]. For example, the DN of a ManagedElement instance could have a format as follows:

SubNetwork=Dallas, MeContext=NR-1, ManagedElement=gNB-1

[Figure 6: Performance Measurement Control NRM Fragment](#_bookmark27) is the class diagram for the Performance Measurement Control NRM fragment:

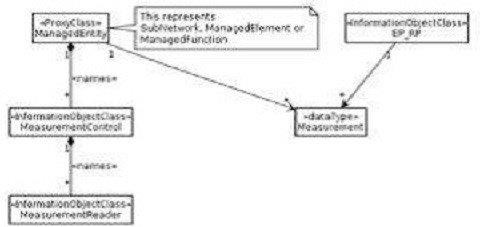


Figure 6: Performance Measurement Control NRM Fragment

[Figure 7: Measurement Threshold Monitoring Control Fragment](#_bookmark28) is the class diagram for the Measurement Threshold Monitoring Control fragment:

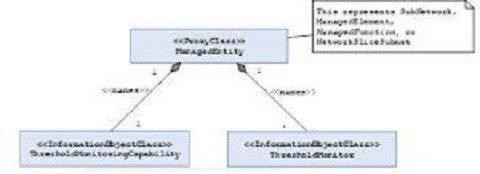


Figure 7: Measurement Threshold Monitoring Control Fragment

[Figure 8: Heartbeat Notification Control Fragment](#_bookmark29) is the class diagram for the Heartbeat Notification Control Fragment:



Figure 8: Heartbeat Notification Control Fragment

[Figure 9: Notification Subscription Control Fragment](#_bookmark30) is the class diagram for the Notification Subscription Control Fragment:

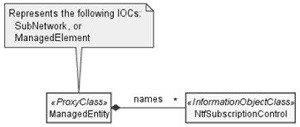


Figure 9: Notification Subscription Control Fragment

* + - 1. Inheritance

This section depicts inheritance relationships.

[Figure 10: Inheritance Hierarchy NRM Fragment](#_bookmark32) shows the Inheritance Hierarchy NRM Fragment:

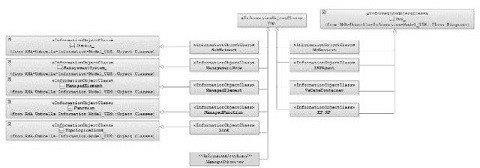


Figure 10: Inheritance Hierarchy NRM Fragment

[Figure 11: Performance Measurement Control NRM Fragment](#_bookmark33) shows the Performance Measurement Control NRM Fragment:

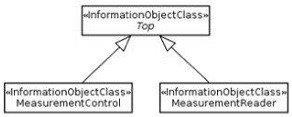


Figure 11: Performance Measurement Control NRM Fragment

[Figure 12: Measurement Threshold Monitoring NRM Fragment](#_bookmark34) shows the Measurement Threshold Monitoring NRM Fragment:

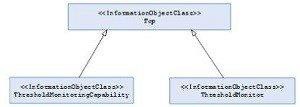


Figure 12: Measurement Threshold Monitoring NRM Fragment

[Figure 13: Heartbeat Notification Control Fragment](#_bookmark35) shows the Heartbeat Notification Control Fragment:

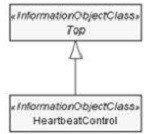


Figure 13: Heartbeat Notification Control Fragment

[Figure 14: Notification Subscription Fragment](#_bookmark36) shows the Notification Subscription Fragment:

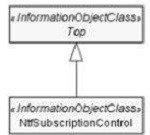


Figure 14: Notification Subscription Fragment

### Class Definitions

Refer to the following sections:

* [ManagedElement](#_bookmark38)
* [ManagedFunction](#_bookmark44)
* [ManagedNFService](#_bookmark49)
* [Top](#_bookmark54)
  + - 1. ManagedElement

Refer to the following sections:

* [Definition](#_bookmark39)
* [Attributes](#_bookmark40)
* [Attribute Constraints](#_bookmark42)
* [Notifications](#_bookmark43)
  + - * 1. Definition

This IOC represents telecommunication equipment or TMN entities within the telecommunications network providing support and service to the subscriber.

An ME communicates directly or indirectly with a manager over one or more management interfaces for the purpose of being monitored and/or controlled. MEs may or may not additionally perform element management functionality.

An ME contains equipment that may or may not be geographically distributed. An ME is often referred to as a Network Element.

A telecommunication equipment has software and hardware components. The IOC represents the case when the software component if designed to run on dedicated hardware component. In the case when the software is designed to run on ETSI NFC defined NFVI, the IOC description excludes the NFVI component supporting the subject software. A ManagedElement may be contained in either a SubNetwork or in a MeContext instance. A single ManagedElement may also exist standalone with no parent at all.

The ManagedElement IOC may be used to represent combined ME functionality (as indicated by the managedElementType attribute and the contained instances of different functional IOCs).

Single function ManagedElement IOC instances have a 1..1 containment relationship to a function IOC instance (in this context a function IOC instance is an instance of an IOC derived from the ManagedFunctino IOC). Multiple function ManagedElement instances will have a 1..N containment relationship to a function IOC instances.

**Note:** For some specific functional IOCs a 1..N containment relationship is permitted. The specific functional entities are identified in the NRMs that define subclasses of ManagedFunction

* + - * 1. Attributes

[Table 1: ManagedElement Attributes](#_bookmark41) describes the attributes:

Table 1: ManagedElement Attributes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Support Qualifier** | **isReadable** | **isWritab** | **leisInvariant** | **isNotifyable** |
| id | M | T | F | T | F |
| dnPrefix | M | T | F | F | T |
| managedElementTypeList | O | T | F | F | T |
| userLabel | M | T | T | F | T |
| locationName | M | T | F | F | T |
| managedBy | O | T | F | F | T |
| vendorName | M | T | F | F | T |
| userDefinedState | M | T | T | F | T |
| swVersion | M | T | F | F | T |
| priorityLabel | O | T | F | T | F |
| measurementsList | M | T | F | F | T |

* + - * 1. Attribute Constraints

The dnPrefix attribute is supported in an instance of ManagedElement which is the local root instance of the MIB. Otherwise, the attribute must be absent or must carry no information.

* + - * 1. Notifications

The common notifications are valid for this IOC.

* + - 1. ManagedFunction

Refer to the following sections:

* [Definition](#_bookmark45)
* [Attributes](#_bookmark46)
* [Attribute Constraints](#_bookmark48)
  + - * 1. Definition

This IOC is provided for sub-classing only. It provides attributes that are common to functional IOCs. A ManagedElement may contain several managed functions. The ManagedFunction may be extended in the future as more common characteristics to functional objects are identified.

This IOC can represent a telecommunication function either realized by software running on dedicated hardware or realized by software running on NFVI. Each ManagedFunction instance directly or indirectly communicates with a manager over one or more management interfaces exposed using its containing ME instance.

* + - * 1. Attributes

[Table 2: ManagedFunction Attributes](#_bookmark47) describes the attributes:

Table 2: ManagedFunction Attributes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Support Qualifier** | **isReadable** | **isWritab** | **leisInvariant** | **isNotifyable** |
| id | M | T | F | T | F |
| userLabel | M | T | T | F | T |
| vnfParametersList | CM | T | T | F | T |
| peeParametersList | CM | T | T | F | T |
| priorityLabel | O | T | F | T | F |
| measurementsList | M | T | F | F | T |

* + - * 1. Attribute Constraints

The following list describes the attribute constraints:

* vnfParametersList: The ManagedFunction instance is realized by one or more VNF instances. Otherwise, this attribute must be absent.
* peeParametersList: The control and monitoring of PEE parameters is supported by the ManagedFunction or sub-class instance.
  + - 1. ManagedNFService

Refer to the following sections:

* [Definition](#_bookmark50)
* [Attributes](#_bookmark51)
* [Attribute Constraints](#_bookmark53)
  + - * 1. Definition

A ManagedNFService represents an NF service. An NF service is one type of capability exposed by an NF Service Producer to another authorized NF Service Consumer through a service-based interface. An NF may expose one or more NF services. Following are criteria for specifying NF services:

* NF services are derived from the system procedures that describe end-to-end functionality, where applicable.
* System procedures are described by a sequence of NF service invocations.
  + - * 1. Attributes

[Table 3: ManagedNFService Attributes](#_bookmark52) describes the attributes:

Table 3: ManagedNFService Attributes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Support Qualifier** | **isReadable** | **isWritab** | **leisInvariant** | **isNotifyable** |
| id | M | T | F | T | F |
| userLabel | O | T | T | F | T |
| nFServiceType | M | T | F | T | F |
| sAP | M | T | T | F | T |
| operations | M | T | T | F | T |
| administrativeState | M | T | T | F | T |
| operationalState | M | T | F | T | T |
| usageState | M | T | F | T | T |
| registrationState | CM | T | F | F | T |

* + - * 1. Attribute Constraints

The registrationState attribute must be supported by instance of a ManagedNFService if the service is designed to be published and discovered by other NFs. It must also be registered to a repository function.

* + - 1. Top

Refer to the following sections:

* [Definition](#_bookmark55)
* [Attributes](#_bookmark56)
  + - * 1. Definition

This IOC is provided for sub-classing only. All information object classes defined in all TS that claim to be conformant to 32.102 [7] inherit from Top.

* + - * 1. Attributes

[Table 4: Top Attributes](#_bookmark57) describes the attributes:

Table 4: Top Attributes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| id | M | T | F | T | F |

### Attribute Definitions

[Table 5: Attribute Definitions](#_bookmark59) defines the properties of attributes:

Table 5: Attribute Definitions

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Description and Values** | **Properties** |
| userLabel | A user-friendly (and user assignable) name of this object. | type: String multiplicity: 0..1 |
|  | allowedValues: N/A | isOrdered: N/A |
|  |  | isUnique: N/A |
|  |  | defaultValue: No default value |
|  |  | isNullable: False |
| locationName | The physical location of this entity (e.g. an address). | type: String multiplicity: 0..1 |
|  | allowedValues: N/A | isOrdered: N/A |
|  |  | isUnique: N/A |
|  |  | defaultValue: No |
|  |  | isNullable: False |
| vendorName | The name of the vendor. allowedValues: N/A | type: String multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False |
| userDefinedState | An operator defined state for operator specific usage. | type: String multiplicity: 0..1 |
|  | allowedValues: N/A | isOrdered: N/A |
|  |  | isUnique: N/A |
|  |  | defaultValue: No default value |
|  |  | isNullable: False |

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Description and Values** | **Properties** |
| swVersion | The software version of the ManagementNode or ManagedElement. This is used for determining the version of the vendor specific information valid for the ManagementNode or ManagedElement. | type: String multiplicity: 0..1 isOrdered: N/A isUnique: N/A  defaultValue: No default value |
|  | allowedValues: N/A | isNullable: False |
| priorityLabel | This is a label that a consumer would assign a value on a concrete instance of the managed object. The management system takes the value of this attribute into account. The effect of this attribute value to the subject managed entity is not standardized. | type: Integer multiplicity: 1 isOrdered: N/A isUnique: N/A  defaultValue: No default value isNullable: False |
| measurementsList | It specifies a list of supported measurements and their GPs. A NULL value indicates there is no measurement supported. | type: Measurements multiplicity: \* isOrdered: N/A |
|  | allowedValues: N/A | isUnique: N/A |
|  |  | defaultValue: None |
|  |  | allowedValues: N/A |
|  |  | isNullable: True |

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Description and Values** | **Properties** |
| vnfParametersList | This attribute contains the pa- | type: String |
|  | rameter set of the VNF in-  stance(s) corresponding to an | multiplicity: \* |
|  | NE. Each entry in the list con- | isOrdered: N/A |
|  | tains: | isUnique: True |
|  | * vnfInstanceId * vnfdId (optional) | defaultValue: None |
|  | * flavourId (optional) | isNullable: True |
|  | * autoScalable |  |
|  | vnfInstanceId: VNF instance |  |
|  | identifier (vnfInstanceId, see |  |
|  | section 9.4.2 of [16] and section |  |
|  | B2.4.2.1.2.3 of [17]). |  |
|  | **Note:** The value of this attribute |  |
|  | is identical to that of the same |  |
|  | attribute in clause 9.4.2 of ETSI |  |
|  | GS NFV-IFA 008 [16]. |  |
|  | vnfdId: Identifier of the VNFD |  |
|  | on which the VNF instance is |  |
|  | based, see section 9.4.2 of [16]. |  |
|  | This attribute is optional. |  |
|  | **Note:** The value of this attribute |  |
|  | is identical to that of the same |  |
|  | attribute in clause 9.4.2 of ETSI |  |
|  | GS NFV-IFA 008 [16]. |  |
|  | flavourId: Identifier of the VNF |  |
|  | Deployment Flavour applied to |  |
|  | this VNF instance, see section |  |
|  | 9.4.3 of [16]. This attribute is op- |  |
|  | tional. |  |
|  | **Note:** The value of this attribute |  |
|  | is identical to that of the same |  |
|  | attribute in clause 9.4.3 of ETSI |  |
|  | GS NFV-IFA 008 [16]. |  |

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Description and Values** | **Properties** |
| vnfParametersList (continued) | autoScalable: Indicator of whether the auto-scaling of this VNF instance is enabled or disabled. The type is Boolean.  **Note:** The value of this attribute is identical to that of the same attribute included in vnfConfigurableProperty in clause 9.4.2 of ETSI GS NFV-IFA 008 [16].  The presence of this attribute indicates that the ManagedFunction represented by the MOI is a virtualized function.  **Note:** The presence of the attribute vnfParametersList, whose vnfInstanceId with a string length of zero, in createMO operation can trigger the instantiation of the related VNF/ VNFC instances.  allowedValues: N/A  A string length of zero for vnfInstanceId means the VNF instance(s) corresponding to the MOI does not exist (For example, has not been instantiated yet, has already been terminated). |  |

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Description and Values** | **Properties** |
| peeParametersList | This attribute contains the parameter list for the control and monitoring of power, energy and environmental parameters of ManagedFunction instance(s). This list contains the following parameters: | type: String multiplicity: 0..\* isOrdered: N/A isUnique: True  defaultValue: No default value |
|  | * siteIdentification | isNullable: True |
|  | * siteLatitude (optional) |  |
|  | * siteLongitude (optional) |  |
|  | * siteDescription |  |
|  | * equipmentType |  |
|  | * environmentType |  |
|  | * powerInterface |  |
|  | siteIdentification: The identification of the site where the ManagedFunction resides. |  |
|  | allowedValues: N/A |  |
|  | siteLatitude: The latitude of the site where the ManagedFunction instance resides, based on World Geodetic System (1984 version) global reference frame (WGS 84). Positive values correspond to the northern hemisphere. This attribute is optional in case of BTSFunction and RNCFunction instance(s). |  |
|  | allowedValues: -90.0000 to  +90.0000 |  |
|  | siteLongitude: The longitude of the site where the ManagedFunction instance resides, based on World Geodetic System (1984 version) global reference frame (WGS 84). Positive values correspond to degrees east of 0 degrees longitude. This attribute is optional in case of BTSFunction and RNCFunction instance(s). |  |
|  | allowedValues: -180.0000 to  +180.0000 |  |

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Description and Values** | **Properties** |
| peeParametersList (continued) | siteDescription: An operator defined description of the site where the ManagedFunction instance resides.  allowedValues: N/A  equipmentType: The type of equipment where the managedFunction instance resides.  allowedValues: see clause 4.4.1 of ETSI ES 202 336-12 [18].  environmentType: The type of environment where the managedFunction instance resides.  allowedValues: see clause 4.4.1 of ETSI ES 202 336-12 [18].  powerInterface: The type of power.  allowedValues: see clause 4.4.1 of ETSI ES 202 336-12 [18]. |  |
| nFServiceType | The parameter defines the type of the managed NF service instance  allowedValues: See clause 7.2 of TS 23.501[22] | type: ENUM multiplicity: 1 isOrdered: N/A isUnique: True  defaultValue: No default value isNullable: False |
| sAP | This parameter specifies the service access point of the managed NF service instance.  allowedValues: N/A | type: SAP multiplicity: 1 isOrdered: N/A isUnique: N/A  defaultValue: No default value  isNullable: False |

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Description and Values** | **Properties** |
| operations | This parameter defines set of operations supported by the managed NF service instance.  allowedValues: See TS 23.502[23] for supporting operations | type: Operation multiplicity: 1..\* isOrdered: False isUnique: False  defaultValue: No default value  isNullable: False |
| administrativeState | This parameter indicates the administrative state of the manaed object instance. It describes the permission to use or prohibition against using the instance, imposed through the management services.  allowedValues: "LOCKED", "UNLOCKED", SHUTTINGDOWN"  The meaning of these values is as defined in 3GPP TS 28.625  [21] and ITU-T X.731 [19]. | type: ENUM multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False |
| operationalState | This parameter indicates the operational state of the managed NF service instance. It describes whether the resource is physically installed and working.  allowedValues: "ENABLED", "DISABLED".  The meaning of these values is as defined in 3GPP TS 28.625  [21] and ITU-T X.731 [19]. | type: ENUM multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False |
| registrationState | This parameter defines the registration status of the managed NF service instance.  allowedValues: "Registered", "Deregistered". | type: ENUM multiplicity: 1 isOrdered: N/A isUnique: N/A  defaultValue: Deregistered  isNullable: False |

# Packaging Network Functions

Mavenir Cloud-Native Network Function (NF) is a containerized and cloud-native network function, which is used for packaging, deploying, and managing network functions.

CNF packaging ensures simplicity and consistency. It ensures that the microservice developer only needs to focus on deployment values, configuration schema and data related to the microservice.

Helm is used as the package manager for CNF. Helm uses a packaging format called charts. A chart is a collection of files that describe a related set of Kubernetes resources. A single chart might be used to deploy a simple single microservice pod, or a complex NF stack with HTTP servers, databases, caches, and so on.

Charts are created as files laid out in a particular directory tree, which can be packaged as versioned archives to be deployed. Refer [Annexure C](#_bookmark243) for details about upgrading the NF chart version. The following is a sample chart structure:

SampleApplication/

Chart.yaml # A YAML file containing

information about the chart LICENSE # OPTIONAL: A plain text file

containing the license for the chart README.md # OPTIONAL: A human-readable README file

requirements.yaml # OPTIONAL: A YAML file listing dependencies for

the chart

values.yaml # The default configuration values for this chart charts/ # A directory containing any charts upon which this

chart depends.

templates/ # A directory of templates that, when combined with values, will generate valid Kubernetes manifest files.

templates/NOTES.txt # OPTIONAL: A plain text file containing short usage

notes

Refer to the following sections for more information on packaging:

###### [Metadata Packaging](#_bookmark61)

###### [Configuration Packaging](#_bookmark62)

###### [CNF Event Definition Packaging](#_bookmark63)

###### [CNF Dashboard Injection](#_bookmark64)

###### [Defining Microservice Secret Definitions](#_bookmark65)

###### [Variable Substitution](#_bookmark66)

## Metadata Packaging

As part of metadata template, each resource name is inherited from the microservice name as defined in the charts.yaml in chart definition. To achieve multiple resource definition for same type of resource, include the “ext\_name” key in values.yaml. As a result, the resource name is specified as <microservicename>[-<ext\_name>]

You can specify customized labels and annotations to any Kubernetes resource using the values.yaml file.

Every NF chart definition must include cnfTplHeader in the values.yaml file and pass the variable cnfHdr as an argument to metadata template API.

## Configuration Packaging

At CNF layer, any microservices static, environment, and configuration management definitions are specified as configuration resources.

The CNF chart template auto injects the files present in following directory structure as configuration resources:

NF/Microservice chart definition config

static : All static/init variables env : All environment variables

mgmt : All config management variables

XGVela does not provide configuration management for static or environment variables.

Each configuration type follows these naming conventions which can be used to mount in pod as volume definitions:

* static: <microservicename>-static-cfg
* env: <microservicename>-env-cfg
* mgmt: <microservicename>-mgmt-cfg

**Note:** Chart template 2.10 onwards provides support for subfolders in static cfg, to support config contents of size greater than 1 MB.

Example Usage:

config/static

spec1.json spec2.json

/nsmf

spec3.json spec4.json

/namf

specx.json specy.json

This structure generates the following configmaps:

* <service-name>-static-cfg with files spec1.json and spec2.json
* <service-name>-nsmf-static-cfg with files spec3.json and spec4.json
* <service-name>-namf-static-cfg with files specx.json and specy.json

You can import all environment variables in a container as part of “envfrom” or import individual environment variables as part of “env” in the container specification.

Configuration management definitions are packaged with yang model (.yang) and json data set (.json) files. NFs can mount the json files at respective locations.

## CNF Event Definition Packaging

At CNF layer, you can specify any microservices event definitions for FMaaS integration as a JSON file, which is converted as configuration resources.

The CNF chart template auto injects the files present in following directory structure as configuration resources.

nf:

NF/Microservice chart definition eventdef

<evendef json files>

The following naming convention is used for eventdef Configmap:

<nf>/<microservicename>-eventdef-cfg

You must set the create flag to true as shown in the following example to enable eventdef template definitions:

nf:

cnftpl\_configmap: cnftpl\_eventdef:

create: true metaspec: {}

## CNF Dashboard Injection

At CNF layer, NF Grafana dashboards for microservices are injected to PAAS Grafana using the CNF chart template.

You must package all JSON dashboard definitions under following directory structure:

NF/Microservice charts Dashboard

<dashboard Json files>

You must set the create flag to true as shown in the following example:

nf:

cnftpl\_dashboard:

create: true metaspec: {}

## Defining Microservice Secret Definitions

You can specify any microservices secret definitions can be specified as secret resources. You must package all secret definitions under the following directory structure.

NF/Microservice charts secret

<token files>

You must set the create flag to true as shown in the following example:

|  |  |
| --- | --- |
| cnftpl\_secret: |  |
| create: true |
| metaspec: {} |
| secretspec: {} | |

## Variable Substitution

At CNF layer, all resources types (metadata and resource specification) can use variable substitution using values.yaml variable declaration.

The CNF template also supports variable substitution, and the following must be defined in the values.yaml file.

* **All Resource Metadata:** Customized metadata is already defined in the values.yaml file, which is injected to the Kubernetes cluster as part of CNF template.
* **ConfigMap**: You must declare the variables in the values.yaml file, and can use static, environment and management variables in files as described in the ConfigMap template.
* **Namespace:** Namespace conventions as explained in namespace template supports variable substitution.

Static Configuration Example:

{{ .Values.nf\_variables.static\_cfg.intf\_name1 }}={{ .Values.nf\_vari- ables.static\_cfg.ipaddr1 }}

{{ .Values.nf\_variables.static\_cfg.intf\_name2 }}="{{ .Values.nf\_vari- ables.static\_cfg.ipaddr2}}" role="active"

In this example, intf\_name1 and intf\_name2 are substituted with the actual interface name during execution. Static configurations must not be dynamically updated.

Environment configuration example:

env\_max\_connection: "{{.Values.nf\_variables.env\_cfg.max\_connection}}" env\_cim\_port: "{{.Values.nf\_variables.env\_cfg.cim\_port}}"

env\_extdb\_uri: "https://mav.couchdb.com:8081"

Environment variables can be dynamically updated during run-time. Management configuration example (json):

{

"nfPeerConfig": { "nssf": [

{

"mcc": {{ .Values.nf\_variables.mgmt\_cfg.nssf.mcc1 }}, "mnc": 1111,

"fqdn": "smffqdn",

"ipv4": "{{.Values.nf\_variables.mgmt\_cfg.nssf.ip}}"

},

{

"mcc": {{ .Values.nf\_variables.mgmt\_cfg.nssf.mcc2 }}, "mnc": 1111,

"fqdn": "smffqdn",

"ipv4": "{{.Values.nf\_variables.mgmt\_cfg.nssf.ip}}"

},

{

"mcc": 9886,

"mnc": 3333,

"fqdn": "smffqdn", "ipv4": "localhost"

}

]

}

}

In this example, mcc1, mcc2, and ip substituted with actual values during run-time.

# XGVela Release Artifacts

The following files and folders are included as part of the XGVela release:

Table 6: XGVela Release Artifacts

|  |  |
| --- | --- |
| **File/Folder Name** | **Description** |
| images | Folder containing the Docker images for the XGVela release. |
| md5sum.info | The MD5 sum of all the artifacts in the release. |
| xgvela-<version>-sdk.tgz | The development artifacts such as configuration and template. |
| xgvela-<version>.tgz | Contains the XGVela package. |

# CIM

Container Interface Module (CIM) provides an interface between XGVela and managed NFs or microservices. NFs or microservices must include CIM as a sidecar in each POD. CIM implements various single node design patterns to enable loose coupling of NF or microservice containers to the infrastructure. It supports a fast pub/sub, streaming and RPC framework (NATS) to support intra pod communication for a complex microservice scenarios.

**Note:** Refer to [Annexure D](#_bookmark244) for information about upgrading the CIM version.

To use CIM, NFs or microservices must mount the cim.json file to /opt/conf, by modifying the deployment.yaml file. The cim.json and cim.yang files are delivered as part of the XGVela SDK, and must be included as part of the mgmt config of individual microservices.

**Note:** CIM 1.10 onwards eliminates the dependency on applications to include the CIM configuration (cim.json and cim.yang) and CIM event definition as part of the application chart. Refer to [CIM Beta Version Enhancements](#_bookmark71) for details.

**Note:** The cim.yang module namespace prefix must be tuned according to application naming conventions to make it unique.

