A Holistic, Innovative Framework for the Design, Development and Orchestration of 5G-ready Applications and Network Services over Sliced Programmable Infrastructure

DELIVERABLE D1.3

VNF/PNF & VNF FORWARDING GRAPH METAMODEL

Due Date of Delivery:	M9 *Mx* (28/02/2018 *dd/mm/yyyy*)
Actual Date of Delivery:	06/03/2018 *dd/mm/yyyy*
Workpackage:	WP1 – MATILDA Reference Architecture, Conceptualization and Use Cases
Type of the Deliverable:	OTHER
Dissemination level:	PU
Editors:	UNIVBRIS, ATOS, NCSRD, AALTO, UBITECH
Version:	1.0

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

Call:
H2020-ICT-2016-2

Type of Action:
IA

Project Acronym:
MATILDA

Project ID:
761898

Duration:
30 months

Start Date:
01/01/201

Project Coordinator:
Name: Franco Davoli
Phone: +39 010 353 2732
Fax: +39 010 353 2154
e-mail: franco.davoli@cnit.it

Technical Coordinator
Name: Panagiotis Gouvas
Phone: +30 216 5000 503
Fax: +30 216 5000 599
e-mail: pgouvas@ubitech.eu

List of the Authors	
UNIVBRIS	UNIVERSITY OF BRISTOL
Anderson Bravalheri, Reza Nejabati, Dimitra Simeonidou	
ATOS	ATOS Spain SA
Aurora Ramos, Javier Melian	
NCSRD	NATIONAL CENTER FOR SCIENTIFIC RESEARCH “DEMOKRITOS”
Eleni Trouva	
AALTO	AALTO-KORKEAKOULUSÄÄTIÖ
Tarik Taleb, Afolabi Ibrahim, Bagaa Miloud	
UBITECH	GIOUMPITEK Meleti Schediasmos Ylopoiisi kai Polisi Ergon Pliroforikis EPE
Panagiotis Gouvas, Anastasios Zafeiropoulos	

Disclaimer

The information, documentation and figures available in this deliverable are written by the MATILDA Consortium partners under EC co-financing (project H2020-ICT-761898) and do not necessarily reflect the view of the European Commission.

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The reader uses the information at his/her sole risk and liability.

Copyright

The MATILDA Consortium consists of:

CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI (CNIT)

ATOS SPAIN SA (ATOS)

ERICSSON TELECOMUNICAZIONI (ERICSSON)

INTRASOFT INTERNATIONAL SA (INTRA)

COSMOTE KINITES TILEPIKOINONIES AE (COSM)

ORANGE ROMANIA SA (ORO)

EXXPERTSYSTEMS GMBH (EXXPERT)

GIOUMPITEK MELETI SCHEDIASMOS YLOPOIISI KAI POLISI ERGON PLIROFORIKIS ETAIREIA PERIORISMENIS EFTHYNIS (UBITECH)

INTERNET INSTITUTE, COMMUNICATIONS SOLUTIONS AND CONSULTING LTD (ININ)

INCELLIGENT IDIOTIKI KEFALAIOUCHIKI ETAIREIA (INC)

SUITE5 DATA INTELLIGENCE SOLUTIONS LIMITED (SUIT5)

NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS" (NCSRD)

UNIVERSITY OF BRISTOL (UNIVBRIS)

AALTO-KORKEAKOULUSAATIO (AALTO)

UNIVERSITY OF PIRAEUS RESEARCH CENTER (UPRC)

ITALTEL SPA (ITL)

BIBA - BREMER INSTITUT FUER PRODUKTION UND LOGISTIK GMBH (BIBA).

Table of Contents

1 Executive Summary

The MATILDA Project aims to provide a unified framework for designing, developing and orchestrating 5G-ready applications. Such ambitious framework must be tailored to respond to the expectations of different users, from application developers to infrastructure providers, which do not necessarily share the same domain knowledge, and may even have conflicting interests.

In this context, the development of a *lingua franca* is fundamental to allow all the users interact and collaborate to create a fruitful environment for not only technological development, but also new business creation.

In MATILDA, this is accomplished by defining **metamodels** for every message exchange. These metamodels work as templates that contain logical (and possibly nested) properties needed to organize the information about the deployment and operational behaviour requirements of services, components, applications etc in a unified way.

In particular, a subset of the MATILDA metamodels targets Network Services. As envisioned by the European Telecommunication Standards Institute (ETSI) [1], the metamodels related to Network Services are used in MATILDA to create a dynamic infrastructure, based on network functions virtualisation. The virtualisation of the resources enables ad-hoc provisioning and management of services, combined with extreme programmability as required in the upcoming 5G networks. Additionally, the resulting hardware and software decoupling is beneficial to decrease CAPEX/OPEX and increase market competitiveness.

MATILDA at the same time relies on and strengthens community efforts, by adopting the descriptors from the OSM community [2] as the basis for its network-related metamodels, whose main objective is to provide a uniform way to describe deployment and operational behaviour of network functions and network services and allow programmability in the lifecycle management of these components.

As defined by MATILDA architecture [3], they will allow developers, service providers and infrastructure providers to collaborate in software development by means of publishing software to the Marketplace. Moreover, MATILDA Slice Broker will use them to realize the slice intent issued by the Application Orchestrator by means of matching the network requirements compiled from the Application Graph and Component models with the Network Services present in the Telecommunication Service Provider catalogue.