CIM needs certain privileges to interface with Kubernetes. These privileges must be provided using RoleBased Access Control (RBAC). The following rules must be defined for the role:

Note:

Optimized Role for CIM. **name** and **namespace** will change as per the application that is used.

kind: Role

apiVersion: rbac.authorization.k8s.io/v1 metadata:

name: test-app-role

namespace: <application namespace>-appln-test-app rules:

* apiGroups: ["extensions", "apps"] resources: ["statefulsets","deployments"] verbs: ["get", "patch"]
* apiGroups: [""] resources: ["pods"]

verbs: ["get", "watch", "delete", "list", "patch"]

* apiGroups: [""] resources: ["configmaps"] verbs: ["get", "create"]

The following annotation is required in the chart:

annotations:

topogw.fqdn: "topo-gw.<XGVela namespace>.svc.cluster.local:8080"

The following is an example of role binding for CIM:

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: test-app-binding

namespace: <application namespace>-appln-test-app subjects:

- kind: ServiceAccount

name: test-app-serviceaccount

namespace: <application namespace>-appln-test-app roleRef:

kind: Role

name: test-app-role

apiGroup: rbac.authorization.k8s.io

You must configure the following values in the cim.json file:

Table 7: cim.json Attribute Descriptions

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Values** | **Description** |
| app\_port | 5000-10000 | Specifies the HTTP port to communicate with server. The default value is 9999. The parameter can be configured according to NF or microservice requirement. |
| http2\_enabled | * true * false | Enables HTTP2 support at CIM. NFs which require HTTP2 communication can enable this flag during deployment time. The default value is false. |

The CIM REST port and NATS port are configurable. The CIM REST port can be configured by providing the CIM\_REST\_PORT environment variable to the CIM container. Similarly, the CIM NATS port can also be configured by providing the CIM\_NATS\_PORT environment variable to the CIM container. Both the values are required to be in string.

Example:

env:

* name: CIM\_REST\_PORT value: "6969"
* name: CIM\_NATS\_PORT value: "4777"

In case these values are not provided then default CIM REST port is 6060 and NATS port is 4222. Refer to the following sections:

###### [CIM Beta Version Enhancements](#_bookmark71)

###### [Additional CIM Settings](#_bookmark72)

###### [XGVela Registration](#_bookmark74)

###### [Shutdown](#_bookmark75)

###### [CIM Liveliness and Readiness Settings](#_bookmark76)

## CIM Beta Version Enhancements

The latest CIM version eliminates the dependency on applications to include the CIM configuration (*cim.json* and *cim.yang* ) and CIM event definition as part of the application chart.

CIM includes basic configurations for all features as follows:

{

"cimConfig": {

"lmaas": {

"max\_file\_size\_in\_mb": 10,

"max\_backup\_files": 10,

"max\_age": 7,

"buffer\_size": 10000,

"flush\_timeout": 30, "logging\_mode": "FILE", "prefer\_local\_daemonset": false

},

"cim\_settings": {

"enable\_kpi": true, "log\_level": "ALL", "cim\_file\_log": true, "num\_garp\_count": 1,

"remote\_svc\_retry\_count": 60,

"ttl\_timeout": 1

},

"app\_settings": {

"app\_port": 9999, "http2\_enabled": false

}

}

}

Applications can override these default values during XGVela registration. Application settings must be set correctly as part of the registration request.

A sample registration request is as follows:

{

"cimConfig": {

"lmaas": { "max\_file\_size\_in\_mb": 10,

"max\_backup\_files": 10,

"max\_age": 7,

"buffer\_size": 10000,

"flush\_timeout": 30, "logging\_mode": "FILE", "prefer\_local\_daemonset": false

},

"app\_settings": {

"app\_port": 9999, "http2\_enabled": false

}

}

}

The latest CIM version has reduced the dependency on environment variables. Only the following environment variables are required from CIM 1.10 onwards:

* K8S\_POD\_ID
* K8S\_NAMESPACE
* MS\_CONFIG\_REVISION
* NF\_CONFIG\_REVISION

The latest CIM version is backward-compatible, where the application can use the previous behavior in which the CIM configuration (cim.json and cim.yang) and CIM event definition had to be included as part of application chart.

## Additional CIM Settings

You can specify CIM settings using the cim.json file as shown in the following example:

"cim\_settings": {

"enable\_kpi": true, "http2\_enabled": false, "log\_level": "ALL", "cim\_file\_log": true, "num\_garp\_count": 1,

"remote\_svc\_retry\_count": 60,

"ttl\_timeout": 1

}

[Table 8: cim.json Attribute Descriptions](#_bookmark73) describes the settings which can be configured from the cim.json file:

Table 8: cim.json Attribute Descriptions

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Values** | **Description** |
| enable\_kpi | * true * false | Indicates whether to enable KPI data for CIM. |
| log\_level | * INFO * DEBUG | Specifies the log level for CIM logs. |
|  | * ERROR |  |
|  | * WARNING |  |
|  | * EXCEPTION |  |
|  | * ALL |  |
| cim\_file\_log | * true * false | Indicates whether to enable file logging for CIM. The default value is true.  **Note:** It is not recommended to change this value to false, as it is currently not supported. |
| num\_garp\_count | 0-5 | Indicates the number of Gratuitous ARP packets sent. |

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Values** | **Description** |
| remote\_svc\_retry\_count | 1-600 | Indicates the number of retries for dependent service discoveries. |
| ttl\_timeout | 1-600 | Indicates the TTL timeout. |

## XGVela Registration

This interface is used for XGVela registration.

**Interface** REST

**URI** /api/v1/\_operations/xgvela/register

**Method** POST

**HTTP Port** 6060

**HTTP Method** 1.1/2 **ContentType** application/json **Request Body**

{

"container\_id" : "<app containerid>", "container\_name" : "<app containername>", "lmaas": {

"max\_file\_size\_in\_mb": 10,

"max\_backup\_files": 10,

"max\_age": 7,

"buffer\_size": 10000,

"flush\_timeout": 30, "logging\_mode": "FILE", "prefer\_local\_daemonset": false

},

"app\_settings": {

"app\_port": 9999, "http2\_enabled": false

}

}

**Note:** lmaas and app\_settings are optional, and are not required if CIM is running in backward-compatible mode. If values are not provided, the default values will be considered. It is recommended to change the default values for app\_settings according to the application requirement.

**Response Status** 200 OK

Response Body

{

"status" : "<success/failure>",

"err\_msg" : "<error message in case of failure>", "tmaas\_fqdn" : "<tmaas fqdn value>", "apigw\_rest\_fqdn" : "<apigw fqdn value>",

"apigw\_rest\_uri\_prefix" : "<apigw rest uri prefix value>", "apigw\_auth\_type" : "<auth type value>",

"apigw\_username" : "<username to be used in auth>", "apigw\_password" : "<password to be used in auth>", "tpaas\_fqdn": <url>,

"topo\_engine\_fqdn": <url>, "tmaas\_annotations": <jsonobject>

}

Other Response Codes

400 Bad request

404 Not Found

405 Method Not Allowed 503 Service Unavailable

505 HTTP Version Not Supported

**Note:** You can use tmaas\_fqdn from the response to access the APIs exposed by TMaaS, such as scaleout and scalein. Example: http://<tmaas\_fqdn>/<API Path>apigw\_auth\_type: XGVela currently supports auth type Basic.apigw\_rest\_fqdn and apigw\_rest\_uri\_prefix can be used to lookup service details from the topology

**Note:** The err\_msg field is only available in case of "status":"failure"

## Shutdown

This interface is used for graceful shutdown.

**Interface** REST

**URI** /api/v1/\_operations/shutdown

**Method** POST

**HTTP Port** Configurable **HTTP Method** 1.1/2 **Request Body** N/A **Response Body**

200 OK

Any [4xx-5xx] response code is treated as a failure.

**Note:** For graceful shutdown to function properly, configure CIM liveliness and readiness probe for CIM and also for the application.

## CIM Liveliness and Readiness Settings

Ensure that the CIM container specification is configured with liveliness and readiness probes. The following is a sample configuration:

livenessProbe:

tcpSocket:

port: 6060

initialDelaySeconds: 120

periodSeconds: 2

timeoutSeconds: 5

failureThreshold: 1 readinessProbe:

tcpSocket:

port: 6060

initialDelaySeconds: 5

periodSeconds: 2

timeoutSeconds: 5

failureThreshold: 1

# API Gateway

API Gateway provides a set of APIs for applications to query and discover services and its details available as per the topology model.

The following APIs are currently supported:

###### [Get NFService By Name](#_bookmark78)

###### [Get NFServiceInstance By ID](#_bookmark79)

## Get NFService By Name

**Interface** : REST

URI: http://<apigw\_rest\_fqdn> <apigw\_rest\_uri\_prefix>

**Method** : POST

**HTTP Version** : 1.1

**ContentType** : application/yang-data+json

**Accept** : application/yang-data+json

**Authorization** : Basic admin:admin //These values are samples. Refer auth\_type and username/password from the XGVela Registration API.

Request Body

{

"immediate-query": {

"foreach": "/ManagedElement/NetworkFunction[attributes/name='<nf name>']/NFService[attributes/name='<service name>']",

"select": [{

"expression": ".", "result-type": "inline"

}

],

"timeout": 10

}}

**Response Status** : 200 OK

Other HTTP error codes as applicable

**Note:** apigw\_rest\_fqdn and apigw\_rest\_uri\_prefix are available as part of XGVela registration response<nf name>: Refers to the nfId from the values.yaml file.Examples: cucp1 and amf1 <service name>: Refers to the micro service name from the chart.yaml file.Examples: gnbmgr and amfee

Sample Response Body :

Refer to [NFService By Name](#_bookmark241) for sample response body.

## Get NFServiceInstance By ID

**Interface** : REST

**URI** : http://<apigw\_rest\_fqdn> <apigw\_rest\_uri\_prefix>

**Method** : POST

**HTTP Version** : 1.1

**ContentType** : application/yang-data+json

**Accept** : application/yang-data+json

**Authorization** : Basic admin:admin //These values are samples. Refer auth\_type and username/password from the XGVela Registration API.

Request Body

{

"immediate-query": {

"foreach": "/ManagedElement/NetworkFunction[attributes/name='<nf name>']/NFService[attributes/name='<service name>']/NFServiceInstance[id='<nfServiceInstanceId>']", "select": [{

"expression": ".", "result-type": "inline"

}

],

"timeout": 10

}

}

**Response Status** : 200 OK

Other HTTP error codes as applicable

**Note:** apigw\_rest\_fqdn and apigw\_rest\_uri\_prefix are available as part of the XGVela registration response.<nf name>: Refers to the nfId from the values.yaml file.Examples: cucp1 and amf1 <service name>: Refers to the micro service name from the chart.yaml file.Examples: gnbmgr and amfee<nfServiceInstanceId>: Refers to the MO ID of the NFServiceInstance, which is available as the nfServiceInstanceId in the addtitionalFields section of the NFServiceInstance topology notifications.

Sample Response Body :

Refer to [NFServiceInstance by ID](#_bookmark242) for sample response body.

# TMaaS

In a networked environment, majority of the network functions, especially carrier class functions, comprise of many interrelated components that need to be individually managed by the management system.

For each function in the network, there are many different components that need to be discovered, monitored and managed by a management system. For the management system to manage a set of network functions, the network functions are represented or modeled as objects that can be stored in a database to monitor state and perform management operations.

The Topology service creates models of networks, functions, and components within each function as managed objects. It provides a hierarchical view of all the network functions and associated resources within XGVela and manages the objects individually or as a group of related objects.

TMaaS provides the following services:

* Modeling and mapping of network functions
* Provides namespace and identifiers
* NF view with associated microservices and resources (containers, pod, volumes, configuration)
* Status monitoring
* NF and microservice state events
* APIs to monitor and manage network elements
* NF and microservice operations

[Figure 15: TMaaS Design](#_bookmark81) describes the TMaaS design:



Figure 15: TMaaS Design

TMaaS consists of the following components:

* **TMaaS-GW** A Kubernetes client service that retrieves information about microservices and associated resources using a Kubernetes API.
* **TMaaS** Implements the core functionality of TMaaS, generates, and manages NF topology. It receives cluster information from KubeC and any changes to them as and when they occur. It generates topology change events whenever there is a change in the microservice or NF states, a new POD is added, or when an existing POD is removed.Topo stores topology information and state in a hierarchical tree model per NF.
  + Responsible for driving various NF and microservice policies. Policies are formulas to derive NF and microservice states, statuses, and overload conditions. TMaaS is shipped with default policies.
  + Provides APIs for Topo modules to publish and subscribe to events from FMaaS.
  + Command execution framework for various ManagedObjects in the managed topology. The ManagedObject model defines generic operations such as AdminMode, OperationsStatus, and Scaling. You can also define custom operations.
  + to access topology information and execute operations.

Refer to the following sections:

###### [TMaaS-GW](#_TMaaS-GW)

###### [TMaaS](#_bookmark83)

###### [TMaaS Chart Specification](#_bookmark98)

###### [TMaaS Events](#_bookmark99)

###### [TMaaS Interfaces](#_bookmark102)

###### [Operation Specification Registration](#_bookmark103)

## TMaaS-GW

TMaaS-GW is a Kubernetes client service that retrieves information about microservices and associated resources using a Kubernetes API.

TMaaS-GW connects to the Kubernetes API server and registers a POD watcher to get notified of the changes within the managed namespace. Managed namespaces are namespaces with the current XGVela context at the root as described in [CNF Deployment Model](#_bookmark9).

It generates the following types of events on a POD:

* ADDED
* DELETED
* MODIFIED
* ERROR

## TMaaS

Topo implements the NF topology builder, microservice, and the NF state machine. It coordinates with Policy Controller (PoliC) for various state derivations and propagation of the same in the managed topology. It generates topology change events to notify occurring changes.

### Kubernetes Mapping

This section describes the mapping between XGVela attributes and corresponding the Kubernetes attributes:

Refer to the following sections:

* [ManagedElement](#_bookmark85)
* [NetworkFunction](#_bookmark87)
* [NFService](#_bookmark89)
* [NFServiceInstance](#_bookmark91)
  + - 1. ManagedElement

[Table 9: ManagedElement Mapping](#_bookmark86) describes the Kubernetes mapping to ManagedElement attributes:

Table 9: ManagedElement Mapping

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Support Qualifier** | **Value Mapping** |
| id | M | me-${xgvela.com/tmaas}.xgvelaId |
| dnPrefix | M | ${xgvela.com/tmaas}.dnPrefix |
| userLabel | M | Default: ${xgvela.com/tmaas}.nfId |
| locationName | M | ${xgvela.com/tmaas}.locationName |
| managedBy | O | ${tmaas.mns.address.primary|secondary} |
| vendorName | M | ${xgvela.com/tmaas}.vendorName |
| userDefinedState | O | Default: NULL |
| swVersion | M | ${xgvela.com/tmaas}.swVersion |
| priorityLabel | O | Not Supported |
| measurementsList | M | Not Supported |
| M = Mandatory O = Optional |  | |

To achieve a unique meId for a cluster, ensure that xgvelaId is unique in the chart for the XGVela deployment.

The locationName, swVersion, and managedBy parameters of ManagedElement can be set using the values.yaml file of the XGVela deployment using the following structure:

xgvela:

xgvela-mgmt: topo-engine:

componentSpec: deployment:

pod\_metaspec: annotations:

xgvela.com/me:

'{"locationName":"bangalore-1","swVersion":"v1.0", "managedBy":{"mns.address.primary": "IP=mns@primary","mns.address.secondary": "IP=mns@secondary"}}'

* + - 1. NetworkFunction

[Table 10: NetworkFunction Mapping](#_bookmark88) describes the Kubernetes mapping to NetworkFunction attributes:

Table 10: NetworkFunction Mapping

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Support Qualifier** | **Value Mapping** |
| id | M | ${xgvela.com/tmaas}.nfId |
| userLabel | M | Default: ${xgvela.com/tmaas}.nfId |
| nfType | M | ${xgvela.com/tmaas}.nfType |
| state | M | As per NF State Policy |
| adminstrativeState | M | Not Supported. Default: UNLOCKED |
| operationalState | M | Not Supported. Default: ENABLED |
| usageState | M | Not Supported. Default: ACTIVE |
| priorityLabel | O | Not Supported |
| vnfParametersList | O | Not Supported |
| peeParametersList | M | Not Supported |
| M = Mandatory O = Optional |  | |

* + - 1. NFService

[Table 11: NFService Mapping](#_bookmark90) describes the Kubernetes mapping to NFService attributes:

Table 11: NFService Mapping

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Support Qualifier** | **Value Mapping** |
| id | M | ${xgvela.com/tmaas}.nfServiceId |
| userLabel | M | ${xgvela.com/tmaas}.dnPrefix |
| nFServiceType | M | ${xgvela.com/tmaas}.nfServiceType |
| state | M | As per NF Service State Policy |
| adminstrativeState | M | Not Supported. Default: UNLOCKED |
| operationalState | M | Not Supported. Default: ENABLED |

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Support Qualifier** | **Value Mapping** |
| usageState | M | Not Supported. Default: ACTIVE |
| M = Mandatory O = Optional |  | |

* + - 1. NFServiceInstance

[Table 12: NFServiceInstance Mapping](#_bookmark92) describes the Kubernetes mapping to NFServiceInstance attributes:

Table 12: NFServiceInstance Mapping

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Support Qualifier** | **Value Mapping** |
| id | M | $k8s.pod.objectmeta.name |
| dnPrefix | M | ${xgvela.com/tmaas}.dnPrefix |
| userLabel | M | Default: $k8s.pod.objectmeta.name |
| state | M | As per NF Service Instance State Policy |
| M = Mandatory O = Optional |  | |

### States and Transitions

An NF can assume a number of internal states to represent its status. Transitions between these states provide architectural patterns for some expected NF functionality. Before an NF can start its lifecycle, the NF must be on-boarded by uploading the NF package which contains charts, images, and data.

[Table 13: NF States](#_bookmark94) describes the states that an NF can assume:

Table 13: NF States

|  |  |
| --- | --- |
| **State** | **Description** |
| Null | A VNF Instance does not exist, and is about to be created. |
| Instantiated Not Configured | A VNF Instance exists, but is not configured for service. |
| Instantiated Configured Inactive | A VNF Instance is configured for service. |
| Instantiated Configured Active | A VNF Instance that participates in service. |
| Terminated | A VNF Instance has ceased to exist. |

[Table 14: State Transitions](#_bookmark95) describes the state transitions:

Table 14: State Transitions

|  |  |  |
| --- | --- | --- |
| **Transition Action** | **Reverse Transition Action** | **Description** |
| Instantiate | Terminate | Instantiate results in the creation of an NF instance in an Instantiated Not Configured state. The Kubernetes resources necessary for a particular NF instance are allocated and are in use.  Terminate deletes a previously created NF instance. All Kubernetes resources are released and made available for other NFs. |
| Configure | N/A | Configure sets or changes NF instance parameters in such way that it enables the NF instance  to participate in a desired service. For example, a virtual DRA (vDRA) configured for participation in VoLTE. |
| Start | Stop | Start results in an Instantiated Configured Inactive NF Instance changing to an Instantiated Configured Active state that triggers it to perform its functionality. For example, able to accept messages from external Network Functions and sending responses.  Stop results in an Instantiated Configured Active VNF Instance starting a "shut-down". This may involve multiple steps, such as terminating jobs and processes or releasing allocated memory.  After the operation is fully executed, a VNF Instance is in an Instantiated Configured Inactive state, but ready to be started. |

|  |  |  |
| --- | --- | --- |
| **Transition Action** | **Reverse Transition Action** | **Description** |
| Scale out | Scale in | Scale out results in adding additional NF Service instances to a NF instance. New NF Service instances require new virtualization containers with compute, network, and storage capacity.  Scale in removes NF Service instances and their associated Kubernetes resources. |
| Scale-up | Scale-down | Scale up results in NF Service instances increasing the compute, network, and storage resources used. For example, replace a dual-core with a quadcore processor.  Scale down results in NF Service instances decreasing compute, network, and storage resources used. For example, replace a quad-core processor with a dual-core processor. |
| Update | Update Rollback | Update results in the deployment of fixes that do not create a new functionality. This includes corrections to existing functionality, such as bug fixes, as well as improvements to an existing functionality.  Update Rollback results in removing the changes (fixes and/ or improvements) provided by the update. |
| Upgrade | Upgrade Rollback | Upgrade results in the deployment of an enhanced version of an NF. After the operation is fully executed, an NF instance is based on a new version.  Upgrade Rollback results in returning the functionality of an NF to its previous version. |

|  |  |  |
| --- | --- | --- |
| **Transition Action** | **Reverse Transition Action** | **Description** |
| Reset | N/A | Reset sets an NF Instance that is currently in Instantiated Configured (Inactive, Active) into Instantiated Not Configured state. |

[Figure 16: NF States and State Transitions](#_bookmark96) provides a graphical overview of the NF, NF Service and NF Service Instance states and state transitions.

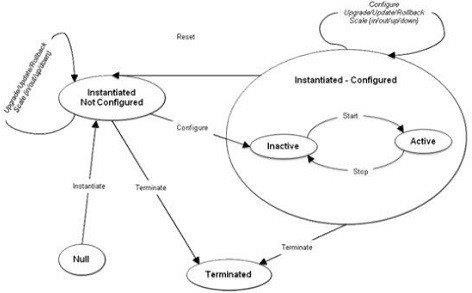


Figure 16: NF States and State Transitions

TMaaS correlates Kubernetes Pod statuses to ETSI states. A Pod instance corresponds to an NF Service Instance.

The NF Service instance state is derived from the Pod status as described below:

* Startup Probe is Success if the Pod.podStatus.containerStatuses[].startup flag is True for all Containers in that Pod.
* Readiness Probe is Success if the Pod.podStatus.containerStatuses[].ready flag is True for all Containers in that Pod.
* Pod Status is the value of the Pod.podStatus.type

[Table 15: NF Service Instance States and Actions](#_bookmark97) describes the NF Service Instance states and actions:

Table 15: NF Service Instance States and Actions

|  |  |  |  |
| --- | --- | --- | --- |
| **NF Service Instance State** | **Startup Probe** | **Readiness Probe** | **Pod Status** |
| Null | False | False | Pending |
| Action: None | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **NF Service Instance State** | **Startup Probe** | **Readiness Probe** | **Pod Status** |
| Instantiated Not Configured | True | False | Pending |
| Action:   * Create an NF Service Instance * Create an NF Service, if it does not exist already * Create an NF, if it does not exist already * Generate Events * managedObjectCreated for a new NF Service instance created * managedObjectModified for an NF Service and NF if these objects already exist. Else generate managedObjectCreated event. * managedObjectStateChanged event for the NF Service Instance, NF Service, and NF. | | |
| Instantiated Configured Inactive | True | False |  |
| Action: This state is not supported. | | |
| Instantiated Configured Active | True | True | Running |
| Action:   * Update an NF Service Instance state * Update the NF Service state * Create an NF state * Generate Events * managedObjectStateChanged event for the NF Service Instance, NF Service, and NF. | | |
| Terminated |  | False | Succeeded | Failed |
| Action:   * Delete an NF Service Instance * Update the NF Service state * Create an NF state * Generate Events * managedObjectDeleted for the deleted NF Service instance * managedObjectModified for the NF Service and NF * managedObjectStateChanged event for the NF Service and NF. | | |

## TMaaS Chart Specification

The following annotations can be specified as part of pod\_metaspec:

* *xgvela.com/tmaas.nf.nfs.nfsi.minReady: <integer>* Denotes the number of pods that must be ready for an NFService to be declared as INSTANTIATED\_CONFIGURED\_ACTIVE. This is optional. The default policy is min = number of NF services deployed.

Example values.yaml:

componentSpec:

deployment:

replicas: 2 imagePullPolicy: "Always" pod\_metaspec:

labels:

app: test-app annotations:

sidecar.istio.io/inject: "false" prometheus.io/scrape: "true" prometheus.io/path: /metrics prometheus.io/port: "6060"

xgvela.com/tmaas.nf.nfs.nfsi.minReady: 2 # 2 active pods are needed for NFService to be active

## TMaaS Events

The following table describes TMaaS events:

Table 16: TMaaS Events

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Local Event Name** | **VES Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| **NFServiceInstance Events** | | | | | | |
| NFServiceInstanceCreated | Notification\_XGVela-Maven FServiceInstanceCreated | INFO  ir\_N- | TOPOLOGY | N/A | This event is generated when an NF service instance is created in the topology. | N/A |
| NFServiceInstanceStateChanged | Notification\_XGVela-Maven FServiceInstanceStateChanged | INFO  ir\_N- | TOPOLOGY | N/A | This event is generated when an NF service instance state change happens in the topology.  The following are the change types: | N/A |
|  |  |  |  |  | * state: Normal state transition. The following are the states for this change type: |  |
|  |  |  |  |  | * NULL |  |
|  |  |  |  |  | * READY |  |
|  |  |  |  |  | * NOT\_READY |  |
| NFServiceInstanceDeleted | Notification\_XGVela-Maven FServiceInstanceDeleted | INFO  ir\_N- | TOPOLOGY | N/A | This event is generated when an NF service instance is deleted from the topology. | N/A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Local Event Name** | **VES Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| **NFService Events** | | | | | | |
| NFServiceCreated | Notification\_XGVela-Maven FServiceCreated | INFO  ir\_N- | TOPOLOGY | N/A | This event is generated when an NF service is created in the topology. | N/A |
| NFServiceStateChanged | Notification\_XGVela-Maven FServiceStateChanged | INFO  ir\_N- | TOPOLOGY | N/A | This event is generated when an NF service state change happens in the topology. | N/A |
|  |  |  |  |  | The following are the possible values of states: |  |
|  |  |  |  |  | * NULL |  |
|  |  |  |  |  | * INSTANTIATED\_NOT\_CONFIGURED |  |
|  |  |  |  |  | * INSTANTIATED\_CONFIGURED\_INACTIVE |  |
|  |  |  |  |  | * INSTANTIATED\_CONFIGURED\_ACTIVE |  |
|  |  |  |  |  | * TERMINATED |  |
| NFServiceDeleted | Notification\_XGVela-Maven FServiceDeleted | INFO  ir\_N- | TOPOLOGY | N/A | This event is generated when an NF service is deleted from the topology. | N/A |
| NFServiceUpgradeStarted | Notification\_XGVela-Maven FServiceUpgradeStarted | INFO  ir\_N- | TOPOLOGY | N/A | This event is generated when NF Service upgrade starts. | N/A |
| NFServiceUpgradeCompleted | Notification\_XGVela-Maven  NFServiceUpgradeCompleted | INFO  ir\_ | TOPOLOGY | N/A | This event is generated when NF Service upgrade is completed. | N/A |
| **NF Events** | | | | | | |
| NetworkFunctionCreated | Notification\_XGVela-Maven workFunctionCreated | INFO  ir\_Net- | TOPOLOGY | N/A | This event is generated when an NF is created in the topology. | N/A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Local Event Name** | **VES Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| NetworkFunctionStateChanged | Notification\_XGVela-Maven workFunctionStateChanged | INFO  ir\_Net- | TOPOLOGY | N/A | This event is generated when an NF state change happens in the topology. | N/A |
|  |  |  |  |  | The following are the possible values of states: |  |
|  |  |  |  |  | * NULL |  |
|  |  |  |  |  | * INSTANTIATED\_NOT\_CONFIGURED |  |
|  |  |  |  |  | * INSTANTIATED\_CONFIGURED\_INACTIVE |  |
|  |  |  |  |  | * INSTANTIATED\_CONFIGURED\_ACTIVE |  |
|  |  |  |  |  | * TERMINATED |  |
| NetworkFunctionDeleted | Notification\_XGVela-Maven workFunctionDeleted | INFO  ir\_Net- | TOPOLOGY | N/A | This event is generated when an NF is deleted from the topology. | N/A |
| NetworkFunctionUpgradeStarted | Notification\_XGVela-Maven workFunctionUpgradeStarted | INFO  ir\_Net- | TOPOLOGY | N/A | This event is generated when NF upgrade starts. | N/A |
| NetworkFunctionUpgradeCompleted | Notification\_XGVela-Maven workFunctionUpgradeCompleted | INFO  ir\_Net- | TOPOLOGY | N/A | This event is generated when NF upgrade is completed. | N/A |

**Note:** The NFServiceInstanceCreated, NFServiceInstanceStateChanged, and NFServiceInstanceDeleted events have pod and node labels prefixed with *pod.label.* or *node.label.* respectively, present as part of additionalInfo. For sample events refer to [Annexure A](#_bookmark232).

The following table describes TMaaS operation related events:

Table 17: TMaaS Operation Related Events

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| OperationsSpecModified | INFO | OPERATION | N/A | This event is generated when an Operations Specification on an NF is created, modified or deleted. This event carries all the common and base properties including the ManagedObject indicating the NF object instance on which the Operations specification was updated. | N/A |
| OperationStateChanged | INFO | OPERATION | N/A | This event is generated upon operation state change. This event is of type state notification. | N/A |
|  |  |  |  | The event has the following additional properties: |  |
|  |  |  |  | * transactionId |  |
|  |  |  |  | * operationId |  |
|  |  |  |  | * target |  |
|  |  |  |  | * createTime |  |
|  |  |  |  | * updateTime |  |
|  |  |  |  | * status |  |

## TMaaS Interfaces

TMaaS provides an interface to query the topology using the GET operation over NETCONF. Refer to [References](#_bookmark2) for the location of the current version of the YANG model.

A sample RPC request is as follows:

<get>

<filter>

<ManagedElement xmlns="urn:rdns:com:mavenir:mwp: mcf-nrm-managed-element"/>

</filter>

</get>

TMaas also provides REST interfaces that provide the following functionalities:

* [Operation Specification Registration](#_bookmark103)

## Operation Specification Registration

This interface is used by microservices to register API specifications with TMaaS.

**URI**: /api/v1/\_operations/specification/register

**Method**: POST

**Request Body**: ApenAPI-3.0 compliant JSON request body

Table 18: Registration Operation Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Value** |
| specifica- | Json-ob- | JSON object body must conform to mcf-nrm-operations-type:specificationgrp |
| tion | ject |  |
|  |  | { |
|  |  | "openapi"": {...}, |
|  |  | "info": {...}, |
|  |  | "servers": [...], |
|  |  | "paths": [...], |
|  |  | "tags": [...] |
|  |  | } |

Response:

200: OK

204: No Content

304: Not Modified

400: Bad Request

# LMaaS

The Log Management as a Service (LMaaS) feature of XGVela enables applications deployed in Kubernetes to manage logs in a distributed model. LMaaS feature is included as part of XGVela CIM. XGVela CIM provides the application microservices with the open API to integrate with LMaaS. Microservices stream logs to the CIM container over NATS.

CIM writes the logs to a file, which is read by Fluent Bit and stored in Elasticsearch. Fluent Bit and Elasticsearch are part of the PaaS layer. Fluent Bit supports output adaptors to forward the logs to external systems for long term storage and analytics.

You can use Kibana, which is available as part of the PaaS layer, to visualize logs data using Elasticsearch.

[Figure 17: LMaaS Design](#_bookmark106) describes the LMaaS design:

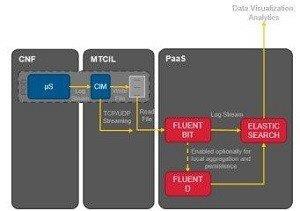


Figure 17: LMaaS Design

Refer to the following sections:

###### [Packaging Model for LMaaS](#_bookmark107)

###### [LMaaS Events](#_bookmark109)

###### [LMaaS Metrics](#_bookmark111)

###### [LMaaS Interface](#_bookmark113)

## Packaging Model for LMaaS

The LMaaS feature is enabled by default as part of CIM configuration. CIM allows applications to exclude the *cim.json* file from the chart. The default values can be changed during XGVela registration. Refer to [CIM Beta Version Enhancements](#_bookmark71) for details.

You can control the behavior of LMaaS functionality by changing the parameters mentioned in [Table 19:](#_bookmark108) [LMaaS Parameter Details](#_bookmark108) from the cim.json file. Any changes to the Init-time parameters during runtime results in restart of the application POD.

{

"cimConfig": {

"lmaas": {

"max\_file\_size\_in\_mb": 10,

"max\_backup\_files": 10,

"max\_age": 7,

"buffer\_size": 10000,

"flush\_timeout": 30, "logging\_mode": "FILE",

"prefer\_local\_daemonset": false

},

………………………………………….

"app\_settings": {

"app\_port": 9999, "http2\_enabled": false

}

[Table 19: LMaaS Parameter Details](#_bookmark108) describes the LMaaS parameters:

Table 19: LMaaS Parameter Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Name** | **Category** | **Range** | **Description** |
| max\_file\_size\_in\_mb | Runtime | 1-50 | Indicates the maximum file size to truncate.  **Unit** : MB  **Default Value** : 10 |
| max\_backup\_files | Runtime | 1-20 | Indicates the maximum number of backup files to keep.  **Default Value** : 10 |
| max\_age | Runtime | 1-30 | Indicates the maximum time to keep backup file.  **Unit** : Days  **Default Value** : 7 |
| buffer\_size | Runtime | 1000-100000 | Indicates the buffer size before writing to file. This is used for performance enhancement.  **Unit** : bytes  **Default Value** : 10000 |
| flush\_timeout | Runtime | 5-30 | Indicates the buffer flush to file timeout  **Unit** : Seconds  **Default Value** : 30 |
| logging\_mode | Init-time | * FILE * TCP * STDOUT * FILEANDSTDOUT | Indicates the logging mode. Select FILE for file-based logging, TCP for TCP-based logging, STDOUT to stream logs over console, and FILEANDSTDOUT for file-based and console-based logging.  **Default Value** : FILE |

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Name** | **Category** | **Range** | **Description** |
| prefer\_local\_daemonset | Init-time | * true * false | Indicates whether the POD sends data to the local daemonset.  Used for further performance improvement on top of TCP logging.This is applicable only if TCP logging is enabled. Enabling prefer\_local\_daemonset ensures that the logs are sent directly to the daemonset running locally on the same worker node, instead of routing it through the service. |

**Note:** To use the prefer\_local\_daemonset feature, ensure that the following env variable is defined for the CIM container as part of the deployment.yaml file:

{

"cimConfig": {

"lmaas": {

"max\_file\_size\_in\_mb": 10,

"max\_backup\_files": 10,

"max\_age": 7,

"buffer\_size": 10000,

"flush\_timeout": 30, "logging\_mode": "FILE", "prefer\_local\_daemonset": false

},

………………………………………….

"app\_settings": {

"app\_port": 9999, "http2\_enabled": false

}

## LMaaS Events

[Table 20: LMaaS Events](#_bookmark110) describes the LMaaS events:

Table 20: LMaaS Events

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Repair Actions** | **Clear Event** | **Description** |
| LmaasTCPFail- | MAJOR | COMMUNI- | Ensure that flu- | N/A | Indicates the |
| ure |  | CATION | ent-bit service |  | event raised by |
|  |  |  | is up and run- |  | CIM when the |
|  |  |  | ning with TCP |  | TCP connection |
|  |  |  | port open for |  | goes down. |
|  |  |  | logging. |  |  |

## LMaaS Metrics

[Table 21: LMaaS Metrics](#_bookmark112) describes the LMaaS metrics:

Table 21: LMaaS Metrics

|  |  |  |
| --- | --- | --- |
| **Metric Name** | **Metric Type** | **Description** |
| cim\_total\_messages\_counter | Counter | Indicates the total number of messages received by CIM. |
| cim\_total\_bytes\_counter | Counter | Indicates the total number of bytes received by CIM. |
| cim\_packet\_size | Gauge | Indicates the current log size in bytes. |
| tcp\_disconnect\_count | Counter | Indicates the total number of TCP connection failures. |

## LMaaS Interface

To integrate with the LMaaS feature, NFs must send the logs to CIM over NATS using the following interface details:

Type: Flatbuffer Schema:

namespace LogInterface; table Log {

container\_id:string; container\_name:string; payload:string;

}

root\_type Log;

NFs must send logs to CIM over NATs on the LOG topic. The payload must follow the following format:

[timestamp][loglevel][correlationId]-logstring

The container\_id and container\_name are the ID and name of the application container sending logs.

**Note:** The NATs URL is nats://localhost:4222

# CMaaS

XGVela uses Configuration Management as a Service (CMaaS) to manage configurations for NFs or microservices deployed using Kubernetes.

[Figure 18: CMaaS Design](#_bookmark115) describes the CMaaS design:



Figure 18: CMaaS Design

CMaaS supports NetConf for configuration management over the NBI. Microservice and NF configuration schema is described in YANG and data is encoded as JSON. The NF or microservice deployment data is packaged as a Helm chart. During deployment, the Helm charts are used to create the necessary artifacts for the functioning of the NFs or microservices.

Configurations for microservices are available as files within the POD. These are injected using the Kubernetes configMaps.

Each NF or microservice which uses CMaaS to manage configurations must have its own chart based on the specification mentioned in [Packaging Model for Configuration Management](#_bookmark116).

When you change the configurations for an NF or microservice over NetConf, CMaaS prepares a JSON patch which contains the configuration updates and delivers it to the corresponding NF or microservices. CIM plays an important role in this process, by delivering the changeset data to the NF. After applying the JSON patch, the NF or microservice generates and publishes a config event to CIM using NATs. For more details about the updateConfig API, refer to [updateConfig](#_bookmark124).

The CMaaS audit mechanism checks whether the changes were successfully applied. If the audit fails, CMaaS reinitiates the NFs or microservices by restarting the pod.

Refer to the following sections for more details:

###### [Packaging Model for Configuration Management](#_bookmark116)

###### [Update Policies](#_bookmark118)

###### [Managing Dependencies](#_bookmark119)

###### [Application Upgrade](#_bookmark120)

###### [Day 1 Configuration](#_bookmark121)

###### [CMaaS Interfaces](#_bookmark123)

###### [NBI Integration](#_bookmark128)

###### [CMaaS Metrics](#_bookmark132)

###### [CMaaS Events](#_bookmark134)

## Packaging Model for Configuration Management

Use the latest the chart template for building NF charts to support the configuration management feature of XGVela. Refer to [CMaaS](#_bookmark114) for more information about charts.

To use the configuration management feature, the NF chart must contain the following files and folders:

* **config\mgmt** : This folder must be present in the NF chart.
* **<filename-x>.yang** : Specifies the YANG data model for the corresponding data set defined with key

<filename-x>.json. You can define supporting YANG files in different files and add the files with the

<filename>.yang key format. This file must be present in the config\mgmt\ folder.

* **<filename-x>.json** : Defines JSON data set for creating the YANG data model which is identified by

<filename-x>.yang. The filename must match with the master yang model filename defined using the

<filename-x>.yang key format. This file must be present in the config\mgmt\ folder.

* **updatePolicy.json** Defines the update policy for each data set defined using <filename-x>.json. The data is encoded in JSON format using name and value pair. For each dataset, the name indicates the filename without the JSON extension, and the value indicates whether the update policy is Restart or Dynamic. This file must be present in the config\mgmt\ folder.
* **dependency.json** Defines the dependent microservice names for each data set specified in the NF level configurations. This is applicable only for NF level configurations. This file must be present in the config\mgmt\ folder.
* **values.yaml** Enables controlling the support for specific configuration categories. Configmaps are categorized as mgmt\_cfg, static\_cfg env\_cfg. Configurations which can be managed over northbound interfaces (NBI) must be kept under mgmt\_cfg. Set the following in the values.yaml file to enable the configuration management feature for the NF:
* Set the create flag to true
* Set the configMgmt annotation to enabled

The example shows a sample values.yaml file with the configuration management feature enabled.

nf:

name: amf type: appln

vendor: mvnrtest cnfId: 5gc cnftpl\_configmap:

mgmt\_cfg:

create: true metaspec:

annotations:

configMgmt: enabled

static\_cfg:

create: false metaspec: {}

env\_cfg:

create: false metaspec: {}

### CIM Configurations

The following environment variables must be added to the CIM container section in the application’s deployment.yml for CIM to enable Day 1 and Day 2 (dynamic updates) configuration management.

**Note:** This is applicable for chart template versions before 2.9

* name: MS\_CONFIG\_REVISION

valueFrom:

configMapKeyRef:

name: {{ $cnfHdr.nfVariables.svcname }}-{{

$cnfHdr.nfVariables.svcVersion }}-mgmt-cfg key: revision

* name: NF\_CONFIG\_REVISION

valueFrom:

configMapKeyRef:

name: <nf\_name>-<NF\_version>-mgmt-cfg key: revision

When using chart template v2.9 onwards, the environment variables should be as follows:

* name: MS\_CONFIG\_REVISION

valueFrom: configMapKeyRef:

name: {{ $cnfHdr.nfVariables.svcname }}-mgmt-cfg key: revision

* name: NF\_CONFIG\_REVISION

valueFrom: configMapKeyRef:

name: <nf\_name>-mgmt-cfg key: revision

## Update Policies

CMaaS allows you to define update policies at an NF or microservice level using the updatePolicy.json file. These update policies allow you to define how CMaaS deals with configuration updates. CMaaS provides the following update policies:

* **Restart** : CMaaS restarts the pod containing the NFs or microservices to apply configuration updates. The updated configuration is loaded when the pods are restarted.
* **Dynamic** : CMaaS dynamically updates the configuration and delivers the changeset to the NF.

The following is a sample updatePolicy.json file for an NF with YANG and JSON files named vcmScpConfig.yang, vcmScpConfig.json, cim.yang, and cim.json:

updatePolicy.json

{

"vcmScpConfig": "Dynamic", "cim": "Dynamic"

}

CMaaS also allows you to specify init-time parameters (IT parameters) which require a restart of the pod containing the NFs or microservices for applying configuration updates. The restartConfigUpdate parameter call point must be mentioned for such parameters. If config updates are detected for parameters with the restartConfigUpdate call point, CMaaS restarts the pod containing the NFs or microservices even if the NF level update policy is set to Dynamic.

You must use the configUpdate parameter call point to specify run-time parameters (RT parameters) which can be updated without restarting the pod.

The following snippet from the YANG file shows a sample usage of IT parameters and RT parameters:

container config {

tailf:callpoint configUpdate

{ tailf:transaction-hook subtree; } list cdr {

key cdr\_id; leaf cdr\_id

{ type string; mandatory true; } leaf cdr\_type {

tailf:callpoint restartConfigUpdate

{ tailf:transaction-hook node; } type string;

mandatory true;

}

}

}

## Managing Dependencies

You can specify multiple microservices that share common configurations at the NF level. CMaaS allows you to define the dependent microservices for the common configuration using the dependency.json file. It checks the dependencies and updates the common configuration in the microservices that share the configurations.

The following is an example of a dependency.json file.

{

"amf": ["amfgw","amfee"]

}

This example specifies that CMaaS must update the configurations for the microservices amfgw and amfee when a configuration update is done for amf.

## Application Upgrade

For CMaaS to support application upgrade (using Helm upgrade), the NF chart of the application must be upgraded to use chart template v2.9 or later, and the nf.version value in the *values.yaml* of the chart must be changed if there is any change in the management configuration files kept under the config/mgmt folder.

The following are applicable if a microservice chart is revised:

* If the microservice chart has no changes to its management configuration files kept under config/mgmt folder, CMaaS ensures that JSON data (DAY\_1/DAY\_2) already present in configmap is not overriden by default JSON data (DAY\_0) present in the chart after upgrade.
* If the microservice chart has a new set of YANG files being added, CMaaS ensures that the new YANG files are compiled and provided after upgrade. DAY\_0 data is loaded into ConfD to the new namespaces. It is advised to also update *updatePolicy.json* and *dependency.json*, if required, when adding the new configuration.
* If the microservice chart has a revised YANG file (Example, 'config.yang' revised to 'con-

fig@10-11-2020.yang'), CMaaS ensures that revised YANG file is compiled again and loaded with the existing JSON data in the configmap after upgrade.

* + The revisioned YANG file must have a different namespace ("urn:mavenir:xgvela:testapp" revised to "urn:mavenir:xgvela:testapp:rev1").
  + The newly added fields must have default values or should not be mandatory fields.

CMaaS generates the *NFMgmtIntfChanged* event upon successful upgrade of NF, similar to *NFMgmtIntfReady* event upon NF installation. If there is any data-load, compilation, or schema load errors, this event is not sent.

## Day 1 Configuration

The configuration which is provided along with the charts during deployment of an NF is called the Day 0 configuration. Day 0 configurations are the minimum set of configurations required by the NF and corresponding microservices to go to the initialized state. Some NFs may require additional configurations to reach the in service state where it can serve the network traffic. These additional configurations which are required for NFs to reach the in service state are called Day 1 configurations.

**Note:** The Day 1 configuration is expected to be part of NF level configurations, and dependency configuration must be used to associate the Day1 configuration to the microservices. In case of an NF which contains only microservice, you must define the chart hierarchy using the NF hierarchy model with a single sub-chart.

The following steps describe the Day 1 configuration process:

1. When an NF is deployed, TMaaS emits the NetworkFunctionCreated event.
2. CMaaS emits the NFMgmtIntfReady event to indicate that the management interface is ready for the NF to accept configuration changes. Refer to [NFMgmtIntfReady](#_bookmark239) for an example of the NFMgmtIntfReady event.
3. The NF checks whether it has all the configurations required to reach the ready state, and emits the Configured event if it has all the required configurations.
4. If the NF requires additional configurations, the NF emits a Not Configured event. CIM checks whether the full configuration is available.
5. If the full configuration is available, CIM sends the full configuration to the NF using the Load Config API.
6. After loading the configuration, the NF repeats steps 4 to 6 till the full configuration is loaded.
7. The NF reaches the Ready state. CMaaS detects that the Day 1 configuration is complete, and treats any further configuration changes coming from the management interface as a dynamic configuration update.

### Full Config On Restart

If an application pod is terminated, and the upcoming pod has to load its full configuration using the Load Full Config API, include the annotation "fullConfigOnRestart: enabled" as part of the chart in the NF-level values.yaml file.

**Note:** This flag is supported from CIM v1.10.2 onwards. The flag must not be included or set to disabled when using CIM version 1.10.1 and earlier.

Sample:

cnftpl\_configmap:

mgmt\_cfg:

create: true metaspec:

annotations:

configMgmt: enabled fullConfigOnRestart: enabled

## CMaaS Interfaces

Refer to the following section for details:

* [updateConfig](#_bookmark124)
* [Load Full Config](#_bookmark126)

### updateConfig

This interface is used by CIM to notify configuration changes.

**Note:** : “app\_port” must be set correctly in config.json for CIM (”app\_port”:”9999”). The app\_port is used while sending the request to NFs or microservices.

Parameters

Table 22: updateConfig API Parameters

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| change-set-key | Specifies the changeset key.  Example: “change-set-key”: “change-set/mvnr-embb/mobilitymgr/amf-1.0/5” |
| data-key | Specifies the data key. Example: “data-key”: “amf-1.0” |
| config-patch | Specifies details about the configuration updates to be done.  Example: “config-patch”: “[{“op”: “replace”, “path”: “/ amfconfig/loglevel”, “value”: “DEBUG”}]”  This example indicates that the loglevel value must be replaced with the value DEBUG. |
| revision | Specifies the revision number. Example: “revision”: “5” |

Request Body

{

“change-set-key”: string, “data-key”: string, “config-patch”: string, “revision”: string

}

Response Body

{

“change-set-key”: string, “data-key”: string, “config-patch”: string, “revision”: string

}

After applying the configuration change, NFs must generate and publish a config event to CIM over NATS using the following interface details:

**Interface** : NATS **Topic** : CONFIG **Request** :

Type: JSON Schema:

{

“change-set-key”: string, “revision”: string,

“status” : string, # success/failure/unused “remarks”: string

}

* change-set-key: As received in the configuration update request
* revision: As received in the configuration update request
* status: Status of config update, success/failure/unused as a string. The unused status code is used if the configuration change is not relevant for the NF.
* remarks: Any specific remarks about the config update status. In case of failure, reason for failure must be mentioned.

### Load Full Config

This API is used for loading the initial configuration (day1 configuration) which is mandatory for the NF to be ready, and is not part of the default deployment. This API must be implemented at the NF side so that CIM can use the API to provide the full configuration data to the NFs.

**Interface** : REST

**URI** : /api/v1/\_operations/loadConfig

**Method** : POST

**HTTP Port** : Configurable **HTTP Version** : 1.1/2 **ContentType** : application/json **Request Body Format**

{

“<dataKey-1>”: “<dataValue-1>”,

...

"<dataKey-n>": "<dataValue-n>"

}

Example Request Body

{

"amfConfig.json":"{"config": {"amfName": "amf1","amfRegionId": "50","amfSetId": "512"}}",

"nfPeerConfig.json":"{"config": {"nssf": [{"mcc": "401","mnc":

"101","fqdn": "nssf1.webscale.com","ipv4": ""},

{"mcc": "402","mnc": "102","fqdn": "nssf2.webscale.com","ipv4": ""}]}}"

}

[Table 23: NF Events](#_bookmark127) describes the events generated by NFs based on the configuration status:

Table 23: NF Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Interface** | **Topic** | **Request** | **Comments** |
| Configured Event Notification | NATS | EVENT | Event Notification Framework | event\_name: Configured |
| Not Configured Event Notification | NATS | EVENT | Event Notification Framework | event\_name: NotConfigured |

These event definitions are available as a part of CIM eventdefs which are available in the XGVela SDK.

## NBI Integration

This section describes Northbound Interface (NBI) integration for CMaaS. Refer to the following sections:

* [NF Onboarding and Configuration Management](#_bookmark129)
* [CMaaS NBI Details Exchange](#_bookmark130)
* [Configuration Update (Day-1/Day-2)](#_bookmark131)

### NF Onboarding and Configuration Management

During NF deployment, XGVela CMaaS onboards corresponding YANG models and loads the Day-0 configuration data. During the onboarding process, CMaaS updates the YANG namespace and YANG prefix to append the nfId to maintain different datasets per NF instance.

**Note:** The nfId corresponds to the nfId defined in the NF package *values.yaml*. The nfId is a global unique identifier for an NF instance. It is specified during NF deployment using the deployment *values.yaml*.

<namespace>:<nfId>

<prefix>-<nfId>

Example:

Actual namespace in NF package:

namespace urn:com:mavenir:\_5gc:yang:mavenir-smf; prefix mavenirsmf;

CMaaS modified namespace/prefix for nfId smf1:

namespace urn:com:mavenir:\_5gc:yang:mavenir-smf:smf1; prefix mavenirsmf-smf1;

### CMaaS NBI Details Exchange

The NBI details for XGVela integration including CMaaS is exchanged through the additionalFIelds of HeartBeat events. Refer to [heartbeat\_xgvela1](#_bookmark238) for a sample.

The following are the additionalFields:

"heartbeatFields":{

"additionalFields":{

"ME\_ID":"47269bda-c715-3892-acd4-6941707705d4", "MGMT\_IP":"10.x.x.x", "CMAAS\_NETCONF\_PORT":"32767", "APIGW\_PORT":"32640", "FMAAS\_HTTP\_PORT":"32666", "APIGW\_USERNAME":"admin", "APIGW\_PASSWORD":"admin", "APIGW\_AUTH\_TYPE":"basic",

"COLLECTOR\_ID":"2"

},

"heartbeatFieldsVersion":"3.0", "heartbeatInterval":60

}

MGMT\_IP, CMAAS\_NETCONF\_PORT, APIGW\_USERNAME, and APIGW\_PASSWORD are used for

Netconf communication towards CMaaS.

### Configuration Update (Day-1/Day-2)

Upon completion of the NF onboarding process, CMaaS generates the NFMgmtIntfReady event to indicate that the management interface is ready for the NF to accept configuration changes. After receiving the NFMgmtIntfReady event, NBIs can trigger the config push.

**Note:** During multiple NF deployment, CMaaS may move to the upgrade mode during NF onboarding process, and hence config push has to be retried in case if it fails indicating so.

Refer to [Day 1 Configuration](#_bookmark121) for more information about Day 1 configuration. Refer to [NFMgmtIntfReady](#_bookmark239) for an example of the NFMgmtIntfReady event.

The following are the additionalFields:

"notificationFields": { "additionalFields": {

"nfId": "abddbb90-8290-3198-8780-ccff7c977be3", "meId": "07adb663-4387-3544-bf8b-35ea1324534c",

"meLabel": "mvnr2,ManagedElement=me-xgvela1",

"nfLabel": "mvnr2,ManagedElement=me-xgvela1,NetworkFunction=udsf1", "nfType": "udsf",

"nfSwVersion": "v0",

},

}

Config push procedure for Netconf

1. Copy the configuration from running to candidate as follows:

<copy-config>

<target>

<candidate/>

</target>

<source>

<running/>

</source>

</copy-config>

1. Lock the candidate as follows:

<lock>

<target>

<candidate/>

</target>

</lock>

1. Edit the configuration as follows:

<edit-config>

<target>

<candidate/>

</target>

<default-operation>replace</default-operation>

<config>

<data xmlns="urn:mavenir:ran:mavenir\_data\_contain- er:5\_0\_1\_201:du-4">

...

</data>

</config>

<edit-config>

Note:

During Config push, ensure that the the nfID is suffixed to the applicable namespace. Contents within the <config> tag are for illustration purpose only.

In the above snippet, the namespace section "urn:mavenir:ran:mavenir\_data\_container: 5\_0\_1\_201" is the namespace as per the YANG model to which :du-4 is suffixed during config push.

1. Commit the changes as follows:

<commit/>

1. Unlock the candidate as follows:

<unlock>

<target>

<candidate/>

</target>

</unlock>

**Note:** Ensure that configuration changes intended to be pushed as part of a config update process which includes delete operaitons are done in a single transaction.

## CMaaS Metrics

[Table 24: CMaaS Metrics](#_bookmark133) describes the metrics supported by CMaaS:

Table 24: CMaaS Metrics

|  |  |  |
| --- | --- | --- |
| **Metric** | **Metric Type** | **Description** |
| cmaas\_operations\_requests\_total | Counter | Indicates the total number of operations requests received by CMaaS. Include the following labels:   * sourceIP: IP address of the client sending the request. * operationId * action {show, exec, get, etc} |
| cmaas\_operations\_failure\_total | Counter | Indicates the total number of operation requests for which CMaaS sent an error response.   * sourceIP: IP address of the client sending the request * operationId * action {show, exec, get, etc} * responseCode |
| cmaas\_configmodel\_onboard\_attempts\_total | Counter | Indicates the number of YANG files that CMaaS has tried to compile. |
| cmaas\_configmodel\_onboard\_failure\_total | Counter | Indicates the number of YANG files that have failed to compile. |
| cmaas\_configdata\_load\_attempts\_total | Counter | Indicates the number of times a REST call has been made to load configuration data into a ConfD namespace. |

|  |  |  |
| --- | --- | --- |
| **Metric** | **Metric Type** | **Description** |
| cmaas\_configdata\_load\_failure\_total | Counter | Indicates the number of times a REST call has been made to load configuration data into a ConfD namespace, and failed to do so successfully. |
| cmaas\_configdata\_update\_attempts\_total | Counter | Indicates the number of transaction callbacks CMaaS has received from ConfD. |
| cmaas\_configmap\_update\_attempts\_total | Counter | Indicates the number of times that CMaaS has attempted to update the ConfigMap. |
| cmaas\_configmap\_update\_failure\_total | Counter | Indicates the number of times that CMaaS has failed to update the ConfigMap. |
| cmaas\_audit\_changeset\_push\_attempted\_total | Counter | Indicates the number of change-set push attempts. |
| cmaas\_audit\_changeset\_push\_failure\_total | Counter | Indicates the number of change-set push failures. |
| cmaas\_appln\_commit\_config\_success\_total | Counter | Indicates the number of audits that succeeded. That is, the number of times that the pods concerned with the configuration change successfully applied the configuration. |
| cmaas\_appln\_commit\_config\_failure\_total | Counter | Indicates the number of audits that failed. That is, the number of times that the pods concerned with the configuration change failed to apply the configuration. |
| cim\_config\_push\_attempt \_total | Counter | Indicates the total number of config push attempts. |

|  |  |  |
| --- | --- | --- |
| **Metric** | **Metric Type** | **Description** |
| cim\_config\_push\_failure\_ total | Counter | Indicates the total number of config push failures. |
| cim\_config\_resp\_received\_total | Counter | Indicates the total number of config responses. |
| cim\_commit\_config\_push\_failure\_total | Counter | Indicates the total number of commit config push failures. |
| cim\_config\_resp\_success\_total | Counter | Indicates the total number of successful config responses. |
| cim\_config\_resp\_failure\_total | Counter | Indicates the total number of failure config responses. |

## CMaaS Events

[Table 25: CMaaS Events](#_bookmark135) describes the CMaaS events:

Table 25: CMaaS Events

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| NFMgmtIntfReady | INFO | LOAD | N/A | Indicates that the management interface is ready for a particular NF to accept configuration changes. | N/A |
| CmaasDataProviderFailure | CRITICAL | COMMUNICATION | CmaasDataProviderFailure with CLEAR severity. | Indicates that the data provider connection has failed. | No specific action required. Con-  fig-service restarts self and recovers. |
| CmaasAuditFailure | INFO | AUDITING | N/A | Indicates audit failure for a pod during dynamic updates. | N/A |
| CmaasConfigUpdateReceived | INFO | UPDATE | N/A | Indicates that a transaction is committed successfully in ConfD. | N/A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| CmaasConfigmapUpdateSuccess | INFO | UPDATE | N/A | Indicates that the ConfigMap corresponding to a ConfD namespace is successfully updated. | N/A |
| CmaasChangeSetPushSuccess | INFO | AUDIT | N/A | Indicates that a change-set key is successfully pushed into Etcd for dynamic updates. | N/A |
| CmaasConfigDataLoadSuccess | INFO | LOAD | N/A | Indicates that a JSON configuration is successfully loaded to its ConfD namespace. | N/A |
| CmaasConfigDataLoadFailure | CRITICAL | LOAD | CmaasConfigDataLoadSuccess | Indicates a failure in loading a JSON configuration to its ConfD namespace. | Redeploy the NF with the correct configuration. |
| CmaasConfigDataLoadSuccess | CLEAR | CLEAR | N/A | Clears the CmaasConfigDataLoadFailure event. | N/A. |
| CmaasConfigModelCompileFailure | CRITICAL | COMPILE | CmaasConfigModelCompileSuccess | Indicates a failure in compiling a YANG mode | Redeploy the NF with the valid YANG-  model. |
| CmaasConfigModelCompileSuccess | CLEAR | CLEAR | N/A | Clears the CmaasConfigModelCompileFailure event. | N/A |
| CmaasSchemasLoadFailure | CRITICAL | LOAD | CmaasSchemasLoadSuccess | Indicates a failure in upgrading the ConfD schema container. | Redeploy the NF with the valid YANG-  model. |
| CmaasSchemasLoadSuccess | CLEAR | CLEAR | N/A | Clears the CmaasSchemasLoadFailure event. | N/A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| CmaasSelfConfigUpdateAttempted | INFO | LOAD | N/A | Indicates that CMaaS config update has started. | N/A |
| CmaasSelfConfigUpdateSuccess | INFO | CONFIG\_UPDATE | N/A | Indicates that CMaaS config update is completed successfully. | N/A |
| NFMgmtIntfChanged | INFO | LOAD | N/A | Indicates that the management interface is upgraded successfully for a particular NF. | N/A |

# MMaaS

Metrics Management as a Service (MMaaS) implements a collect and forward service for KPI metrics generated by microservices. MMaaS follows the Prometheus architecture and software for metrics management. The Prometheus instance is provisioned with NF specific alerting and aggregation rules.

The Metric Service Interface can be instrumented at the microservice level using one of the available Prometheus client libraries. If an NF wants to avoid library dependencies, it can implement a simple textbased exposition format.

[Figure 19: MMaaS Design](#_bookmark137) describes the design of MMaaS:



Figure 19: MMaaS Design

Refer to the following sections for details:

###### [Packaging Model for MMaaS](#_bookmark138)

###### [Enabling Scraping of Metrics](#_bookmark139)

###### [Custom Application Metrics](#_bookmark141)

###### [Defining Metrics](#_bookmark143)

###### [Sample Metrics](#_bookmark145)

###### [Prometheus Configuration](#_bookmark149)

###### [Metrics Exposition Format](#_bookmark150)

###### [Metric Types](#_bookmark151)

###### [Clock-Aligned Metrics](#_bookmark152)

###### [Metrics with Prometheus Operator](#_bookmark153)

###### [MMaaS Deployment Models](#_bookmark154)

###### [Prerequisites for Helm Charts](#_bookmark157)

###### [Multiple Metrics Remote Storage](#_bookmark166)

## Packaging Model for MMaaS

Use the latest chart template for building application charts to support the Metrics management feature of XGVela.

To use the metrics management feature, the NF chart must contain the following:

* **metrics/** : Folder containing the metrics.yml file. This folder is required to use the metrics management feature.
* **metrics.yml** : Allows you to define custom metric definitions for NFs. You can define multiple rules as per the requirement. Refer [Defining Metrics](#_bookmark143) for details.
* **values.yaml** : To enable metrics management, set the cnftpl\_metricsdef create flag to true in the values.yaml file.

cnftpl\_metricsdef:

create: true

## Enabling Scraping of Metrics

This section is applicable for NF scraping deployment with Prometheus (non-Prometheus operator based deployment). This section is applicable for the Cluster Scrape and NF Scrape deployment models.

**Note:** Refer to[Metrics with Prometheus Operator](#_bookmark153) for NF metrics scraping with Prometheus Operator.

You can enable scraping of metrics using the following ways:

* **Single Port Non Clock-Aligned Metrics** Enable the following annotations to enable scraping metrics in a pod for non clock-aligned metrics:

annotations:

prometheus.io/scrape: "true" prometheus.io/path: /metrics prometheus.io/port: <port-number> prometheus.io/scheme: http

**Note:** This is the default way for enabling scraping of metrics for a single port. By default, when CIM is the sidecar configure as per Multi-Container configuration specifications.

* **Multi-Container or Multi-Port Non Clock-Aligned Metrics** This is used in the following scenarios:
  + If pods have multiple containers and these containers expose ports for scraping metrics.
  + If a container exposes multiple ports for scraping metrics.
  + A combination of these scenariosEnable the following annotations to enable scraping metrics in a pod for non clock-aligned metrics:

**Note:** The prometheus.io/port annotation is not used.

annotations:

prometheus.io/scrape: "true" prometheus.io/path: /metrics prometheus.io/scheme: http

**Note:** All the ports to be scraped must be specified as the container port in the charts.Since CIM is used as the side car and exposes own metrics apart from Application metrics, this should be the default configuration. Sample Part of Charts Deployment Spec

ports:

- containerPort: 8095

protocol: TCP

For CIM Sidecar:

ports:

- containerPort: 6060 protocol: TCP

* **Clock-Aligned Metrics** Enable the following annotations to enable scraping metrics in a pod for clock- aligned metrics:

annotations:

prometheus.io/scrape: "true" prometheus.io/path: /metrics/aligned prometheus.io/port: <6060> prometheus.io/scheme: http xgvela.com/mmaas: '{

"scrape": "true",

"path": "/metrics",

"ports": [<app port1>, <app port2>], "scheme": "http", "measurementInterval": "15s",

"maxRetention": "20"

}

* **Multi-Container or Multi-Port support** - To scrape the metrics ports in multiple containers or multiple ports in single container specify the multiple ports (comma separated list in square bracket) in ports. Since CIM is used as the side car and exposes its own metrics apart from application metrics, this should be the default configuration.

**Note:** All the ports to be scraped must be specified as the container port in the charts and a name should be given to the port. The default name is "metrics"

Sample - Part of Charts Deployment Spec

ports:

- containerPort: 8095 protocol: TCP

For CIM Side Car

ports:

- containerPort: 6060 protocol: TCP

* Refer to [Clock-Aligned Metrics](#_bookmark152) for information about clock-aligned metrics.
* Refer to [Metrics with Prometheus Operator](#_bookmark153) for information about non clock-aligned and clock-aligned metrics with Prometheus Operator at PaaS,

[Table 26: MMaaS Annotation Structure for Clock Aligned Metrics](#_bookmark140) describes the MMaaS annotation structure:

Table 26: MMaaS Annotation Structure for Clock Aligned Metrics

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Description** | **Default Value** | **Sample Values** |
| scrape (M) | Indicates whether scraping by CIM is enabled. | false | true/false |
| path (M) | Indicates the metrics path for scraping. | /metrics/aligned | /metrics/aligned |
| ports (M) | Indicates the comma-separated metrics  port of the application for scraping. | Based on application endpoints. | ["9090"]or ["8085","9095"] |
|  | Examples: |  |  |
|  | Single endpoint: 9090 |  |  |
|  | Multiple endpoints: 8085,9095 |  |  |
| scheme (M) | Indicates the scheme. | http | http |
| measurementInterval (M) | Indicates the interval, in seconds, for scraping the metrics from the application. | 15s | 15s |
| maxRetention (M) | Indicates the maximum number of buckets to retain. | 20 | 20 |
| (M) = mandatory |  | | |

## Custom Application Metrics

[Table 27: Metric Fields](#_bookmark142) describes the metric fields:

Table 27: Metric Fields

|  |  |
| --- | --- |
| **Metric Field** | **Description** |
| Metric Name | Specifies the general feature of a system that is measured. |
| Labels | Enables the dimensional data model of Prometheus. Any given combination of labels for the same metric name identifies a particular dimensional instantiation of that metric. The labels are represented as a key-value array. You can create a metric without labels. |

|  |  |
| --- | --- |
| **Metric Field** | **Description** |
| Current Metric Value | Indicates the current metric value. |

NFs must expose a metrics endpoint for instrumenting code with Prometheus metrics. Microservices can include specific identifiers as labels in the metrics exposition data, which is scraped by MMaaS or Prometheus from the REST end-point exposed by each microservice container.

**Note:** MMaaS can only pass the labels, and the NFs can choose the value to send for these labels. MMaaS cannot send any additional properties.

The following example shows custom application metrics:

Metrics exposed by CU microservice

# HELP num\_active\_sessions Number of active sessions. # TYPE num\_active\_sessions counter

# HELP num\_requests\_sessions Total number of requests. # TYPE num\_active\_sessions counter

num\_requests\_total {cuId="gNB1"} 1111

num\_active\_sessions {cuId="gNB1"} 1027

## Defining Metrics

MMaaS allows you to define custom metrics rules for an NF using the metricss.yaml file. The following fields allow you to describe custom metrics:

metrics:

AdditionalObjects:

-

object\_name: object\_instance: expr: object\_keys:

- name: expr:

[Table 28: Field Descriptions](#_bookmark144) describes the fields:

Table 28: Field Descriptions

|  |  |
| --- | --- |
| **Field** | **Description** |
| *object\_name* | Indicates the metrics group name used to identify and group similar metrics.The identifier or name to array of object instances (JSON objects) is described by this name.  Examples:   * AdditionalCMaaSCommitCounters * CUMetrics * DUMetrics   **Note:** This field can be repeated to group similar metrics. |
| *object\_instance* | Indicates the name of the metric. The value of this metric is specified by the expr field.  Examples:   * CMaasCommitCounters * ActiveSessions * TotalRequest |
| *expr* | Indicates the PromQL expression giving the value for the metric.  Examples:  "rate (cmaas\_appln\_commit\_config\_success\_total [5m])"  "cu\_number\_live\_session {DUId=''}" "num\_total\_request{DUId!=''}" |
| object\_keys: | Indicates an ordered set of keys [keyName:keyValue] that identifies this instance of jsonObject.  **Note:** This is an array so multiple keys can be configured. |
| *name* | Indicates the name of the key. Examples:  Id DUId  The value associated with the key name is configured using *expr*. |

|  |  |
| --- | --- |
| **Field** | **Description** |
| *expr* | Indicates the value of the key.  The label expression is used to fetch specific labels from MMaaS to uniquely identify the object.  **Note:** This label must exist in the response from MMaaS.  Examples:   * *'{{.labels.Id}}'* * *'{{.labels.CPId}}'* * *'{{.labels.DUId}}' '{{.labels.CPId}}:{{.labels.DUId}}* |

## Sample Metrics

Refer to the following sections:

* [Metrics Exposed by DU Microservices](#_bookmark146)
* [VES Gateway Measurement Event Mapping for DU](#_bookmark147)
* [Generated Measurement VES Event for DU](#_bookmark148)

### Metrics Exposed by DU Microservices

# HELP num\_active\_sessions Number of active sessions. # TYPE num\_active\_sessions counter

# HELP num\_requests\_sessions Total number of requests. # TYPE num\_active\_sessions counter

num\_requests\_total {cuId="gNB1", duId="du1"} 1111

num\_active\_sessions {cuId="gNB1", duId="du1"} 1027

### VES Gateway Measurement Event Mapping for DU

**Note:** These metrics must be included as part of the NF chart.

./metrics/metricss.yml

::::::::::::::

metrics:

AdditionalObjects:

-

object\_name: DUMetrics

object\_instance: ActiveSessions expr: "num\_active\_session" object\_keys:

-

expr: "{{.labels.duId}}" name: DUID

-

expr: "{{.labels.cuId}}" name: CUID

-

object\_name: DUMetrics object\_instance: TotalRequests expr: "num\_requests\_total"

object\_keys:

-

expr: "{{.labels.duId}}" name: DUID

-

expr: "{{.labels.cuId}}" name: CUID

### Generated Measurement VES Event for DU

This section provides a sample of the Measurement VES Event for DU generated based on [Metrics Exposed by DU Microservices](#_bookmark146) and [VES Gateway Measurement Event Mapping for DU](#_bookmark147).

"measurementFields": {

"additionalObjects": [

{

"objectInstances": [

{

"objectInstance": { "ActiveSessions": 1027,

"TotalRequest": 1111

},

"objectKeys": [

{

"keyName": "CUID", "keyValue": "gNB1"

}

{

"keyName": "DUID",

"keyValue": "du1"

}

]

},

{

"objectInstance": { "ActiveSessions": 1027,

"TotalRequest": 1111

},

"objectKeys": [

{

"keyName": "CUID", "keyValue": "gNB2"

}

{

"keyName": "DUID",

"keyValue": "du2"

}

]

},

"objectName": "DUMetrics"

}

]

}

## Prometheus Configuration

Prometheus is configured using command-line flags and a configuration file. The configuration file defines scraping jobs and their instances, and specifies the rule files to load.

As part of Kubenetes service discovery and scraping, Prometheus supports the kubernetes\_sd\_config template. Kubernetes SD configurations allow retrieving scrape targets from the Kubernetes REST API. It always stays synchronized with the cluster state.

The following are the supported roles:

* Node
* Service
* Pod
* Endpoints
* Ingress

## Metrics Exposition Format

The following items describe the format for specifying metrics:

* **Text Format** : The Prometheus text-based format is line oriented. Lines are separated by a line feed character (\n). The last line must end with a line feed character. Empty lines are ignored.
* **Line Format** : Within a line, tokens can be separated by any number of blanks and tabs. The tokens must be separated by at least one blank or tab. It would otherwise merge with the previous token. Leading and trailing whitespaces are ignored.
* **Comments** : Lines with a # as the first non-whitespace character are comments. It is ignored unless the first token after # is either HELP or TYPE.
* **HELP** : If the token after #[“ “]\* is HELP, at least one more token is expected, which is the metric name. All remaining tokens are considered as the docstring for that metric name. HELP lines may contain any sequence of UTF-8 characters after the metric name, but the backslash and the line feed characters have to be escaped as \\ and \n, respectively. Only one HELP line may exist for any given metric name.
* **TYPE** : If the token after #[“ “]\* is TYPE, exactly two more tokens are expected. The first token is the metric name, and the second token is either counter, gauge, histogram, summary, or untyped, defining the type for the metric of that name. Only one TYPE line may exist for a given metric name. The TYPE line for a metric name must appear before the first sample is reported for that metric name. If there is no TYPE line for a metric name, the type is set to untyped. The remaining lines describe samples (one per line) using the following Extended Backus-Naur Form (EBNF) syntax:
* metric\_name [ "{" label\_name "=" `"` label\_value `"` { "," label\_name "=" `"` label\_value `"` } [ "," ] "}“ ] value [ timestamp ]
* metric\_name and label\_name have the Prometheus expression language restrictions.
* label\_value can be any sequence of UTF-8 characters, but the backslash (\, double-quote ("}, and line feed (\n) characters have to be escaped as \\, \", and \n, respectively.
* value is a float represented as required by the Go language ParseFloat() function. In addition to standard numerical values, Nan, +Inf, and -Inf are valid values representing not a number, positive infinity, and negative infinity, respectively.
* The timestamp is an int64 (milliseconds since epoch, i.e. 1970-01-01 00:00:00 UTC, excluding leap seconds), represented as required by the Go language ParseInt() function.

## Metric Types

MMaaS provides the following types of metrics:

* **Counter** : A counter is a cumulative metric that represents a single monotonically increasing counter. The value can only increase or be reset to zero on restart. For example, you can use a counter to represent the number of requests served, tasks completed, or errors.
* **Gauge** : A gauge is a metric that represents a single numerical value that can arbitrarily go up and down, like the number of concurrent requests.
* **Histogram** : A histogram samples observations such as request durations or response sizes, and counts them in configurable buckets. It also provides a sum of all observed values. A histogram with a base metric name of <basename> exposes multiple time series including the following during a scrape:
* Cumulative counters for the observation buckets, exposed as <basename>\_bucket{le="<upper inclusive bound>"}
* The total sum of all observed values, exposed as <basename>\_sum
* The count of events that have been observed, exposed as <basename>\_count (identical to <basename>\_bucket{le="+Inf"} )
* **Summary** : Similar to a histogram, a summary samples observations such as request durations and response sizes. While it also provides a total count of observations and a sum of all observed values, it calculates configurable quantiles over a sliding time window. A summary with a base metric name of

<basename> exposes multiple time series during a scrape:

* Streaming φ-quantiles (0 ≤ φ ≤ 1) of observed events, exposed as <basename>{quantile="<φ>"}
* The total sum of all observed values, exposed as <basename>\_sum
* The count of events that have been observed, exposed as <basename>\_count The following example shows how to define metrics, metric types, and samples:

# HELP http\_requests\_total The total number of HTTP requests.

http\_requests\_total{method="post",code="200"} 1027 1395066363000

http\_requests\_total{method="post",code="400"} 3 1395066363000

# Escaping in label values: msdos\_file\_access\_time\_seconds{path="C:\\DIR\\FILE.TXT",error="Canno t find

file:\n\"FILE.TXT\""} 1.458255915e9

# TYPE http\_requests\_total counter

# Minimalistic line: metric\_without\_timestamp\_and\_labels 12.47

# A histogram, which has a pretty complex representation in the text format:

# HELP http\_request\_duration\_seconds A histogram of the request duration.

# TYPE http\_request\_duration\_seconds histogram

http\_request\_duration\_seconds\_bucket{le="0.05"} 24054

http\_request\_duration\_seconds\_bucket{le="0.1"} 33444

http\_request\_duration\_seconds\_bucket{le="0.2"} 100392

http\_request\_duration\_seconds\_bucket{le="0.5"} 129389

http\_request\_duration\_seconds\_bucket{le="1"} 133988

http\_request\_duration\_seconds\_bucket{le="+Inf"} 144320

http\_request\_duration\_seconds\_sum 53423

http\_request\_duration\_seconds\_count 144320

# Finally a summary, which has a complex representation, too:

# HELP rpc\_duration\_seconds A summary of the RPC duration in seconds.

# TYPE rpc\_duration\_seconds summary

rpc\_duration\_seconds{quantile="0.01"} 3102

rpc\_duration\_seconds{quantile="0.05"} 3272

rpc\_duration\_seconds{quantile="0.5"} 4773

rpc\_duration\_seconds{quantile="0.9"} 9001

rpc\_duration\_seconds{quantile="0.99"} 76656 rpc\_duration\_seconds\_sum 1.7560473e+07 rpc\_duration\_seconds\_count 2693

Example:

The following is an example of a Prometheus C++ client:

#include <chrono> #include <map> #include <memory> #include <string> #include <thread>

#include <prometheus/exposer.h> #include <prometheus/registry.h> int main(int argc, char\*\* argv) {

using namespace prometheus;

// create an http server running on port 8080 Exposer exposer{"127.0.0.1:8080"};

// create a metrics registry with component=main labels applied to

// all its metrics

auto registry = std::make\_shared<Registry>();

// add a new counter family to the registry (families combine values

// with the same name, but distinct label dimensions) auto& counter\_family = BuildCounter()

.Name("time\_running\_seconds\_total")

.Help("How many seconds is this server running?")

.Labels({{"label", "value"}})

.Register(\*registry);

// add a counter to the metric family

auto& second\_counter = counter\_family.Add({{"another\_label", "value"},

{"yet\_another\_label", "value"}});

// ask the exposer to scrape the registry on incoming scrapes exposer.RegisterCollectable(registry);

for (;;) {

std::this\_thread::sleep\_for(std::chrono::seconds(1));

// increment the counter by one (second) second\_counter.Increment();

}

return 0;

}

## Clock-Aligned Metrics

Applications that are natively instrumented with Prometheus metrics expose the metrics using an HTTP endpoint, usually /metrics. Applications use the Prometheus annotations in the charts for auto-discovery of endpoints.

Application-exposed metrics normally do not have any timestamp. Some cases, such as 3GPP Specifications, requires metrics to be calculated for predefined clock-aligned time intervals. In such cases, metrics must be exposed to Prometheus with a clock-aligned timestamp.

**Note:** If application exposed metrics have a timestamp, the same timestamp is used.

To use the Clock-Aligned Metrics feature, xgvela.com/mmaas annotations must be available, and the scrape key must be set to true. Refer to [Packaging Model for MMaaS](#_bookmark138) for details.

CIM scrapes and stores clock-aligned metrics from the application based on the measurementInterval, path, and port. MMaaS or Prometheus discovers the target based on the pod labels xgvela.com/

mmaas\_clock\_aligned ="true" and "xgvelaId: <xgvelaid>", portname "metrics", and scrapes the path /metrics/aligned.

Refer to [Metrics with Prometheus Operator](#_bookmark153) for information about clock-aligned metrics with Prometheus at Operator at PaaS,

**Note:** Applications must bind to all interfaces on the port exposing the metrics.

## Metrics with Prometheus Operator

The pod label xgvela.com/mmaas\_clock\_aligned is mandatory for both clock-aligned metrics and non clock-aligned metrics. For non clock-aligned metrics, values must be set to false and for clock-aligned metrics the values must be set to true.

**Note:** Default Prometheus annotations are not required. XGVela CNF Template version 2.8 or higher is required.

Container Port Configuration

As mentioned in [Enabling Scraping of Metrics](#_bookmark139), all the ports to be scraped must be specified as the container port in the charts. With Prometheus Operator, it is mandatory to specify the port name also.

Sample - Part of Charts Deployment Spec

ports:

- name: metrics containerPort: 8095 protocol: TCP

Specifying Non Default Container Port Name

The default container port name is *metrics* . If any application needs to use a different port name, then it must be specified in *values.yaml* as follows:

non\_clock\_aligned\_port\_names: "<App port Name1> [<App port Name2>,<App port Name3>…."

clock\_aligned\_port\_names: "<App port Name1> ,<App port Name2>,<App port Name3>…."

**Note:** If multiple ports are exposing metrics in a single container, you must configure a unique container port name. The default port name is metrics . Other port names can be metrics1 , metricscu , and so on. Define the port name as mentioned in the following example.

Sample - Part of Charts Deployment Spec

ports:

* containerPort: 8095 protocol: TCP

name: "metrics"

* containerPort: 9000 protocol: TCP

name: "metricscu"

For CIM Side Car

ports:

- containerPort: 6060 protocol: TCP

name: "metrics"

## MMaaS Deployment Models

MMaaS supports the following deployment models:

* MMaaS with NF Scrape
* MMaaS with Prometheus Operator

The following are details about each deployment model

* **MMaaS with NF Scrape** The MMaaS scrape scope for this deployment model is NFs deployed at the XGVela layer. XGVela is deployed using Helm commands. Prometheus is deployed at the XGVela layer. Prometheus is also deployed at the PaaS layer.The MMaaS Controller is deployed for configuration management at MMaaS. Measurements, faults, threshold crossing alerts, and notifications are available in this deployment model. Grafana is not configured in this deployment model, and reports are not available at Grafana.
* **MMaaS with Prometheus Operator** In this deployment model, MMaaS not deployed at the XGVela layer. XGVela is deployed using Helm commands. Prometheus Operator is deployed at the CaaS or PaaS layer in a separate namepsace. The MMaaS Controller is deployed for configuration management at MMaaS. Measurements, faults, threshold crossing alerts, and notifications are available in this

deployment model. Grafana is not configured in this deployment model, and reports are not available at Grafana.

**Note:** Alert manager of Prometheus Operator, need to have FMaaS service of XGVela as a receiver.

**Note:** In case of Prometheus Operator, the group\_by must be added in the values.yaml file for Prometheus operator under the alertmanager section. Sample:

alertmanager: config:

global:

receivers:

- name: web-hook webhook\_configs:

- url: [http://fault-service.<XGVela](http://fault-service/) namespace>.svc.cluster.lo- cal:12090/v1/alerts/

route:

group\_by: ['alertname'] group\_interval: 15s group\_wait: 10s receiver: web-hook repeat\_interval: 1h

PaaS Configuration Requirement

XGVela FMaaS must be added as a receiver in Alert Manager. The following is a sample configuration for adding receivers:

receivers:

* name: web-hook webhook\_configs:
* url: [http://fault-service.<XGVela](http://fault-service/) Name Space>.cluster.local:12090/v1/alerts/

[Table 29: MMaaS Deployment Models](#_bookmark155) provides a comparison of the different deployment models:

Table 29: MMaaS Deployment Models

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MMaaS with NF Scrape** | **MMaaS with Prometheus Operator** |  |
| **MMaaS Enabled** | Enabled | Enabled |
| **MMaaS Scrape Type** | NF | N/A |
| **XGVela Deployment Using** | Helm | Helm |
| **MMaaS Controller** | Yes | Yes |
| **Prometheus Type** | XGVela/ Prometheus | PaaS/ Prometheus Operator |

[Table 30: Number of Running Pods for MMaaS Deployment Models](#_bookmark156) describes the number of pods running for each MMaaS deployment model:

Table 30: Number of Running Pods for MMaaS Deployment Models

|  |  |  |  |
| --- | --- | --- | --- |
| **XGVela Pod Name** | **Num-** | **Num-** |  |
|  | **ber of** | **ber of** |
|  | **Running** | **Running** |
|  | **Pods for** | **Pods for** |
|  | **MMaaS** | **MMaaS** |
|  | **with NF** | **with** |
|  | **Scrape** | **Prometheus** |
|  |  | **Operator** |
| mmaas-controller | 1 | 1 |
| xgvela-prometheus-alertmanager | 1 | N/A |
| xgvela-prometheus-server | 1 | N/A |

## Prerequisites for Helm Charts

This section provides the prerequisites for Helm charts. Refer to the following sections:

* [Non Clock Aligned Metrics with XGVela Prometheus](#_bookmark158)
* [Clock Aligned Metrics with XGVela Prometheus](#_bookmark160)
* [Prometheus Operator and Non Clock Aligned Metrics](#_bookmark162)
* [Prometheus Operator and Clock Aligned Metrics](#_bookmark164)

### Non Clock Aligned Metrics with XGVela Prometheus

[Table 31: Prerequisites for Helm Charts for Non Clock Aligned Metrics with XGVela Prometheus](#_bookmark159) describes the prerequisites for Helm charts for the following XGVela deployment models:

* MMaaS with NF Scrape

Table 31: Prerequisites for Helm Charts for Non Clock Aligned Metrics with XGVela Prometheus

|  |  |
| --- | --- |
| **Annotation** | **Value** |
| **Pod Labels** | |
| xgvela.com/mmaas\_clock\_aligned: | "false" [optional] |
| **Pod Annotations** | |

|  |  |
| --- | --- |
| **Annotation** | **Value** |
| prometheus.io/port: | <as per app>  **Note:** When metrics are exposed by multiple containers or by multiple ports, this annotation is not required. Also, when using CIM as a side car, this annotation is not required. |
| prometheus.io/path: | Standard practice is to use "/metrics". If not defined, Prometheus defaults to "/metrics" |
| xgvela.com/mmaas: | Not Required |
| **ContainerPort** | |
| **Deployment->ports** | |
| Containerport | To be defined for all ports exposing metrics. For CIM side car, the port is 6060. |
| ContainerPort name: | Optional. Suggested way is to use name "metrics" for port exposing metrics for Prometheus. |

### Clock Aligned Metrics with XGVela Prometheus

[Table 32: Prerequisites for Helm Charts for Clock Aligned Metrics with XGVela Prometheus](#_bookmark161) describes the prerequisites for Helm charts for the following XGVela deployment models:

* MMaaS with NF Scrape

Table 32: Prerequisites for Helm Charts for Clock Aligned Metrics with XGVela Prometheus

|  |  |
| --- | --- |
| **Annotation** | **Value** |
| **Pod Labels** | |
| xgvela.com/mmaas\_clock\_aligned: | "true" |
| **Pod Annotations** | |
| prometheus.io/port: | "6060" |
| prometheus.io/path: | "/metrics/aligned" |

|  |  |
| --- | --- |
| **Annotation** | **Value** |
| xgvela.com/mmaas: | Required Sample:  xgvela.com/mmaas: '{ "scrape":true", "path": "/metrics", "ports": ["9999","8081"], "scheme": "http",  "measurementInterval": "5s", "maxRetention": "10" }' |
| **ContainerPort** | |
| **Deployment->ports** | |
| Containerport | To be defined for all ports exposing metrics. For CIM side car, the port is 6060. |
| ContainerPort name: | Optional. Suggested way is to use name "metrics" for port exposing metrics for Prometheus. |

### Prometheus Operator and Non Clock Aligned Metrics

[Table 33: Prerequisites for Helm Charts for Prometheus Operator and Non Clock Aligned Metrics](#_bookmark163) describes the prerequisites for Helm charts for the MMaaS with Prometheus Operator XGVela deployment models

Table 33: Prerequisites for Helm Charts for Prometheus Operator and Non Clock Aligned Metrics

|  |  |
| --- | --- |
| **Annotation** | **Value** |
| **Pod Labels** | |
| xgvela.com/mmaas\_clock\_aligned: | "false" |
| **Pod Annotations** | |
| prometheus.io/port: | <as per app>  **Note:** This annotation is not required when metrics are exposed by multiple containers or by multiple ports. Also, When using CIM as a side car, this annotation is not required. |
| prometheus.io/path: | Standard practice is to use "/metrics". If not defined, Prometheus defaults to "/metrics" |
| xgvela.com/mmaas: | Not Required |
| **ContainerPort** | |
| **Deployment->ports** | |

|  |  |
| --- | --- |
| **Annotation** | **Value** |
| Containerport | To be defined for all ports exposing metrics. For CIM side car, the port is 6060. |
| ContainerPort name: | Mandatory as "metrics" for all ports exposing metrics. |

### Prometheus Operator and Clock Aligned Metrics

[Table 34: Prerequisites for Helm Charts for Prometheus Operator and Clock Aligned Metrics](#_bookmark165) describes the prerequisites for Helm charts for Prometheus Operator with clock aligned metrics

Table 34: Prerequisites for Helm Charts for Prometheus Operator and Clock Aligned Metrics

|  |  |
| --- | --- |
| **Annotation** | **Value** |
| **Pod Labels** | |
| xgvela.com/mmaas\_clock\_aligned: | "true" |
| **Pod Annotations** | |
| prometheus.io/port: | "6060" |
| prometheus.io/path: | "/metrics/aligned" |
| xgvela.com/mmaas: | Required Sample:  xgvela.com/mmaas: '{ "scrape":true", "path": "/metrics", "ports": ["9999","8081"], "scheme": "http", "measurementInterval": "5s", "maxRetention":  "10" }' |
| **ContainerPort** | |
| **Deployment->ports** | |
| Containerport | To be defined for all ports exposing metrics. For CIM side car, the port is 6060. |
| ContainerPort name: | Mandatory as "metrics" for all ports exposing metrics.  Refer to [Metrics with Prometheus Operator](#_bookmark153) for details. |

## Multiple Metrics Remote Storage

This feature enables support for storage of multiple metrics at XGVela. For example, this feature can be used to send metrics data to XGVela metric storage and to another metrics storage.

**Note:** This feature is disabled by default.

To enable this feature, include the following highlighted section in the XGVela *values.yaml* file:

xgvela:

xgvela-mgmt:

mmaas-controller: mutiple\_remote\_storage: enable

xgvela\_metrics\_remote\_write\_endpoints:

- http://10.69.21.71:30888/api/prom/remote\_write

- http://10.69.16.71:31764/api/v1/prom/remote/write

# VES Gateway

The VES Gateway service act as a bridge between MMaaS and FMaaS of XGVela, and the Open Net- working Automation Platform (ONAP) VES-Collector.

The VES Gateway is compliant to VES Event Listener Service Specification version 7.1, and supports the following VES event domains:

* Heartbeat
* Fault
* Measurement
* Threshold Crossing Alerts
* Notifications

[Figure 20: VES Gateway Architecture](#_bookmark168) describes the VES Gateway architecture:



Figure 20: VES Gateway Architecture

Refer to the following sections:

###### [Packaging Model for VES Gateway](#_bookmark169)

###### [Northbound Interface Configuration](#_bookmark170)

###### [Discovery of XGVela Management Connection Details - Management IP and Ports](#_bookmark172)

###### [VES Gateway Events](#_bookmark173)

###### [VES - General Introduction and Events](#_bookmark175)

###### [VES Heartbeat Domain](#_bookmark179)

###### [VES Measurement Domain](#_bookmark183)

###### [VES Fault Domain](#_bookmark187)

###### [VES TCA Domain](#_bookmark191)

###### [VES Notifications Domain](#_bookmark195)

## Packaging Model for VES Gateway

To use the VES Gateway feature, the NF chart must contain the following:

* values.yaml: Used to provide the connectivity information for VES Collectors.

## Northbound Interface Configuration

The VES Gateway Northbound Interface can be configured using the XGVela values.yaml file during deployment or with dynamic configuration support using the CMaaS ConfD prompt.

The VES Gateway Northbound Interface supports REST/KAFKA based delivery of events. HTTP and HTTPS are supported as part of REST, where HTTPS limits to basic authentication.

The VES Gateway supports Northbound Interface configuration for multiple collectors. It also allows you to send an event to multiple collectors.

**Note:** KAFKA supports events of all domains.

**Note:** Heartbeat is configurable per collector using the boolean flag heartbeat.

The VES Gateway Northbound Interface can be configured in the values.yaml file as follows: xgvela-mgmt:

ves-gateway: primaryCollector:

fqdn: FQDN for connecting to NBI

Example: "ves\_collector.mvnr-xgvela1-mgmtxgvela-xgvela1.svc.cluster.local"

port: Port for connection.

Example: 8443

secure: Indicates whether the connection is a secure connection:

* Set to true for HTTPS
* Set to false for HTTP

user: Indicates the user name.

**Note:** Applicable only for secure connections.

password: Indicates the password.

**Note:** Applicable only for secure connections.

passphrase: Indicates the passphrase.

**Note:** Applicable only for secure connections.

heartbeat: Indicates whether to send the heartbeats to the CMS or Kafka endpoint. If the value is “true”, heartbeats are sent to the collector, and if the value

is “false”, heartbeats are not sent to the collector. This flag is available in dynamic config update while adding collector details.

nbiFormat: Enables the Rakuten-specific proprietary format conversion. If nbiFormat is set to VES, VES Gateway continues with the normal VES format.

The following are the attribute values:

* VES

nbiType: Enables events to be sent over KAFKA if the nbiType value is set to KAFKA. Setting the nbiType value to REST enables events to be sent with VES Collectors

kafkaBrokers: Indicates the broker list for publishing events in the following format:

[KAFKA ENDPOINT URL:PORT]

kafkaTopic: Indicates the topic name for publishing events.

**Note:** VES Gateway defaults to VES over REST if the nbiFormat, nbiType, kafkaBrokers, and kafkaTopic attributes are not available.

The following is a sample configuration:

xgvela-mgmt: ves-gateway:

primaryCollector:

fqdn: "xgvela-ves-simu.<collector namespace>.svc.cluster.local" port: 8443

secure: false user: "user" password: "pass"

passphrase: "mypassphrase" nbiFormat: "VES"

nbiType: "REST"

kafkaBrokers: "kafka-svc.<XGVela namespace>.svc.cluster.local:9092" kafkaTopic: "<Kafka topic>"

This helps in pushing of events to separate Kafka topics and producing messages to specific partition based on keys. The nfIDs are attached as a key with each event message produced from the VES Gateway. For message keys, if nfID is not available, meID is used as the nfID by default. For heartbeats, the meID is attached as the message key. Refer to [Annexure F](#_bookmark245) for a sample VES Gateway configuration for collector details and domains.

You can configure the prometheus\_address from the values.yaml as follows. The prometheus\_addressfield indicates the Prometheus address to configure for pulling metrics to generate measurements. :

xgvela-mgmt:

ves-gateway:

prometheus\_address: ["http://xgvela-prometheus-server.<XGVela](http://xgvela-prometheus-server/) name- space>.svc.cluster.local:9090"

Note:

The Prometheus address can be configured in VES Gateway using values.yaml at the time of deployment. The address can be either XGVela Prometheus or PAAS Prometheus depending on the deployment needs. The value defaults to XGVela Prometheus in *values.yaml* as mentioned above.

If any invalid Prometheus end point is configured, VES Gateway generates a PrometheusDown alarm. If the Prometheus address is not configured, measurements are not processed by VES Gateway.

### Dynamic Configuration Support for VES Gateway

VES Gateway provides dynamic configuration support through CMaaS. This feature helps to remove the dependency on the ves-gateway config map (for dynamic values) and restart of the VES pod after an update happens.

## Discovery of XGVela Management Connection Details Management IP and Ports

A VES heartbeat event contains XGVela connection details as part of the additionalFields of the event. The following fields are included:

* **ME\_ID** Indicates the UUID of the managed element
* **MGMT\_IP** Indicates the IP address.

The MGMT\_IP can be configured as follows:

xgvela-mgmt: ves-gateway:

data:

XGVelaInfo:

MGMT\_IP: "0.0.0.0" # Management IP of XGVela to be used by CMS.

* **CMAAS\_NETCONF\_PORT** Indicates the Netconf port for configuration management.
* **FMAAS\_HTTP\_PORT** Indicates the HTTP port for alarm management.

**Note:** Currently, you cannot set ME\_ID, CMAAS\_NETCONF\_PORT, and FMAAS\_HTTP\_PORT.

## VES Gateway Events

[Table 35: VES Gateway Events](#_bookmark174) describes the VES Gateway events:

Table 35: VES Gateway Events

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Local Event Name** | **VES Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| **Prometheus Events** | | | | | | |
| PrometheusConnectivityDown | Fault\_XGVelaMvnr\_PrometheusConnectivityDown | WARNING | COMMUNICATION | PrometheusConnectivityUp | This event is generated  when the connectivity from VES Gateway to Prometheus is down. | Check Prometheus pod or connectivity status. |
| PrometheusConnectivityUp | Fault\_XGVelaMvnr\_PrometheusConnectivityUp | CLEAR | COMMUNICATION | N/A | This event is generated  when the connectivity from VES Gateway to Prometheus is up. | N/A |
| **Collector Events** | | | | | |  |
| CollectorDown | Fault\_XGVelaMvnr\_CollectorDown | WARNING | COMMUNICATION | CollectorUp | This event is generated  when the connectivity from VES Gateway to collector is down. | Check collector status. |
| CollectorUp | Fault\_XGVela\_CollectorUp | CLEAR | COMMUNICATION | N/A | This event is generated  when the connectivity from VES Gateway to collector is up. | N/A |

## VES General Introduction and Events

VES (Virtual-function Event Streaming), supports network-function event streaming.The VES Event Listener, which is referred to as VES collector, is capable of receiving any event sent in the VES Common Event Format. The Common Event Format is expressed in the JSON schema.

In the Common Event Format, an event consists of a required Common Event Header block (object) accompanied by zero or more event domain blocks.

Events are well structured packages of information, identified by an eventName. Events can convey measurements, faults, threshold crossing alerts, and other types of information. Events are a way of communicating well-structured packages of information to one or more instances of an Event Listener service.

Event Registration

All events must be compliant with the common event format.

Event Name Naming Standards

Event names sent as part of the commonEventHeader. To ensure the uniqueness of the event name, should conform to the following naming convention:

*{DomainAbbreviation}\_{PublisherName}\_{Description}*

The DomainAbbreviation subfield is derived from the ‘domain’ field in the commonEventHeader, as follows:

* ‘Fault’ for the fault domain
* ‘Heartbeat’ for the heartbeat domain
* ‘Measurement’ for the measurement domain
* ‘Notification’ for the notification domain
* ‘Other’ for the other domain
* ‘Tca’ for the thresholdCrossingAlert domain

The PublisherName subfield describes the vendor product or application publishing the event.*{productName}-{vendorName}*The productName subfield may describe a NF or a NFC. Where NFC names may be reused across different NF’s, they should be specified as: *{NfName}:{NfcName}*

Description - Describes in a compact camel case forma,t the specific information being conveyed by the event.

eventId - A unique key for each domain consisting of domain followed by an integer domain Example: fault000001, heartbeat000001, mfvs000005.

For all domains except Fault, each time a subsequent event is sent, the integer part of eventId is incremented by 1. If eventId is repeated, it assumes a duplicate event. The sequence number is set to 0 for all domains except fault.The eventId on Fault events is the same every time a given fault is raised (onset), raised again at fixed time interval, until it is cleared. Once the fault is cleared, a new eventId is used.

Field Block Versions - A summary of the latest field block version enums as of this version of the API spec is as follows:

commonEventHeader version 4.1

commonEventHeader vesEventListenerVersion enum: 7.1 faultFieldsVersion:4.0

heartbeatFieldsVersion: 3.0

measurementFieldsVersion: 4.0

notificationFieldsVersion: 2.0

thresholdCrossingFieldsVersion: 4.0

The event datatype consists of the following fields which constitute the ‘root level’ of the common event format:

Table 36: Event Datatype Field Descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| commonEventHeader | commonEventHeader | Yes | Fields common to all events |
| faultFields | faultFields | No | Fields specific to fault events |
| heartbeatFields | heartbeatFields | No | Fields specific to heartbeat events |
| measurementFields | measurementFields | No | Fields specific to measurement events |
| notificationFields | notificationFields | No | Fields specific to notification events |
| thresholdCrossingAlertFields | thresholdCrossingAlertFields | No | Fields specific to threshold crossing alert events |

### VES - Common Event Header

The commonEventHeader consists of the following fields common to all events:

Table 37: Event Datatype Field Descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| domain | string | Yes | Fields common to all events  Event domain enumeration: ‘fault’, ‘heartbeat’, ‘measurement’, ‘mobileFlow’ , ‘notification’, ‘other’, ‘perf3gpp’, ‘pnfRegistration’, ‘sipSignaling’, ‘stateChange’, ‘syslog’, ‘thresholdCrossing Alert’,  ‘voiceQuality’  **Note:** VES Gateway populates based on domain. Domain schema is maintained. |
| eventId | string | Yes | Event key that is unique to the event source. The key must be unique within notification life cycle similar to EventID from 3GPP. It could be a sequential number, or a composite key formed from the event fields, such as domain\_sequence. The eventIdshould not include whitespaces. For fault events, eventId is the eventId of the initial alarm; if the same alarm is raised again for changed, acknowledged or cleared cases, eventId must be the same as the initial alarm (along with the same startEpochMicrosec but with a different sequence number).  **Note:** Generated dynamicaly by VES Gateway. fault0000000001 or Measurements00000000 02 |
| eventName | string | Yes | Domain Specific. Check details in Domain specific section. |
| eventType | string | No | Domain Specific. Check details in Domain specific section. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| lastEpochMi crosec | Number | Yes | The latest UNIX time or epoch time associated with the event from any component–as microseconds elapsed since 1 Jan 1970 not including leap seconds.  **Note:** This is calculated and populated by VES Gateway. |
| nfcNamingC ode | string | No | Network function component type: 3 characters (aligned with vfc naming standards) |
| nfNamingCode | string | No | Network function type: 4 characters (aligned with vnf and pnf naming standards) |
| nfVendorName | string | No |  |
| priority | string | Yes | Normal For Heartbeat and measurement. For Other domain, received from FMaaS. |
| reportingEntityId | string | No | UUID identifying the entity reporting the event or detecting a problem in another vnf/vm or pnf which is experiencing the problem. (Note: the AT&T internal enrichment process shall ensure that this field is populated). The reportingEntityId is an id for the reportingEntityNa me. See ‘reportingEntityNa me’ for more information.  Domain Specific. Check details in Domain specfic section |
| reportingEntityName | string | Yes | Name of the entity reporting the event or detecting a problem in another vnf/vm or pnf which is experiencing the problem. May be the same as the sourceName. For synthetic events generated by DCAE, it is the name of the app generating the event.  Domain Specific. Check details in Domain specfic section. |
| sequence | string | Yes | Ordering of events communicated by an event source instance (or 0 if not needed)  Generated by VES Gateway |
| sourceId | string | No | UUID identifying the entity experiencing the event issue, which may be detected and reported by a separate reporting entity.  **Note:** The AT&T internal enrichment process ensures that this field is populated.  The sourceId is an id for the sourceName. See ‘sourceName’ for more information. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| sourceName | string | Yes | Name of the entity experiencing the event issue, which may be detected and reported by a separate reporting entity. The sourceName identifies the device for which data is collected. A valid sourceName must be inventoried in A&AI. If sourceName is a xNF (vnf or pnf), xNFC or VM, then the event must be reporting data for that particular xNF, xNFC or VM. If the sourceName is a xNF, comprised of multiple xNFCs, the data must be reported/aggregated at the xNF level. Data for individual xNFC must not be included in the xNF sourceName event. |
| startEpoch Microsec | number | Yes | The earliest UNIX time or epoch time associated with the event from any component–as microseconds elapsed since 1Jan 1970 not including leap seconds. For measurements and heartbeats, where events are collected over predefined intervals, startEpochMicros ec shall  be rounded to the nearest interval boundary (Example: the epoch equivalent of 3:00 PM, 3:10PM, 3:20PM, and so on). For fault events, startEpochMicrosec is the timestamp of the initial alarm; if the same alarm is raised again for changed, acknowledged or cleared cases, startEpochMicrosec must be the same as the initial alarm (along with the same eventId and an incremental sequence number). For devices with no timing source (clock), the default value will be 0 and the VES collector will replace it with Collector timestamp (when the event is received) |
| version | string | Yes | Version of the event header as “#.#” where # is a digit; see section 1 for the correct digits to use.  **Note:**  commonEventHeader version. VES Specifications 7.1 specific version is "4.1" |
| vesEventList enerVersion | string | Yes | Version of the VES Event Listener API spec that this event is compliant with (as “#” or “#.#” or “#.#.#” where # is a digit. |

## VES Heartbeat Domain

Heartbeats are sent between VES Gateway and VES collector. The heartbeat event consists of Common Event header and Hearbeat specific fields. The heartbeatFields with additional fields are as follows:

Table 38: Heartbeat Domain Fields Descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** | **Implementation** |
| heartbeatFieldsVersion | string | Yes | Version of the heartbeatFields block as “#.#” where # is a digit; see section 1 for the correct digits to use.  Version of the heartbeatFields. VES 7.1 Specficversion is “3.0” | Version of the heartbeatFields. VES 7.1 Specific version is “3.0” |
| heartbeatInterval | Integer | Yes | Current heartbeatInterval in seconds | Current heartbeatInterval in seconds |
| additionalFields | hashMap | No | Additional expansion fields, if needed.  The following are the additional fields   * APIGW\_PASSWORD   + Credential provided in heartbeat additional field * APIGW\_AUTH\_TYPE   + Authentication type mentioned in heartbeatadditional fields * COLLECTOR\_ID CMS endpoint ID to which heartbeat is sent. * APIGW\_PORT API GW port in heartbeat additional fields * MGMT\_IP Management IP mentioned in heartbeat additional fields. * FMAAS\_HTTP\_PORT   + Service port for FMAAS in heartbeat additional fields. * APIGW\_USERNAME   + API GW credential in heartbeat additional fields. * ME\_ID Unique ID generated by VES indicated managed elemen   CMAAS\_NETCONF\_PORT Service port for CMAAS Netconf mentioned in heartbeat additional fields |  |

### Common Event Header Fields for Heartbeat Domain

The following table describes the Common Event Header fields for the Heartbeat domain:

Table 39: Common Event Header Fields for Heartbeat Domain

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Implementation** |
| domain | Yes | Heartbeat  Version of the heartbeatFields. VES 7.1 Specficversion is “3.0” |
| eventId | Yes | Event key that is unique to the event source generated dynamicaly by VES Gateway. Example: heartbeat0000009681 |
| eventName | Yes | This is unique event name generated by VES in the following format: *Heartbeat\_{NfName}-{vendorName}* |
| eventType | Yes | The type of event generated by VES. Example: HEARTBEAT |
| lastEpochMicrosec | Yes | The latest UNIX time or epoch time associated with the event from any component. Calculated and Populated by VES Gateway. Example: 1584100020000428 |
| nfcNamingCode | Yes | This is network function component type, aligned with vfc naming standards. Example: "testapp” |
| nfNamingCode | Yes | This is a network function type, aligned with nf naming standards. Example: "testapp” |
| nfVendorName | Yes | Network function vendor name. Example: “mvnr” |
| priority | Yes | This is processing priority of event Normal -Default For Heartbeat |
| reportingEntityId | Yes | UUID corresponding to reportingEntityName Example: "ca7d67df-  be2b-39cb-917a-8d435d42558f" |

## VES Measurement Domain

The measurement event consists of Common Event header and measurementFields specific fields. The following table describes the VES Measurement domain fields:

Table 40: Measurement Domain Field Descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** | **Implementation** |
| additionalFields | hashMap | No | Additional measurement fields, if needed. |  |
| additionalMeasurements | arrayOfNamedHashMap | No | Array of named hashMap, if needed | Yes |
| additionalObjects | arrayOfJsonObject | No | Array of JSON objects described by name, schema and other metainformation, if needed | Yes |
| codecUsageArray | codecsInUse [] | No | Array of codecs in use |  |
| concurrentSessions | integer | No | Depending on the context over the measurementInterval: peak total number of users, subscribers, devices, adjacencies for the VM, or peak total number of subscribers or devices for the xNF |  |
| cpuUsageArray | cpuUsage [] | No | Usage of an array of CPUs | Yes |
| diskUsageArray | diskUsage [] | No | Usage of an array of disks | Yes |
| featureUsageArray | hashMap | No | The hashMap key should identify the feature, while the value defines the number of times the identified feature was used |  |
| filesystemUsageArray | filesystemUsage [] | No | Filesystem usage of the VM on which the xNFC reporting the event is running | Yes |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** | **Implementation** |
| latencyDistribution | latencyBucketMeasure [ ] | No | Array of integers representing  counts of requests whose latency in milliseconds falls within per-xNFconfigured ranges; where latency is the duration between a service request and its fulfillment. |  |
| meanRequestLatency | number | No | Mean seconds required to respond to each request for the VM on which the xNFC reporting the event is running |  |
| measurementFieldsVersion | string | Yes | Version of themeasurementFields block as “#.#” where # is a digit. | VES 7.1 Specific  version is “4.0” |
| measurementInterval | Number | Yes | Interval over whichmeasurements are being reported in seconds | Yes. Configured per NF. |
| memoryUsageArray | memoryUsage [] | No | Memory usage of an array of VMs | Yes |
| nfcScalingMetric | integer | No | Represents the busyness of the network function from 0 to 100 as reported by the NFC |  |
| nicPerformanceArray | nicPerformance [ ] | No | Performance metrics of an array of network interface cards |  |
| numberOfMediaPortsInUse | integer | No | Number of media ports in use |  |
| requestRate | number | No | Peak rate of service requests per second to the xNF over the measurementInterval |  |

### Common Event Header Fields for the Measurement Domain

The following table describes common event header fields for the measurement domain:

Table 41: Common Event Header Fields for the Measurement Domain

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Description** |
| domain | Yes | The eventing domain associated with the event.  Example: measurement |
| eventId | Yes | The event key that is unique to the event source which is generated dynamically by the VES Gateway.  Example: Measurements0000019361 |
| eventName | Yes | Unique event name for measurement with the following format: Measurement\_ + NfNamingCode  + "\_Measurements" |
| eventType | Yes | Type of event generated by the VES Gateway. Example: MEASUREMENT |
| lastEpochMicrosec | Yes | The latest UNIX time or epoch time associated with the event from any component calculated and populated by the VES Gateway. Example: 1584100020000428 |
| nfcNamingCode | Yes | The network function component type, aligned with vfc naming standards. |
| nfNamingCode | Yes | The network function type, aligned with nf naming standards |
| nfVendorName | Yes | The network function vendor type. |
| priority | Yes | The processing priority for measurements. The default for measurements is Normal. |
| reportingEntityId | Yes | UUID identifying the entity reporting the event corresponding to the reportingEntityName. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Description** |
| reportingEntityName | Yes | The MO name of fault-service in case of Fault, TCA, and Notification and the MO name of VES  Gateway in case of Measurement and HB  dnPrefix,ManagedElement=me-<xgvelaId>,NetworkFunc-  tion=<nfId>,NFService=<nfServic eId>  Example : "mvnr,Manage dElemen-  t=me-xgvela1,NetworkFunc-  tion=xgvela1,NFService=vesgateway” |
| sequence | Yes | The ordering of events communicated by an event source instance or 0 if not needed. |
| sourceId | Yes | The UUID identifying the entity experiencing the event issue corresponding to the SourceName. |
| sourceName | Yes | name of the entity experiencing the event issue.  [namespace]\_<micro service> |
| startEpochMicrosec | Yes | The earliest UNIX time or epoch time associated with the event from any component populated by VES Gateway. Example: 1584100020000428 |
| version | Yes | The commonEventHeader version. The VES Specifications 7.1 specific version is "4.1" |
| vesEventListenerVersion | Yes | The VES Specifications 7.1 specific version is "7.1" |

## VES Fault Domain

The fault event consists of Common Event header and FaultField specific fields. The VES Fault domain fields are as follows:

Table 42: VES Fault Domain Fields Descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** | **Implementation** |
| alarmAdditionalInformation | hashMap | No | Additional alarm information | Received from CIM. |
| alarmCondition | string | Yes | Short name of the alarm condition/problem, such as a trap name. | Received from FMaaS. |
| alarmInterfaceA | string | No | Card, port, channel or interface name of the device generating the alarm. This could reflect the managed object. | Empty by default |
| eventCategory | string | No | Event category. Example: ‘license’, ‘link’, ‘routing’, ‘security’, ‘signaling’ | eventType as per eventDef |
| eventSeverity | string | Yes | Event severity enumeration: ‘CRITICAL’, ‘MAJOR’,‘MINOR’, ‘WARNING’ | perceivedSeverity as per eventDef |
| eventSourceType | string | Yes | Examples: ‘card’, ‘host’, ‘other’, ‘port’, ‘portThreshold ’, ‘router’, ‘slotThreshold’ , ‘switch’, ‘virtualMachine’, ‘virtualNetwor kFunction’. This could be the managed object class. | Received from CIM. |
| faultFieldsVersion | string | Yes | Version of the faultFields block as “#.#” where # is a digit. | VES 7.1  Specfic version is “4.0”  Received from FMaaS. |
| specificProblem | string | Yes | Description of the alarm or problem (Example: ‘eNodeB 155197 in PLMN 310-410 with eNodeB name KYL05197 is lost’). 3GPP probable cause would be included in this field. | specificProblem as per eventDef |
| vfStatus | string | Yes | Virtual function status.  enumeration: ‘Active’, ‘Idle’, ‘Preparing to terminate’, ‘Ready to terminate’, ‘Requesting Termination’ | Active by default |

### Common Event Header Fields for Fault Domain

The following table describes the common event header fields for the Fault domain. The values are received from FMaaS:

Table 43: VES Fault Domain Common Event Header Fields Descriptions

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Implementation** |
| domain | Yes | fault |
| eventId | Yes | MD5 digest of sourceId and eventName and the value is a 32 character hex string |
| eventName | Yes | Fault\_{NfName}-{vendorName}\_<localEventName> |
| eventType | No | eventType as per eventDef |
| lastEpochMicrosec | Yes | Calculated by VES Gateway. |
| nfcNamingCode | Yes | Corresponds to nfServiceType |
| nfNamingCode | Yes | Corresponds to nfType |
| priority | Yes | Normal by default |
| reportingEntityId | No | UUID corresponding to reportingEntityName |
| reportingEntityName | Yes | The MO name of fault-service in case of Fault,TCA, and Notification and MO name of VES Gateway in case of Measurement and Heartbeat.  dnPrefix,ManagedElement=me-<xgvelaId>,NetworkFunction=<nfId>,NFService=<nfServic eId>  Example :  "mvnr,Manage dElement=me-xgvela1,NetworkFunc-  tion=xgvela1,NFService=vesgateway" |
| sequence | Yes | Incremented per event per collector. |
| sourceId | No | UUID corresponding to SourceName |
| sourceName | Yes | The MO name of the source in general, with the following conditions,   1. If the application includes managed\_objects, then sourceName is the MO name of the source suffixed with the managed\_objects. 2. If the application sends the sourceName as part of the event to FMaaS, the same value is used as is. |
| startEpochMicrosec | Yes | Calculated by VESGW. |
| version | Yes | commonEventHeader version. VES Specifications 7.1 specific version is "4.1" |
| vesEventListenerVersi on | Yes | VES Specifications 7.1 specific version is "7.1" |

## VES TCA Domain

The thresholdCrossingAlert event consists of Common Event header and thresholdCrossingAlert specific fields. The TCA domain fields are as follows:

Table 44: TCA Domain Fields Descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** | **Implementation** |
| criticality | string | Yes | Enumeration: ‘CRIT’, ‘MAJ’ | Corresponding to eventSeverity |
| hashMap | hashMap | Yes | Key is the name of the counter and value is the current value of the counter | NA:NA by default |
| threshholdCrossed | string | Yes | Last threshold that was crossed | NA by default |
| additionalFields | hashMap | No | Additional threshold crossing alert fields if needed | Received through annotations |
| additionalParameters | counter [ ] | Yes | Array of performance counters |  |
| alertAction | string | Yes | Enumeration: ‘SET’, ‘CONT’, ‘CLEAR’ | Corresponding to eventSeverity |
| alertDescription | string | Yes | Unique short alert description (e.g., NECPUMEM) | specificProblem as per eventDef |
| alertType | string | Yes | Enumeration: ‘CARDANOMALY’, ‘INTERFACEANOMALY’, ELEMENTANOMALY’, ‘SERVICEANOMALY’ | SERVICE-ANOMALY by  default |
| collectionTimestamp | string | Yes | Time when the Fault service picked up the data | Time when the Fault service picked up the data |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** | **Implementation** |
| eventSeverity | string | Yes | Event severity or priority enumeration: ‘CRITICAL’, ‘MAJOR’,  ‘MINOR’, ‘WARNING’ , ‘NORMAL’ | perceivedSeverity as per eventDef |
| eventStartTimestamp | string | Yes | Time closest to when the measurement was made. | As received from Prometheus |
| thresholdCrossing FieldsVersion | string | Yes | Version of the thresholdCrossingAlertFiel ds block as “#.#” where # is a digit. | VES 7.1 Specific version is “4.0” |

### Common Event Header Fields for TCA Domain

The following table describes the common event header fields for TCA domain:

Table 45: TCA Domain Fields Descriptions

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Implementation** |
| domain | Yes | thresholdCrossingAlert |
| eventId | Yes | MD5 digest of sourceId and eventName and the value is a 32 character hex string |
| eventName | Yes | Tca\_{NfName}-{vendorName}\_<localEventName> |
| eventType | Yes | eventType as per eventDef |
| lastEpochMicrosec | Yes | Calculated and Populated by VES Gateway. e.g. 1584100020000428 |
| nfcNamingCode | No | Corresponds to nfServiceType |
| nfNamingCode | No | Corresponds to nfType |
| priority | Yes | Normal by default. |
| reportingEntityId | No | UUID Corresponding to reportingEntityName |

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Implementation** |
| reportingEntityName | Yes | reportingEntityName: MO name of fault-service in case of Fault,TCA and Notification & MO name of vesgw in case of Measurement and HB  dnPrefix,ManagedElement=me-  <xgvelaId>,NetworkFunction=<nfId>,NFService=<nfServic eId>  Example :  "mvnr,Manage dElement=me-  xgvela1,NetworkFunction=xgvela1,NFService=faultservice" |
| sequence | Yes | Incremented per event per collector. |
| sourceId | No | UUID corresponding to SourceName |
| sourceName | Yes | sourceName is the MO name of the source in general, with following other conditions,   1. If application includes managed\_objects, then sourceName is the MO name of the source suffixed with managed\_objects, 2. If application sends sourceName as part of the event to FMaaS, the same value is used as it is. |
| startEpochMicrosec | Yes | Calculated by ves-gateway |
| version | Yes | commonEventHeader version. VES Specifications 7.1 specific version is "4.1" |
| vesEventListenerVersi on | Yes | VES Specifications 7.1 specific version is "7.1" |

## VES Notifications Domain

The thresholdCrossingAlert event consists of Common Event header and thresholdCrossingAlert specific fields.

The following table describes the VES Notifications Domain fields:

Table 46: TCA Domain Fields Descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** | **Implementation** |
| additionalFields | hashMap | No | Additional notification fields if needed | As received from CIM |
| changeIdentifier | string | Yes | System or session identifier associated with the change | As received from CIM |
| changeType | string | Yes | Describes what has changed for the entity, for example: configuration changed, capability added, capability removed… | As received from CIM |
| newState | string | No | New state of the entity, for example: ‘inService’, ‘maintenance’, ‘outOfService’ | As received from CIM |
| notificationFieldsVersion | string | Yes | Version of the notificationFields block as “#.#” where # is a digit; see section 1 for the correct digits to use. | VES 7.1 Specfic version is “2.0" |
| oldState | string | No | Previous state of the entity. Example: ‘inService’, ‘maintenance’, ‘outOfService’ | As received from CIM |

### Common Event Header Fields for Notifications Domain

The following table describes the common event header fields for the Notifications domain :

Table 47: TCA Domain Fields Descriptions

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Implementation** |
| domain | Yes | Notification |
| eventid | Yes | MD5 digest of sourceId and eventName and the value is a 32 character hex string |
| eventName | Yes | Notification \_{NfName}-{vendorName}\_<localEventName> |
| eventType | Yes | eventType as per eventDef |
| lastEpochMicrosec | Yes | Calculated by ves-gateway.  Example: 1584100020000428 |
| nfcNamingCode | Yes | Corresponds to nfServiceType |
| nfNamingCode | Yes | Corresponds to nfType |
| priority | Yes | Normal by default. |
| reportingEntityId | Yes | UUID Corresponding to reportingEntityName |

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Implementation** |
| reportingEntityName | Yes | reportingEntityName: MO name of fault-service in case of Fault,TCA and Notification & MO name of vesgw in case of Measurement and HB  dnPrefix,ManagedElement=me-<xgvelaId>,NetworkFunction=<nfId>,NFService=<nfServic eId>  Example :  "mvnr,Manage dElemen-  t=me-xgvela1,NetworkFunction=xgvela1,NFService=fault-service" |
| sequence | Yes | Incremented per event per collector. |
| sourceId | Yes | UUID corresponding to SourceName |
| sourceName | Yes | sourceName is the MO name of the source in general, with following other conditions,   1. If application includes managed\_objects, then sourceName is the MO name of the source suffixed with managed\_objects, 2. If application sends sourceName as part of the event to FMaaS, the same value is used as is. |
| startEpochMicrosec | Yes | Calculated by ves-gateway.  Example: 1584100020000428 |

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Implementation** |
| version | Yes | commonEventHeader version. VES Specifications 7.1 specific version is "4.1" |
| vesEventListenerVersi on | Yes | VES Specifications 7.1 specifc version is "7.1" |

# FMaaS

XGVela provides Fault Management as a Service (FMaaS) to define events and generate events. FMaaS implements correlation and alarm management functions for microservices. FMaaS implements the following categories of events:

* Threshold Crossing Events
* Application Runtime Events

All events are stored in a highly-available storage.

[Figure 21: FMaaS Design](#_bookmark200) describes the design of FMaaS:



Figure 21: FMaaS Design

FMaaS uses Elasticsearch as the storage backend. FMaaS is tested to work with Elasticsearch versions

6.4.3 and 7.8. By default, it is configured to use Elasticsearch 6.4.3.

**Note:** Zookeeper configuration is required.

The Elasticsearch endpoint details can be changed using the following annotations in the deployment values.yaml of XGVela:

Refer to the following sections for more details:

xgvela-mgmt: fault-service:

es:

host: "elasticsearch.<elastic-search namespace>.svc.cluster.local" #elasticsearch fqdn

port: "9200" scheme: "http"

version: "6.4.3" # elasticsearch version. Tested versions are

6.4.3 and 7.8.

###### [Packaging Model for Fault Management](#_bookmark201)

###### [Defining Events](#_bookmark202)

###### [Defining TCAs](#_bookmark204)

###### [System TCAs](#_bookmark205)

###### [Event Subscriptions](#_bookmark206)

###### [FMaaS Events](#_bookmark211)

###### [FMaaS Metrics](#_bookmark213)

###### [FMaaS Interfaces](#_bookmark215)

## Packaging Model for Fault Management

Use the latest the chart template for building NF charts to support the fault management feature of XGVela. Refer to [CMaaS](#_bookmark114) for more information about charts.

To use the fault management feature, the NF chart must contain the following file:

* **eventdef/** : Folder containing the events.json file.
* **alerts/** : Folder containing the alerts.yaml file.
* **alerts.yaml** : Allows you to define custom alert rules for application threshold crossing alerts (TCAs). You can also create alert groups to add the custom alert rules and monitor TCAs. This file must be present in the alerts folder. Refer to [Defining TCAs](#_bookmark204) for details on defining TCAs.
* **events.json** : Allows you to define events. This file must be present in the eventdef folder. Refer to [Defining Events](#_bookmark202) for details on defining application runtime events.
* **values.yaml** : Set the create flag to true to enable fault management. Ensure that the enable\_event flag is set to true in the CIM configuration.

## Defining Events

You can define the events using the following format:

{

"events" : {

"<event-name>" : {

"eventType": "", "perceivedSeverity": "", "probableCause": "", "specificProblem": "", "rootCauseIndicator": "", "trendIndication": "", "correlatedNotifications": "", "additionalText": "", "proposedRepairAction":"", "domain":"" "serviceTypeAffected": "", "alarmHierarchy": "", "classification": "", "alarmType": ""

},

"<event-name>" : {

"eventType": "",

… "domain":""

}

}

}

**Note:** You can specify one of the following values for domain:

* thresholdCrossingAlert
* notification
* fault

[Table 48: FMAaaS Domains and Severities](#_bookmark203) describes the various domains and the supported severities:

Table 48: FMAaaS Domains and Severities

|  |  |
| --- | --- |
| **Domain** | **Supported Severities** |
| fault | * CRITICAL * MAJOR * MINOR * WARNING * CLEAR |
| thresholdCrossingAlert | * CRITICAL * MAJOR * MINOR * WARNING * CLEAR |
| notiification | * INFO |

Any event with severity INFO will be automatically converted to notification domain if "domain:notification" is not explicitly provided in the event definition.

The CIM sidecar provides the support for integrating application events to the FMaaS system. In order to generate an event, the NF must push the event notification message to CIM over NATS topic EVENT using the flatbuffer schema mentioned in [FMaaS Interfaces](#_bookmark215).

## Defining TCAs

FMaaS allows you to define custom alert rules for application TCAs. You can also create alert groups to add the custom alert rules and monitor TCAs.

Alert rules are used to detect condition based on thresholds.

Once the condition is resolved, Prometheus triggers a resolved notification, and FMaaS generates a CLEAR event. You must define alert rules in the alerts.yaml file.

The following is a sample alert rule for application TCA.

- alert: "CimMessages"

expr: cim\_total\_messages\_counter > 100 for: 1m

labels:

severity: CRITICAL

annotations:

sourceId: {{ `"{{ $labels.instance }}"` }} eventName: "CimTotalMessages"

message: "Cim messages counter has reached its set threshold" namespace: {{ `"{{ $labels.kubernetes\_namespace }}"` }} podId: {{ `"{{ $labels.pod }}"` }}

microservice: {{ `"{{ $labels.microSvcName}}"` }} nfType: {{ $.Values.nf.nfType | quote}}

nfId: {{ $.Values.nf.nfId | quote}}

nfServiceType: {{ `"{{ $labels.microSvcName}}" `}} svcVersion: {{ $.Values.nf.version | quote}} pod\_uuid : {{ `"{{ $labels.pod\_uuid }}" `}}

uhn: {{ `"{{ $labels.uhn }}" `}}

cnfc\_uuid: {{ `"{{ $labels.cnfc\_uuid }}" `}}

The above rules check the condition for one minute before raising an alert.EventName from the annotation section will be used for correlating the event from the event definition. Details from definition will override any details in the alert.

**Note:** Missing values for the annotations such as eventName, namespace, sourceID, podId, and microservice will lead to an eventdefinition look up failure, and the event will not be processed further. A log message stating the reason is added to the FMaaS log.

## System TCAs

System TCAs are events which are triggered when the CPU, memory, or disk usage of a particular node crosses the defined thresholds. XGVela supports the following TCAs at the system level:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Default Value** | **Repair Action** | **Clear Events** | **Description** |
| SystemOutOfDiskSpace | WARNING | HostOS | 70% | Check and allocate sufficient disk. The alarm is auto-cleared when the used disk space goes below the threshold. | SystemOutOfDiskSpace with CLEAR severity | This event is triggered if the disk  space usage is greater than the defined threshold for one minute. |
| SystemOutOfMemory | WARNING | HostOS | 80% | Check and rectify mem- | SystemOutOfMemory | This event is triggered if |
|  |  |  |  | ory overload | with CLEAR | the mem- |
|  |  |  |  | condition. | severity. | ory usage |
|  |  |  |  | The alarm is |  | for a node |
|  |  |  |  | auto-cleared |  | is greater |
|  |  |  |  | when the |  | than the de- |
|  |  |  |  | available |  | fined thresh- |
|  |  |  |  | memory |  | old for one |
|  |  |  |  | goes above |  | minute. |
|  |  |  |  | the thresh- |  |  |
|  |  |  |  | old. |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Default Value** | **Repair Action** | **Clear Events** | **Description** |
| SystemHighCpuLoad | WARNING | HostOS | 80% | Check and rectify the | SystemHighCpuLoad | This event is triggered if |
|  |  |  |  | CPU over- | with CLEAR | the CPU |
|  |  |  |  | load con- | severity | usage for |
|  |  |  |  | dition. The |  | a node is |
|  |  |  |  | alarm is au- |  | greater |
|  |  |  |  | to-cleared |  | than the de- |
|  |  |  |  | when the |  | fined thresh- |
|  |  |  |  | available |  | old for one |
|  |  |  |  | CPU goes |  | minute. |
|  |  |  |  | above the |  |  |
|  |  |  |  | threshold. |  |  |

The System TCA limits for SystemOutOfDiskSpace, SystemHighCpuLoad, and SystemOutOfMemory can be configured during XGVela deployment using the following variables in the deployment values.yaml of XGVela:

xgvela:

xgvela-mgmt:

alerts:

diskspace: 70

cpu: 80

memory: 80

## Event Subscriptions

This section describes the Notification/Event subscription APIs. NFs can subscribe to receive events from XGVela and other NF microservices.

The subscription API is a REST endpoint exposed by CIM (local to the POD). Subscribed events are notified by CIM to the application over NATS and follows the VES format. The NF must register to NATS topic 'EVENT-NOTIFICATION' using the following steps before adding a subscription.

1. Subscribe to NATS topic 'EVENT-NOTIFICATION' to receive subscribed events.
2. Subscribe to CIM over REST to notify events.An event subscription is a filter condition. Multiple subscriptions are supported. Subscription filter can contain one of more of the following fields:
3. localEventName : Name of the event. Example: NetworkFunctionCreated, NFServiceCreated, NetworkFunctionDeleted, NetworkFunctionStateChanged
4. nfNamingCode : NF type. Example: cuup, cucp, du, amf, smf, nssf
5. nfcNamingCode : Microservice type. Example: amfee, amfcomm

**Note:** Each field can have multiple values in the form of array of string.

The filter evaluation rule is as follows:

1. OR across field values
2. AND across fields

### Event Subscription Interfaces

Refer to the following sections:

* [Create Subscription](#_bookmark208)
* [Modify Subscription](#_bookmark209)
* [Delete Subscription](#_bookmark210)
  + - 1. Create Subscription

This interface is used to create a subscription. **URI** /api/v1/\_operations/event/subscriptions **HTTP Method** POST

**HTTP Port** 6060

**Content-Type** application/json **Request Parameters** N/A **Request Body**

{

"localEventName" : [<array of event names>], "nfNamingCode" : [<array of nfNamingCodes>], "nfcNamingCode" : [<array of nfcNamingCodes>], "localNfId" : [<array of localNfIds>]

}

**Note:** localEventName maps to the local event name defined as per eventDef.nfNamingCode maps to nfType.nfcNamingCode maps to nfServiceType. localNfId maps to nfId.

Response

Success Case

200 OK

{

"subscriptionId" : "<value of subscriptionId>"

}

Failure Case

* **400 Bad Request** in case of input error.

Sample Request Body

{

"localEventName": ["NetworkFunctionCreated", "NetworkFunctionDeleted"],

"nfNamingCode": ["amf", "smf"], "nfcNamingCode": ["amfee", "pdusession"]

"NFServiceCreated",

}

* + - 1. Modify Subscription

This interface is used to modify an existing subscription. This interface replaces the subscription details with new content.

**URL** /api/v1/\_operations/event/subscriptions/{subscriptionId}

HTTP Method PUT

**HTTP Port** 6060

**Content-Type** application/json

Request Parameters

"name": " subscriptionId ", "in": "path",

"description": "Identifies the subscription id to be updated", "required": true

Request Body

{

"localEventName" : [<array of event names>], "nfNamingCode" : [<array of nfNamingCodes>], "nfcNamingCode" : [<array of nfcNamingCodes>], "localNfId" : [<array of localNfIds>]

}

Response

200 OK 400 Bad Request in case of input error.

* + - 1. Delete Subscription

This interface is used to delete an existing subscription. **URL** /api/v1/\_operations/event/subscriptions/{subscriptionId} **HTTP Method** DELETE

**HTTP Port** 6060

**Content-Type** application/json

Request Parameters

"name": " subscriptionId ", "in": "path",

"description": "Identifies the subscription id to be unsubscribed", "required": true

Request Body

N/A

Response

*Success Case*

204 No content

*Failure Case*

400 Bad Request in case of input error.

## FMaaS Events

[Table 49: FMaaS Events](#_bookmark212) describes the FMaaS events:

Table 49: FMaaS Events

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| FmaasCon- | INFO | CONFIG\_UP- | N/A | This event in- | N/A |
| figUpdateAt- |  | DATE |  | dicates that |  |
| tempted |  |  |  | FMaaS config- |  |
|  |  |  |  | uration update |  |
|  |  |  |  | was attempted. |  |
| FmaasCon- | INFO | CONFIG\_UP- | N/A | This event in- | N/A |
| figUpdateSuc- |  | DATE |  | dicates that |  |
| cess |  |  |  | FMaaS configu- |  |
|  |  |  |  | ration update is |  |
|  |  |  |  | successful. |  |

## FMaaS Metrics

[Table 50: FMaaS Metrics](#_bookmark214) describes the FMaaS metrics

Table 50: FMaaS Metrics

|  |  |
| --- | --- |
| **Metric Name** | **Description** |
| fmaas\_config\_update\_attempt\_total | Indicates the number of times that FMaaS configuration update was attempted. |
| fmaas\_config\_update\_failure\_total | Indicates the number of times that FMaaS configuration update is successful |
| fmaas\_config\_update\_success\_total | Indicates the number of times that FMaaS configuration update has failed. |

## FMaaS Interfaces

Refer to the following sections:

* [Application Interface](#_bookmark216)
* [Management Interfaces](#_bookmark218)

### Application Interface

To integrate with the FMaaS feature, and to create an application runtime event, applications must send the event to CIM over NATS using the EVENT topic. The following is a request format:

Type: Flatbuffer Schema:

namespace EventInterface; table KeyValue {

key:string(key);

value:string;

}

table Event {

event\_name:string; event\_time:long; container\_id:string; managed\_object:[KeyValue]; additional\_info:[KeyValue]; threshold\_info:[KeyValue];

state\_change\_definition:[KeyValue]; monitored\_attributes:[KeyValue]; source\_id:string; source\_name:string; event\_source\_type:string;

}

root\_type Event;

[Table 51: Event Fields](#_bookmark217) describes the event fields::

Table 51: Event Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Qualifier** | **Description** |
| event\_name | String | Mandatory | Indicates the event name as per the event definition. |
| event\_time | Long | Mandatory | Indicates the epoch time in milliseconds. |
| container\_id | String | Mandatory | Indicates the container ID of the originating event. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Qualifier** | **Description** |
| managed\_object | KV Array | Optional | Indicates an ordered list of name-value pairs identifying the managed object instance. For example, if an application wants to raise an event particular to a link which it manages, it can mention the link ID in the managed object.  The managed object appended to the Source Name of the event. By default, Source Name is the instance generating the event. The Source ID is a UUID derived from the Source Name. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Qualifier** | **Description** |
| additional\_info | KV Array | Optional | Indicates additional information about the event.  If the application wants to pass additional information using the event to NBI/subscriber, the necessary fields can be provided as additional information.  The additional\_info attribute is available in the alarmAddtionalInformation section of the fault domain event and the additionalFields section of the notification and thresholdCrossingAlert domain events. XGVela includes the following additional information by default wherever applicable:   * meId * meLabel * nfId * nfLabel * nfServiceId * nfServiceLabel * nfServiceInstanceId * nfServiceInstanceLabel   In addition to these fields, XGVela Topology notifications include the following additional optional fields as applicable:   * nfServiceName * nfServiceSwVersion * nfType * nfSwVersion * nfName * probableCause * haRole * nfServiceType * nfServiceInstanceName   All these keywords are reserved for XGVela. If the application provides the same key, the value set by the application gets precedence. |
| threshold\_info | KV Array | Optional | Indicates threshold information. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Qualifier** | **Description** |
| state\_change\_definition | KV Array | Optional | Indicates the state change definition.  To generate a notification event with state change informa-  tion, the application must fill state\_change\_definition with following set of keys and corresponding values:   * "changeIdentifier" System or session identifier associated with the change * "changeType" Identifies the type of change. This is an optional field which defaults to the event name defined as per the definition. * "oldState" Previous state of the entity * "newState" New state of the entity   **Note:** The listed parameters, if available, will be mapped to the fields with corresponding names in the VES notification fields. |
| monitored\_attributes | KV Array | Optional | Indicates monitored attributes. |
| source\_id | String | Optional | Indicates an identifier for the entity experiencing the event issue, which may be detected and reported by a separate reporting entity. This can be used to override the sourceId derivation in FMaaS and if provided, XGVela uses this value to fill the VES sourceId parameter. |
| source\_name | String | Optional | Indicates the name of the entity experiencing the event issue, which may be detected and reported by a separate reporting entity. This field is mandatory  if value for source\_id is provided. XGVela uses this value to fill the VES sourceName parameter. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Qualifier** | **Description** |
| event\_source\_type | String | Optional | Indicates the source type for the event. This field is optional, and is applicable only in cases where the event is generated by an XGVela managed element for an externally managed element, such as an SDaaS generating an event for RU. The field maps to eventSourceType in VES and FMaaS fills the value of nfType in this field by default. If the value for the event\_source\_type default. If the value for the event\_source\_type field is provided by the application, the default behavior of FMaaS is overridden, and it will use the value provided by the application. |

### Management Interfaces

Refer to the following sections for information about management interfaces:

* [Get Alarms](#_bookmark219)
* [Comment Multiple Alarms](#_bookmark220)
* [Comment a Single Alarm](#_bookmark221)
* [Clearing Acknowledging or Unacknowledging Multiple Alarms](#_bookmark222)
* [Clearing Acknowledging or Unacknowledging a Single Alarm](#_bookmark223)
* [Alarm Count Per Perceived Severity](#_bookmark224)
* [Events Replay](#_bookmark225)
  + - 1. Get Alarms

This API is used to retrieve the alarms identified by alarmAckState, if provided, as query parameter otherwise retrieve all alarms.

**URL** /api/v1/fmaas/alarms **HTTP Method** GET **Request Parameters**

"name": "alarmAckState", "in": "query", "required": false, "type": "string", "enum": [

"allActiveAlarms", "allActiveAndAcknowledgedAlarms", "allActiveAndUnacknowledgedAlarms"

]

Request Body N/A

Response

200 OK

Success: List of alarms in VES structure is returned. Error: Error response in case of any error

{

"error" : {

"errorReason" : "<error message>"

}

}

* + - 1. Comment Multiple Alarms

This API is used to add a comment to multiple alarms.

**URL** /api/v1/fmaas/alarms

**HTTP Method** POST

**Content-Type** application/json

Request Parameters

"name": "alarmId",

"in": "query",

"description":"Identifies the alarms to which the comment shall be added", "required": true,

"schema": {

"type": "array", "items": { <alarmID>}

}

Request Body

"data": {

"commentTime":"",

"commentText":"",

"commentUserId":"", "commentSystemId":""

}

Response

200 OK

Success:

"data": {

"commentTime":"",

"commentText":"",

"commentUserId":"", "commentSystemId":""

}

Error Response:

{

"error" : [

{

"alarmId" : "",

"errorReason":""

}

]

}

* + - 1. Comment a Single Alarm

This API is used to add a comment to a single alarm.

**URL** /api/v1/fmaas/alarms/{alarmId}/comment

**HTTP Method** POST

**Content-Type** application/json

Request Parameters

"name": "alarmId",

"in": "path",

"description": "Identifies the alarm to which the comment shall be added.", "required": true

Request Body

"data": {

"commentTime":"",

"commentText":"",

"commentUserId":"", "commentSystemId":""

}

Response

200 OK

Success:

"data": {

"commentTime":"",

"commentText":"",

"commentUserId":"", "commentSystemId":""

}

Error Response:

{

"error" : [

{

"alarmId" : "",

"errorReason":""

}

]

}

* + - 1. **Clearing Acknowledging or Unacknowledging Multiple Alarms** This API is used to clear, acknowledge, or unacknowledge multiple alarms. **URL** /api/v1/fmaas/alarms

**HTTP Method** PATCH **Content-Type** application/json **Request Parameters**

"name": "alarmId",

"in": "query",

"description": "Identifies the alarms to be patched.", "required": true,

"schema": {

"type": "array", "items": { <alarmID>}

}

Request Body

"schema": {

"oneOf": [

{

"ackUserId": "",

"ackSystemId": "", "ackstate": "acknowledged"

},

{

"ackUserId": "",

"ackSystemId": "", "ackstate": "unacknowledged"

}

Or

{

"clearUserId":"",

"clearSystemId":"", "perceivedSeverity": "cleared"

}

]

}

"required": true

Response

200 OK

Success: In case of success, the response body shall be empty. Error Response:

{

"error" : [

{

"alarmId" : "",

"errorReason":""

}

]

}

* + - 1. **Clearing Acknowledging or Unacknowledging a Single Alarm** This API is used to clear, acknowledge, or unacknowledge a single alarm. **URL** /api/v1/fmaas/alarms/{alarmId}

**HTTP Method** PATCH Content-Type application/json **Request Parameters**

"name": "alarmId",

"in": "path",

"description": "Identifies the alarm to be patched.", "required": true

Request Body/Schema

"schema": {

"oneOf": [

{

"ackUserId": "",

"ackSystemId": "", "ackstate": "acknowledged"

},

{

"ackUserId": "",

"ackSystemId": "", "ackstate": "unacknowledged"

}

Or

{

"clearUserId":"",

"clearSystemId":"", "perceivedSeverity": "cleared"

}

]

}

"required": true

Response

200 OK

Success: In case of success, the response body shall be empty. Error Response:

{

"error" : [

{

"alarmId" : "",

"errorReason":""

}

]

}

* + - 1. Alarm Count Per Perceived Severity

This API gets the alarm count per perceived severity.

**URL** /api/v1/fmaas/alarms/$alarmsCount

HTTP Method GET

**Content-Type** application/json

Request Parameters

"name": "alarmAckState", "in": "query",

"type": "string", "enum": [

"allActiveAlarms", "allActiveAndAcknowledgedAlarms","allActiveAndUnacknowl- edgedAlarms"

]

**Request Body** N/A **Response** 200 OK

Success: The alarm count per perceived severity is returned.

* + - 1. Events Replay

This API initiates a request to replay the events. The following domain events are available for replay:

* Fault
* ThresholdCrossingAlerts
* Notification

Heartbeat and measurement events are not available for replay using this API, and heartbeat always has sequence 0.

The following two versions are available for this API:

* [Version 1](#_bookmark226)
* [Version 2](#_bookmark227)
  + - * 1. Version 1

**URL** /api/v1/fmaas/events/replay/{start}/{end}

HTTP Method GET

**Content-Type** application/json

Request Parameters

1. "name": "start", "in": "path", "required": true, "type": "int"
2. "name": "end", "in": "path", "required": false, "type": "int"

**Note:** The start and end parameters are the alarm sequence and are inclusive.

**Request Body** N/A **Response** 200 OK

Success:

{

"success" :

{

"message":"Event replay initiated"

}

}

Error Response:

{

"error" :

{

"errorReason":""

}

}

**Note:** This API will be deprecated in the future.

* + - * 1. Version 2

**URL** /api/v2/fmaas/events/replay/{collectorId}/{start}/{end}

HTTP Method GET

**Content-Type** application/json

Request Parameters

1. "name": "collectorId", "in": "path", "required": true, "type": "int
2. "name": "start", "in": "path", "required": true, "type": "int"

3) "name": "end", "in": "path", "required": false, "type": "int"

**Note:** The start and end parameters are the alarm sequence and are inclusive.

**Request Body** N/A **Response** Success:

HTTP Code-202

Message- Replay Initiated for collectorId: <collectorId> Failure:

1. In case of no events found for requested replay HTTP Code- 204

Message- N/A

1. In case of connection down between VESGW and Collector HTTP Code- 406

Message- Connection Down between VESGW and Collector for collectorId:

<collectorId>.

1. In case of Replay already in progress HTTP Code-406

Message- Replay Already in Process for collectorId: <collectorId>

1. In case of end sequence less than start sequence HTTP Code- 406

Message- End sequence number cannot be smaller than Start Sequence

1. In case of Internal server error HTTP Code- 503

Message- Cannot Process Request right now, try later!!

# XGVela Events

This section describes all XGVela events grouped based on the domain. The following table describes the Notification domain events:

Table 52: Notification Domain Events

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| NFServiceInstanceCreated | INFO | TOPOLOGY | N/A | This event is generated when an NF service instance is created in the topology. | N/A |
| NFServiceInstanceStateChanged | INFO | TOPOLOGY | N/A | This event is generated when an NF service instance state change happens in the topology. | N/A |
|  |  |  |  | The following are the change types: |  |
|  |  |  |  | * state: Normal state transition. The following are the states for this change type: |  |
|  |  |  |  | * NULL |  |
|  |  |  |  | * READY |  |
|  |  |  |  | * NOT\_READY |  |
| NFServiceIn- | INFO | TOPOLOGY | N/A | This event is gener- | N/A |
| stanceDeleted |  |  |  | ated when an NF ser- |  |
|  |  |  |  | vice instance is deleted |  |
|  |  |  |  | from the topology. |  |
| NFServiceCreated | INFO | TOPOLOGY | N/A | This event is generated when an NF service is created in the topology. | N/A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| NFServiceStateChanged | INFO | TOPOLOGY | N/A | This event is generated when an NF service state change happens in the topology. | N/A |
|  |  |  |  | The following are the possible values of states: |  |
|  |  |  |  | * NULL |  |
|  |  |  |  | * INSTANTIATED\_NOT\_CONFIGURED |  |
|  |  |  |  | * INSTANTIATED\_CONFIGURED\_INACTIVE |  |
|  |  |  |  | * INSTANTIATED\_CONFIGURED\_ACTIVE |  |
|  |  |  |  | * TERMINATED |  |
| NFServiceDelet- | INFO | TOPOLOGY | N/A | This event is gener- | N/A |
| ed |  |  |  | ated when an NF ser- |  |
|  |  |  |  | vice is deleted from the |  |
|  |  |  |  | topology. |  |
| NFServiceUpgradeStarted | INFO | TOPOLOGY | N/A | This event is generated when NF Service upgrade starts. | N/A |
| NFServiceUpgradeCompleted | INFO | TOPOLOGY | N/A | This event is generated when NF Service upgrade is completed. | N/A |
| NetworkFunctionCreated | INFO | TOPOLOGY | N/A | This event is generated when an NF is created in the topology. | N/A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| NetworkFunctionStateChanged | INFO | TOPOLOGY | N/A | This event is generated when an NF state change happens in the topology. | N/A |
|  |  |  |  | The following are the possible values of states: |  |
|  |  |  |  | * NULL |  |
|  |  |  |  | * INSTANTIATED\_NOT\_CONFIGURED |  |
|  |  |  |  | * INSTANTIATED\_CONFIGURED\_INACTIVE |  |
|  |  |  |  | * INSTANTIATED\_CONFIGURED\_ACTIVE |  |
|  |  |  |  | * TERMINATED |  |
| NetworkFunction- | INFO | TOPOLOGY | N/A | This event is gener- | N/A |
| Deleted |  |  |  | ated when an NF is |  |
|  |  |  |  | deleted from the topol- |  |
|  |  |  |  | ogy. |  |
| NetworkFunctionUpgradeStarted | INFO | TOPOLOGY | N/A | This event is generated when NF upgrade starts. | N/A |
| NetworkFunctionUpgradeCompleted | INFO | TOPOLOGY | N/A | This event is generated when NF upgrade is completed. | N/A |
| OperationsSpecModified | INFO | OPERATION | N/A | This event is generated when an Operations Specification on an NF is created, modified  or deleted. This event carries all the common and base properties including the ManagedObject indicating the NF object instance on which the Operations specification was updated. | N/A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| OperationStateChanged | INFO | OPERATION | N/A | This event is generated upon operation state change. This event is of type state notification. | N/A |
|  |  |  |  | The event has the following additional properties: |  |
|  |  |  |  | * transactionId |  |
|  |  |  |  | * operationId |  |
|  |  |  |  | * target |  |
|  |  |  |  | * createTime |  |
|  |  |  |  | * updateTime |  |
|  |  |  |  | * status |  |
| NFMgmtIntfReady | INFO | LOAD | N/A | Indicates that the management interface is ready for a particular NF to accept configuration changes. | N/A |
| CmaasAuditFailure | INFO | AUDITING | N/A | Indicates audit failure for a pod during dynamic updates. | N/A |
| CmaasConfigUpdateReceived | INFO | UPDATE | N/A | Indicates that a transaction is commit-  ted successfully in ConfD. | N/A |
| CmaasConfigmapUpdateSuccess | INFO | UPDATE | N/A | Indicates that the ConfigMap corresponding to a ConfD namespace is successfully updated. | N/A |
| CmaasChangeSet- | INFO | AUDIT | N/A | Indicates that a | N/A |
| PushSuccess |  |  |  | change-set key is suc- |  |
|  |  |  |  | cessfully pushed into |  |
|  |  |  |  | Etcd for dynamic up- |  |
|  |  |  |  | dates. |  |
| CmaasConfigDataLoadSuccess | INFO | LOAD | N/A | Indicates that a JSON configuration is successfully loaded to its ConfD namespace. | N/A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| CmaasSelfConfigUpdateAttempted | INFO | LOAD | N/A | Indicates that CMaaS config update has started. | N/A |
| CmaasSelfConfigUpdateSuccess | INFO | CONFIG\_UPDATE | N/A | Indicates that CMaaS config update is completed successfully. | N/A |
| NFMgmtIntfChanged | INFO | LOAD | N/A | Indicates that the management interface is upgraded successfully for a particular NF. | N/A |
| FmaasConfigUpdateAttempted | INFO | CONFIG\_UPDATE | N/A | This event indicates that FMaaS configuration update was attempted. | N/A |
| FmaasConfigUpdateSuccess | INFO | CONFIG\_UPDATE | N/A | This event indicates that FMaaS configuration update is successful. | N/A |

The following table describes the Fault domain events:

Table 53: Fault Domain Events

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| CmaasDataProviderFailure | CRITICAL | COMMUNICATION | CmaasDataProviderFailure with CLEAR severity. | Indicates that the data provider connection has failed. | No specific action required. Con-  fig-service restarts self and recovers. |
| CmaasConfigDataLoadFailure | CRITICAL | LOAD | CmaasConfigDataLoadSuccess | Indicates a failure in loading a JSON configuration to its ConfD namespace. | Redeploy the NF with the correct configuration. |
| CmaasConfigDataLoadSuccess | CLEAR | CLEAR | N/A | Clears the CmaasConfigDataLoadFailure event. | N/A. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| CmaasConfigModelCompileFailure | CRITICAL | COMPILE | CmaasConfigModelCompileSuccess | Indicates a failure in compiling a YANG mode | Redeploy the NF with the valid YANG model. |
| CmaasConfigModelCompileSuccess | CLEAR | CLEAR | N/A | Clears the CmaasConfigModelCompileFailure event. | N/A |
| CmaasSchemasLoadFailure | CRITICAL | LOAD | CmaasSchemasLo Success | aIndd-icates a failure in upgrading the ConfD schema container. | Redeploy the NF with the valid YANG model. |
| CmaasSchemasLoadSuccess | CLEAR | CLEAR | N/A | Clears the CmaasSchemasLoa Failure event. | N/A d- |
| PrometheusConnectivityDown | WARNING | COMMUNICATION | PrometheusConnectivityUp | This event is generated when the connectivity from VES Gateway to Prometheus is down. | Check Prometheus pod or connectivity status. |
| PrometheusConnectivityUp | CLEAR | COMMUNICATION | N/A | This event is generated when the connectivity from VES Gateway to Prometheus is up. | N/A |
| CollectorDown | WARNING | COMMUNICATION | CollectorUp | This event is generated when the connectivity from VES Gateway to collector is down. | Check collector status. |
| CollectorUp | CLEAR | COMMUNICATION | N/A | This event is generated when the connectivity from VES Gateway to collector is up. | N/A |

The following table describes the TCA domain events:

Table 54: TCA Domain Events

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| SystemOut- | WARNING | HostOS | Check | SystemOutOfDiskSpace | This event |
| OfDiskSpace |  |  | and allo- | with CLEAR severity | is triggered |
|  |  |  | cate suf- |  | if the disk |
|  |  |  | ficient |  | space us- |
|  |  |  | disk. The |  | age is greater |
|  |  |  | alarm |  | than the de- |
|  |  |  | is au- |  | fined thresh- |
|  |  |  | to-cleared |  | old for one |
|  |  |  | when the |  | minute. |
|  |  |  | used disk |  |  |
|  |  |  | space |  |  |
|  |  |  | goes be- |  |  |
|  |  |  | low the |  |  |
|  |  |  | thresh- |  |  |
|  |  |  | old. |  |  |
| SystemOut- | WARNING | HostOS | Check | SystemOutOfMemory | This event |
| OfMemory |  |  | and rec- | with CLEAR severity. | is triggered |
|  |  |  | tify mem- |  | if the mem- |
|  |  |  | ory over- |  | ory usage |
|  |  |  | load |  | for a node is |
|  |  |  | condi- |  | greater than |
|  |  |  | tion. The |  | the defined |
|  |  |  | alarm |  | threshold for |
|  |  |  | is au- |  | one minute. |
|  |  |  | to-cleared |  |  |
|  |  |  | when the |  |  |
|  |  |  | available |  |  |
|  |  |  | memo- |  |  |
|  |  |  | ry goes |  |  |
|  |  |  | above |  |  |
|  |  |  | the |  |  |
|  |  |  | thresh- |  |  |
|  |  |  | old. |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event Name** | **Severity** | **Event Type** | **Clear Event** | **Description** | **Repair Action** |
| SystemHigh- | WARNING | HostOS | Check | SystemHighCpuLoad | This event |
| CpuLoad |  |  | and rec- | with CLEAR severity | is triggered |
|  |  |  | tify the |  | if the CPU |
|  |  |  | CPU |  | usage for |
|  |  |  | overload |  | a node is |
|  |  |  | condi- |  | greater than |
|  |  |  | tion. The |  | the defined |
|  |  |  | alarm |  | threshold for |
|  |  |  | is au- |  | one minute. |
|  |  |  | to-cleared |  |  |
|  |  |  | when |  |  |
|  |  |  | the avail- |  |  |
|  |  |  | able CPU |  |  |
|  |  |  | goes |  |  |
|  |  |  | above |  |  |
|  |  |  | the |  |  |
|  |  |  | thresh- |  |  |
|  |  |  | old. |  |  |

# Annexure A

This section provides samples of events. Refer to the following sections:

###### [Fault\_XGVela\_FmaasConfigUpdateFailed](#_bookmark233)

###### [Notification\_XGVela\_NFServiceInstanceCreated](#_bookmark234)

###### [Notification\_XGVela\_NFServiceInstanceDeleted](#_bookmark235)

###### [Notification\_XGVela\_NetworkFunctionStateChanged](#_bookmark236)

###### [Mvfs\_xgvela1\_Measurements](#_bookmark237)

###### [heartbeat\_xgvela1](#_bookmark238)

###### [NFMgmtIntfReady](#_bookmark239)

## Fault\_XGVela\_FmaasConfigUpdateFailed

This fault event is generated when there is a failure in FMaaS config update.

{

"event": {

"commonEventHeader": { "domain": "fault",

"eventId": "fcc0e8ab3f61ef154e738858ef8769e8",

"eventName": "Fault\_XGVela-Mavenir\_FmaasConfigUpdateFailed", "eventType": "CONFIGUPDATE",

"lastEpochMicrosec": 1587112393881000, "nfNamingCode": "xgvela", "nfcNamingCode": "fault-service", "priority": "Normal",

"reportingEntityId": "f78ae0a3-3411-3810-9c2e-c92a3f124448", "reportingEntityName":

"mvnr,ManagedElement=me-xgvela1,NetworkFunction=xgvela1, NFService=fault-service,NFServiceInstance=fault-service

-795947f6-hwmv9",

"sequence": 100,

"sourceId": "15a4cfcc-f1a3-33d7-8664-238edbfccaa2", "sourceName": "mvnr,ManagedElement=me-xgvela1,NetworkFunction=

xgvela1,NFService=fault-service,NFServiceInstance=fault

-service-795947f6-hwmv9, ContainerId=1db619e34267ea97b5442bdd3d950935167c8442a 769a4231b305d61d58bb31f,

change-set-key=change-set/xgvela1

/fault-service/v1/ms/fmaas.json/1", "startEpochMicrosec": 1587112393881000,

"version": "4.1",

"vesEventListenerVersion": "7.1"

},

"faultFields": { "alarmAdditionalInformation": {

"additionalText": "FMaaS Config Update Failed", "meId": "47269bda-c715-3892-acd4-6941707705d4",

"nfId": "62b851cc-f5c3-358d-94f3-1f533e938c9c", "nfServiceID": "ac9dae27-885b-348e-b1cb-fa32b87a19be",

"nfServiceInstanceId": "f78ae0a3-3411-3810-9c2e-c92a3f124448", "probableCause": "FMaaS Config Update Failed", "proposedRepairAction": "Verify the configuration and update"

},

"alarmCondition": "Fault\_XGVela-Mavenir\_FmaasConfigUpdateFailed", "eventCategory": "CONFIGUPDATE",

"eventSeverity": "WARNING", "eventSourceType": "xgvela", "faultFieldsVersion": "4.0", "specificProblem": "", "vfStatus": "Active"

}

}

}

## Notification\_XGVela\_NFServiceInstanceCreated

This notification is generated when an NF service instance is created in the topology.

{

"event": {

"commonEventHeader": { "domain": "notification",

"eventId": "5686986538b805447f56ca1c548a4984",

"eventName": "Notification\_XGVela-Mvnr\_NFServiceInstanceCreat-

ed",

"localEventName": "NFServiceInstanceCreated", "eventType": "TOPOLOGY",

"lastEpochMillis": 1612783140141, "nfcNamingCode": "config-service", "nfNamingCode": "xgvela", "priority": "Normal",

"reportingEntityId": "ac9dae27-885b-348e-b1cb-fa32b87a19be", "reportingEntityName": "mvnr,ManagedElement=me-xgvela1,Network-

Function=xgvela1,NFService=fault-service", "sequence": 0,

"sourceId": "f35bbe06-b057-385d-9d69-eaa30037ce78", "sourceName": "mvnr,ManagedElement=me-xgvela1,NetworkFunc-

tion=xgvela1,NFService=config-service,NFServiceInstance=config-service-766ccd487b-dplbk",

"startEpochMillis": 1612783140141,

"version": "4.1",

"vesEventListenerVersion": "7.1"

},

"notificationFields": { "additionalFields": {

"nfId": "62b851cc-f5c3-358d-94f3-1f533e938c9c", "node.label.paas": "true",

"meLabel": "mvnr,ManagedElement=me-xgvela1", "nfServiceName": "config-service",

"nfLabel": "mvnr,ManagedElement=me-xgvela1,NetworkFunc-

tion=xgvela1",

"nfServiceSwVersion": "v2.0", "serviceTypeAffected": "NONE", "pod.label.microSvcName": "config-service",

"node.label.feature.node.kubernetes.io/system-os\_re-

lease.VERSION\_ID": "7",

"svcVersion": "v2.0",

"nfName": "xgvela1",

"probableCause": "NF Service Instance created", "pod\_uuid": "91962e0b-69e6-11eb-960c-fa163ec901ef",

"nfServiceInstanceName": "config-service-766ccd487b-dplbk",

eaa30037ce78",

"node.label.beta.kubernetes.io/arch": "amd64", "node.label.kubernetes.io/arch": "amd64", "meId": "47269bda-c715-3892-acd4-6941707705d4",

"alarmHierarchy": "Independent", "pod.label.pod-template-hash": "766ccd487b", "pod.label.app": "config-service",

"nfType": "xgvela",

"node.label.appln": "true", "nfServiceInstanceId": "f35bbe06-b057-385d-9d69-

"node.label.node-role.kubernetes.io/master": "", "nfServiceInstanceLabel": "mvnr,ManagedElemen-

t=me-xgvela1,NetworkFunction=xgvela1,NFService=config-service,NFServiceInstance=config-service-766ccd487b-dplbk",

"classification": "Notification", "nfSwVersion": "v1",

"node.label.infra": "true", "node.label.kubernetes.io/os": "linux", "node.label.mgmt": "true",

"alarmType": "",

"node.label.caas": "true", "pod.label.xgvelaId": "xgvela1",

"nfServiceLabel": "mvnr,ManagedElement=me-xgvela1,NetworkFunction=xgvela1,NFService=config-service",

"nfServiceType": "config-service", "node.label.beta.kubernetes.io/os": "linux", "nfServiceId": "9d0598b3-c646-3435-b7e9-27cad51c82dd", "node.label.kubernetes.io/hostname": "k8s-standalone"

},

"changeIdentifier": "f35bbe06-b057-385d-9d69-eaa30037ce78", "changeType": "NFServiceInstanceCreated", "notificationFieldsVersion": "2.0"

}

}

}

## Notification\_XGVela\_NFServiceInstanceDeleted

This notification is generated when an NF service instance in the topology is deleted.

{

"event": {

"commonEventHeader": { "domain": "notification",

"eventId": "1706464180a716c343c9a731e45befed",

"eventName": "Notification\_XGVela-Mvnr\_NFServiceInstanceDelet-

ed",

"localEventName": "NFServiceInstanceDeleted", "eventType": "TOPOLOGY",

"lastEpochMillis": 1612783171827, "nfcNamingCode": "config-service", "nfNamingCode": "xgvela", "priority": "Normal",

"reportingEntityId": "ac9dae27-885b-348e-b1cb-fa32b87a19be", "reportingEntityName": "mvnr,ManagedElement=me-xgvela1,Network-

Function=xgvela1,NFService=fault-service", "sequence": 0,

"sourceId": "d579b7b8-9d05-35e8-b8a4-0461a6b4da53",

"sourceName": "mvnr,ManagedElement=me-xgvela1,NetworkFunction=xgvela1,NFService=config-service,NFServiceInstance=config-service-766c- cd487b-t9brj",

"startEpochMillis": 1612783171827,

"version": "4.1",

"vesEventListenerVersion": "7.1"

},

"notificationFields": { "additionalFields": {

"nfId": "62b851cc-f5c3-358d-94f3-1f533e938c9c", "node.label.paas": "true",

"meLabel": "mvnr,ManagedElement=me-xgvela1", "nfServiceName": "config-service",

"nfLabel": "mvnr,ManagedElement=me-xgvela1,NetworkFunc-

tion=xgvela1",

"nfServiceSwVersion": "v2.0", "serviceTypeAffected": "NONE", "pod.label.microSvcName": "config-service",

"node.label.feature.node.kubernetes.io/system-os\_re-

lease.VERSION\_ID": "7",

"svcVersion": "v2.0",

"nfName": "xgvela1",

"probableCause": "NF Service Instance deleted", "pod\_uuid": "91962e0b-69e6-11eb-960c-fa163ec901ef",

"nfServiceInstanceName": "config-service-766ccd487b-t9brj", "node.label.beta.kubernetes.io/arch": "amd64", "node.label.kubernetes.io/arch": "amd64",

"meId": "47269bda-c715-3892-acd4-6941707705d4",

"alarmHierarchy": "Independent", "pod.label.pod-template-hash": "766ccd487b", "pod.label.app": "config-service",

"nfType": "xgvela",

"node.label.appln": "true", "nfServiceInstanceId": "d579b7b8-9d05-35e8-

b8a4-0461a6b4da53",

"node.label.node-role.kubernetes.io/master": "", "nfServiceInstanceLabel": "mvnr,ManagedElemen-

t=me-xgvela1,NetworkFunction=xgvela1,NFService=config-service,NFServiceInstance=config-service-766ccd487b-t9brj",

"classification": "Notification", "nfSwVersion": "v1",

"node.label.infra": "true", "node.label.kubernetes.io/os": "linux", "node.label.mgmt": "true",

"alarmType": "",

"node.label.caas": "true", "pod.label.xgvelaId": "xgvela1",

"nfServiceLabel": "mvnr,ManagedElement=me-xgvela1,Network- Function=xgvela1,NFService=config-service",

"nfServiceType": "config-service", "node.label.beta.kubernetes.io/os": "linux", "nfServiceId": "9d0598b3-c646-3435-b7e9-27cad51c82dd", "node.label.kubernetes.io/hostname": "k8s-standalone"

},

"changeIdentifier": "d579b7b8-9d05-35e8-b8a4-0461a6b4da53", "changeType": "NFServiceInstanceDeleted",

"newState": "TERMINATED",

"oldState": "NOT\_READY", "notificationFieldsVersion": "2.0"

}

}

}

## Notification\_XGVela\_NetworkFunctionStateChanged

This notification is generated when there is a state change for an NF

{

"event": {

"commonEventHeader": { "domain": "notification",

"eventId": "7985cbd598785d62384277aba913cd8e", "eventName": "Notification\_XGVela-Mavenir

\_NetworkFunctionStateChanged", "eventType": "TOPOLOGY", "lastEpochMicrosec": 1586961319744000, "nfNamingCode": "amf",

"priority": "Normal",

"reportingEntityId": "457a0d64-b05a-3fbb-a46d-25ec18a47a18", "reportingEntityName": "mvnr,ManagedElement=me-xgvela1,

NetworkFunction=xgvela1,NFService=topo-engine, NFServiceInstance=topo-engine-684548459d-m646s",

"sequence": 4,

"sourceId": "c0b1f989-ab2a-3679-b2d1-7f6b407aa253", "sourceName": "mvnr,ManagedElement=me-xgvela1,

NetworkFunction=amf1", "startEpochMicrosec": 1586961319744000,

"version": "4.1",

"vesEventListenerVersion": "7.1"

},

"notificationFields": { "additionalFields": {

"meId": "47269bda-c715-3892-acd4-6941707705d4",

"nfId": "c0b1f989-ab2a-3679-b2d1-7f6b407aa253" "meLabel": "mvnr,ManagedElement=me-xgvela1”,

"nfLabel": "mvnr,ManagedElement=me-xgvela1,NetworkFunc-

tion=amf1”,

}

"nfType": "amf”,

"nfSwVersion": "1.0”

},

"changeIdentifier": "c0b1f989-ab2a-3679-b2d1-7f6b407aa253", "changeType": "NetworkFunctionStateChanged",

"newState": "INSTANTIATED\_CONFIGURED\_ACTIVE",

"notificationFieldsVersion": "2.0", "oldState": "INSTANTIATED\_NOT\_CONFIGURED"

}

}

## Mvfs\_xgvela1\_Measurements

This notification is generated for application specific metrics or custom metrics.

{

"eventList": [

{

"commonEventHeader": { "domain": "measurement",

"eventId": "Measurements0000000001", "eventName": "Mvfs\_amf\_Measurements", "eventType": "mvnr-xgvela-appln-amf-amf1", "lastEpochMicrosec": 1589390070000000, "nfNamingCode": "amf",

"nfcNamingCode": "test-app1", "priority": "Normal",

"reportingEntityId": "fd4e93b1-5422-3e47-85f4-05ca9df75f75", "reportingEntityName": "mvnr,ManagedElement=me-xgvela,

NetworkFunction=amf1,NFService=test-app1", "sequence": 0,

"sourceId": "6fe97221-9242-38a9-b80f-065a299b35c8", "sourceName": "mvnr,ManagedElement=me-xgvela,

NetworkFunction=amf1,NFService=test-app1", "startEpochMicrosec": 1589390040000000,

"version": "4.1",

"vesEventListenerVersion": "7.1"

},

"measurementFields": { "additionalObjects": [

{

"objectInstances": [

{

"objectInstance": { "ActiveSessions": 500,

"TotalRequest": 250

},

"objectKeys": [

{

"keyName": "CUId",

"keyValue": "CU0"

}

]

},

{

"objectInstance": { "ActiveSessions": 1000,

"TotalRequest": 500

},

"objectKeys": [

{

"keyName": "CUId",

"keyValue": "CU1"

}

]

}

],

"objectName": "CUMetrics"

}

],

"measurementFieldsVersion": "4.0",

"measurementInterval": 30

}

}

]

}

## heartbeat\_xgvela1

This event is generated for the heartbeat sent between the VES Gateway and collector.

{

"commonEventHeader":{ "domain":"heartbeat", "eventId":"heartbeat0000000141", "eventName":"heartbeat\_cim-mvnr", "eventType":"HEARTBEAT", "lastEpochMicrosec":1612441020013659, "nfNamingCode":"cim", "nfVendorName":"mvnr", "priority":"Normal",

"reportingEntityId":"e819b318-92f9-3300-995a-9ac8214ae8bb", "reportingEntityName":"mvnr,ManagedElement=me-xgvela1,NetworkFunc-

tion=xgvela1",

"sequence":0,

"sourceId":"f89d3b69-90b2-3a31-bd4c-708dda0a00a8", "sourceName":"mvnr,ManagedElement=me-xgvela1,NetworkFunction=tpaas", "startEpochMicrosec":1612441020013659,

"version":"4.1", "vesEventListenerVersion":"7.1"

},

"heartbeatFields":{ "additionalFields":{

"ME\_ID":"47269bda-c715-3892-acd4-6941707705d4", "MGMT\_IP":"10.1.34.134", "CMAAS\_NETCONF\_PORT":"32767", "APIGW\_PORT":"32640", "FMAAS\_HTTP\_PORT":"32666", "APIGW\_USERNAME":"admin", "APIGW\_PASSWORD":"admin", "APIGW\_AUTH\_TYPE":"basic",

"COLLECTOR\_ID":"2"

},

"heartbeatFieldsVersion":"3.0", "heartbeatInterval":60

}

}

## NFMgmtIntfReady

The following is an example of the NFMgmtIntfReady event:

{

"event": {

"commonEventHeader": { "domain": "notification",

"eventId": "c4975e64b08995bd3e1d514849c7992e", "eventName": "Notification\_XGVela-Mvnr\_NFMgmtIntfReady", "localEventName": "NFMgmtIntfReady",

"eventType": "LOAD", "lastEpochMillis": 1613559667833, "nfcNamingCode": "", "nfNamingCode": "udsf", "priority": "Normal",

"reportingEntityId": "de89b0ca-ddcc-36b6-98fa-013902b7cadc", "reportingEntityName": "mvnr2,ManagedElement=me-xgvela1,NetworkFunc-

tion=xgvela1,NFService=fault-service", "sequence": 0,

"sourceId": "abddbb90-8290-3198-8780-ccff7c977be3",

"sourceName": "mvnr2,ManagedElement=me-xgvela1,NetworkFunction=uds-

f1",

},

"startEpochMillis": 1613559667833,

"version": "4.1",

"vesEventListenerVersion": "7.1"

"notificationFields": { "additionalFields": {

"nfId": "abddbb90-8290-3198-8780-ccff7c977be3", "meId": "07adb663-4387-3544-bf8b-35ea1324534c",

"meLabel": "mvnr2,ManagedElement=me-xgvela1",

"nfLabel": "mvnr2,ManagedElement=me-xgvela1,NetworkFunction=uds-

f1",

failures",

},

"serviceTypeAffected": "MULTIPLE", "alarmHierarchy": "Independent", "nfType": "udsf", "classification": "Notification", "nfSwVersion": "v0",

"alarmType": "",

"nfName": "udsf1",

"probableCause": "Nf Day-0 configuration loaded without any "pod\_uuid": "5b3b3743-ee21-4997-b5f7-5ba09cc19839"

"changeIdentifier": "abddbb90-8290-3198-8780-ccff7c977be3", "changeType": "NFMgmtIntfReady", "notificationFieldsVersion": "2.0"

}

}

}

# Annexure B

This section provides samples for API Gateway interfaces. Refer to the following sections:

###### [NFService By Name](#_bookmark241)

###### [NFServiceInstance by ID](#_bookmark242)

## NFService By Name

{

"tailf-rest-query:query-result": { "result": [

{

"select": [

{

"data": {

"mcf-nrm-managed-element:NFService": {

"id": "a428d140-84ac-380d-8ca0-f588e9198b4b",

"attributes": {

"name": "test-demo-app", "userLabel": "mvnr,ManagedElemen-

t=me-xgvela1,NetworkFunction=overlayha1,NFService=test-demo-app",

"nfServiceType": "test-demo-app", "swVersion": "v1",

"state": "INSTANTIATED\_CONFIGURED\_ACTIVE",

"administrativeState": "UNLOCKED", "operationalState": "ENABLED", "usageState": "ACTIVE",

},

"NFServiceInstance": [

{

"id": "307caf5c-7761-314d-95a1-1fa0619702d7",

"attributes": {

"name": "test-demo-app-7955c44c9d-g7fhj", "userLabel": "mvnr,ManagedElemen-

t=me-xgvela1,NetworkFunction=overlayha1,NFService=test-demo-app,NFServiceIn- stance=test-demo-app-7955c44c9d-g7fhj",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.84"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::2"]

},

{

"name": "macvlan-hostlocal-2",

"interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::f"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.169"],

}

]

}

},

{

"id": "b650121a-b08f-3b1b-91e1-85fa161a6dbb", "attributes": {

"name": "test-demo-app-7955c44c9d-g4bcw", "userLabel": "mvnr,ManagedElemen-

t=me-xgvela1,NetworkFunction=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-app-7955c44c9d-g4bcw",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.122"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.175"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::9"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::16"]

}

]

}

},

{

"id": "34e95e1e-2939-315d-b84b-11e53d88c89b",

"attributes": {

"name": "test-demo-app-7955c44c9d-crr6n", "userLabel": "mvnr,ManagedElement=me-xgvela1,Net-

workFunction=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-app-7955c44c9d-crr6n",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0",

"default": true,

"ips": ["192.168.0.101"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::7"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::14"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.173"],

}

]

}

},

{

"id": "82be6d91-3b01-3977-96a2-b21a2b57fc6d",

"attributes": {

"name": "test-demo-app-7955c44c9d-xl5cb", "userLabel": "mvnr,ManagedElement=me-xgvela1,Net-

workFunction=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-app-7955c44c9d-xl5cb",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.82"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::1"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::b"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.167"],

}

]

}

},

{

"id": "95ab5f46-4d00-33d0-806c-90b56ded27d8",

"attributes": {

"name": "test-demo-app-7955c44c9d-nrpd8", "userLabel": "mvnr,ManagedElement=me-xgvela1,Network-

Function=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-app-7955c44c9d-nrpd8",

"state": "READY", "networkList": [

{

f:fffd"]

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.78"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f8:ffff:ffff:fff-

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::9"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.163"],

}

]

}

},

{

"id": "95dad3f4-bf7a-37a6-952d-3fcecce48247", "attributes": {

"name": "test-demo-app-7955c44c9d-bz4kt", "userLabel": "mvnr,ManagedElement=me-xgvela1,Network-

Function=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-app-7955c44c9d-bz4kt",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.102"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.174"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::8"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::15"]

}

]

}

},

{

"id": "4b599a39-f39d-3780-929d-011849a3a508",

"attributes": {

"name": "test-demo-app-7955c44c9d-d44r2",

"userLabel": "mvnr,ManagedElement=me-xgvela1,NetworkFunction=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-ap-

p-7955c44c9d-d44r2",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.77"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f8:ffff:ffff:ffff:fffc"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::8"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.162"],

}

]

}

},

{

"id" "d45 b47 05 7 3 4f 8d2 f18 493048f9"

"id": "d45acb47-05e7-3a4f-8d2c-f18c493048f9", "attributes": {

"name": "test-demo-app-7955c44c9d-9947m", "userLabel": "mvnr,ManagedElement=me-xgvela1,Net-

workFunction=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-app-

7955c44c9d-9947m",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0",

"default": true,

"ips": ["192.168.0.83"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::3"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::10"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.168"],

}

]

}

},

{

"id": "c6f9087b-e8df-34c8-87db-2eb5748fd4a4", "attributes": {

"name": "test-demo-app-7955c44c9d-l6kvt", "userLabel": "mvnr,ManagedElement=me-xgvela1,Net-

workFunction=overlayha1,NFService=test-demo-app,NFServiceInstance=test-demo-app-7955c44c9d-l6kvt",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.90"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::6"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::13"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.172"],

}

]

}

},

{

"id": "945a0d00-d6a4-3d73-b135-804ddc6cc16b", "attributes": {

"name": "test-demo-app-7955c44c9d-xvftf", "userLabel": "mvnr,ManagedElement=me-xgvela1,Net-

workFunction=overlayha1,NFService=test-demo-app,NFServiceInstance=test-de- mo-app-7955c44c9d-xvftf",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.79"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f8:ffff:ffff:fff-

f:fffe"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::c"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.164"],

}

]

}

}

]

}

}

}

]

}

}

]

}

}

## NFServiceInstance by ID

{

"tailf-rest-query:query-result": { "result": [

{

"select": [

{

"data": {

"mcf-nrm-managed-element:NFServiceInstance": { "id": "307caf5c-7761-314d-95a1-1fa0619702d7",

"attributes": {

"name": "test-demo-app-7955c44c9d-g7fhj", "userLabel": "mvnr,ManagedElemen-

t=me-xgvela1,NetworkFunction=overlayha1,NFService=test-demo-app,NFSer- viceInstance=test- demo-ap-

p-7955c44c9dg7fhj",

"state": "READY", "networkList": [

{

"name": "k8s-pod-network",

"interface": "eth0", "default": true,

"ips": ["192.168.0.84"]

},

{

"name": "macvlan-hostlocal-1", "interface": "net2", "default": false,

"ips": ["2409:4073:381:77f9::2"]

},

{

"name": "macvlan-hostlocal-2", "interface": "net3", "default": false,

"ips": ["3ffe:ffff:1::f"]

},

{

"name": "macvlan-hostlocal-0", "interface": "net1", "default": false,

"ips": ["171.0.0.169"],

}

]

}

}

}

}

]

}

]

}

}

1. **Annexure C**

This section provides information about upgrading older versions of NF charts to chart template version

2.9. Release namespace support is available from template version 2.7 onwards. Complete the following steps to upgrade NF charts:

1. Modify the values.yaml file to incorporate the latest template definitions. Also copy folders for any new features.

######## CNF TEMPLATE DEFINITIONS ###########

global:

xgvela:

use\_release\_ns: false create\_ns: true

topogw\_fqdn: "topo-gw.xgvela1.svc.cluster.local:8080" #Correct the namespace part in topogw\_fqdn to match xgvela installation.

nf:

version: "<version>" vendorId: "mvnr" xgvelaId: "xgvela1"

nfClass: "<mgmt | appln | infra>" nfType: "<amf|smf|xgvela>"

nfId: "<nfId>" cnftpl\_configmap:

mgmt\_cfg:

create: false metaspec:

annotations:

configMgmt: enabled

static\_cfg:

create: false metaspec: {}

env\_cfg:

create: false metaspec: {}

cnftpl\_secret:

create: false metaspec: {} secretspec: {}

cnftpl\_namespaces: create: false metaspec: {}

cnftpl\_dashboard: create: false metaspec: {}

cnftpl\_eventdef: create: false metaspec: {}

cnftpl\_alertsdef: create: false metaspec: {}

cnftpl\_metricsdef: create: false

metaspec: {}

###########################################

use\_release\_ns : Indicates whether to use release namespace. Set this to false and **create\_ns** to true for XGVela template to create the namespace as per XGVela standards. The highlighted section of global variables is preferred to be set in the parent chart, and hence can be ignored for subcharts.

**version** : This identifies the NF/Service version. CMaaS adds this version internally as part of yang namespace and prefix during YANG onboarding procedure.

**cnftpl\_alertsdef** : This adds capability to support ThresholdCrossingAlerts (TCA). Set create:true to use the feature. alerts.yml file in the alerts folder should be used to define alert rules. Refer XGVela Developer Guide for sample.

**cnftpl\_metricsdef** : This adds capability to support metric definitions. Set create:true to use the feature.

1. Copy the \_cnf\_tpl\_def\_9.tpl and cnf\_tpl\_charts\_9.yaml files from templates folder of the chart template to the templates folder of the NF chart. Remove the files belonging to the older version.
2. Replace the cnf template usages in all the YAML files in the template to reflect the latest template version. Use the following command to replace it globally in a single execution:

find <path-chart-folder-to-upgrade> -type f -name '\*yaml' | xargs sed -i 's/

\_2\_8 /\_2\_9/g'

**Note:** This command upgrades the NF chart from version\_2\_8 to \_2\_9. Change the source version based on the version from which the NF chart is upgraded.

1. Refer to the Chart.yaml file from the template and copy the line apiVersion: v1 to the respective Chart.yaml file to add helm3 compatibility.

**Note:** Chart template 2.9 does not include nf.version in the configmap name. Update the deployment.yaml accordingly to refer to the correct configmap wherever used.

# 21. Annexure D

Follow these steps to upgrade applications from an older version of CIM to using the latest version:

1. Update the deployment.yaml file to reflect the new CIM image for the CIM container.
2. Copy the CIM configuration files (cim.json and cim.yang) from xgvela-sdk configs folder to the config/mgmt folder of the service chart which is using CIM. Refer to the XGVela Developer Guide and update necessary configuration settings based on NF requirement.
3. Copy the CIM event definition file (eventdef-cim.json) from the xgvela-sdk configs folder to the eventdef folder of the service chart. Set cnftpl\_eventdef create to true in the TEMPLATE VARIABLES section of the values.yaml file.

# Annexure F

The following is a sample VES Gateway Configuration for Collector Details and Domains:

"collectorDetails": [

{

"id": 1,

"primaryCollector": {

"fqdn": "xgvela-ves-simu.mvnr-xgvela1-mgmt-xgvela- xgvela1.svc.cluster.local",

"port": 8443, "secure": false, "user": "user",

"password": "pass", "passphrase": "mypassphrase", "nbiFormat": "VES",

"nbiType": "KAFKA",

"kafkaBrokers": ["kafka-svc.mvnr-xgvela1-infra-xgvela- xgvela1.svc.cluster.local:9092"],

"kafkaTopic": "xgvela-heartbeat-events", "heartbeat": true

}

},

],

"vesdomains": { "fault": {

"collectorList": [

{

"id": 1

}

]

},

"measurement": { "collectorList": [

{

"id": 1

}

]

},

"notification": { "collectorList": [

{

"id": 1

}

]

},

"tca": { "collectorList": [

{

"id": 1

}

]

}

}