This document presents an overall description of the metamodels required to specify a Network Service, more specifically, the Network Service Descriptor (NSD), Virtual Network Function Descriptor (VNFD), Physical Network Function Descriptor (PNFD), Virtual Network Function Forwarding Graph Descriptor (VNFFGD) and the Virtual Link Descriptor (VLD). It complements the work presented in the other work package 1 deliverables -in special the Reference Architecture in D1.1 [3] and the Slice Intent Metamodel in D1.4 [4]- and determine the base data structures for the implementation work that will be performed in work packages 2, 3 and 4.

2 Introduction

MATILDA Architecture [3] highlights two distinctive administrative zones involved in the creation of 5G-ready applications. The first administrative zone regards the "Application Orchestrator", that usually resides with the vertical service provider and composes the top layer of the framework. This layer allows developers to easily create software components in a familiar way, by abstract some infrastructure implementation aspects, but still providing means of enforcing the proper provisioning of resources.

The second administrative zone regards the infrastructure provider, a role usually played by telecommunication companies, and composes the lower layer of the framework. This layer is responsible for slicing the available infrastructure and allocating the required network and computational resources for running the applications of each tenant in an isolated way.

The integration between those two layers is the responsibility of MATILDA's Slice Broker, an entity capable of translating **slice intents** associated with the original **application graph** (as specified by the metamodels in D1.4 [5] and D1.2 [4]) into valid allocation requests to the Infrastructure Provider. In terms of network resources, slice intents are matched against a set of network services made available by the telecommunication company.

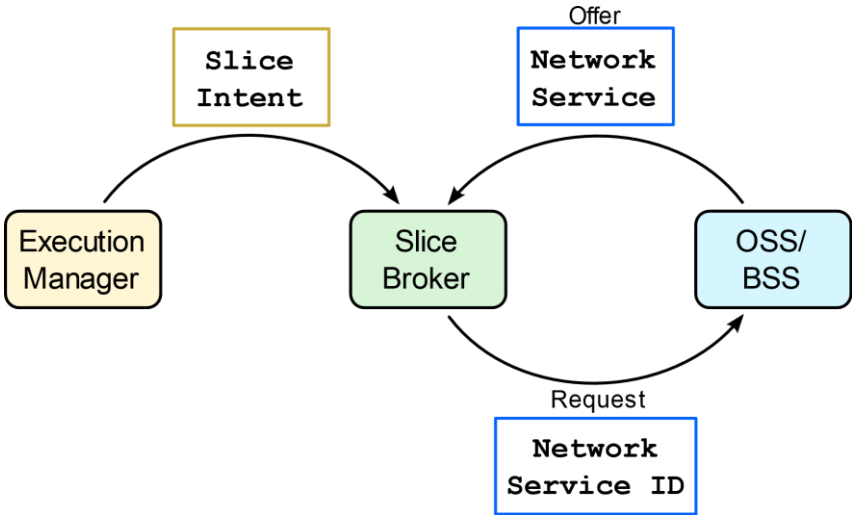

Figure 1: Slice Intent and Network Service matching by the Slice Broker

This process is illustrated in fig. 1 and requires the information about the network services to be organized according to **models** so the correct control details can flow between the Slice Broker and the telecommunication company operations and business support systems (OSS/BSS). Moreover, to have consistency between those entities, it is necessary to define a set of rules about how to write those modes. For each model the set of rules is also known as **metamodel**

Additionally, MATILDA relies on the same models and metamodels to enable code sharing and collaboration within the community. When published to MATILDA Marketplace, network service models (and all the related constituents sub-models) can be adopted by the telecommunication companies, improving the availability of products and new features for their infrastructure.

In order to lower the barriers of adoption and contribute to 5G environment, the concept of Network Service (NS) in MATILDA, and therefore the related metamodels, derives directly from ETSI specification [1]. More precisely, MATILDA relies as much as possible in the Open Source MANO project [6] augmented descriptors -since those are the *de facto* standards, not only being supported and promoted by ETSI but also counting with a series of practical demonstrations that provide industrial-level implementation and validation- adding extra fields related to specific business logic.

3 Network-related Metamodels Overview

The following sections present a high-level description of these metamodels, discuss about the underlying data structures and details their implementation.

3.1 Network Service Descriptor

In MATILDA, the top-level metamodel regarding infrastructure provided by telecommunication companies is the Network Service Descriptor (NSD). This metamodel abstracts chains of network functions (i.e. functional blocks that compose the network and have well-defined external interfaces, usually performing some kind of operation or transformation with the data flow) while providing important information for the infrastructure orchestrator.

The NSD specify a template from which instances of the same network service can be created but changing the input parameters. It contains the service description, including SLAs, deployment flavours, references to the virtual links, the constituent network functions, and to the forwarding graph that describes the data flow between them. Monitoring parameters and placement and scaling options can also be specified to instruct the infrastructure orchestrator.

Within the infrastructure provider administrative zone of MATILDA, network services are used to provide interconnectivity between application components or user equipment in physically separated locations, realizing a network slice.

3.2 Virtual Network Function Descriptor

The Virtual Network Function Descriptor (VNFD) specify not only how a single network function can be deployed in a virtualised environment, but also how it should be managed overtime. Moreover, this descriptor enumerates all the external interfaces exposed by the VNF, so service chains can be created.

It contains the VNF description and -since VNFs can be composed by separated software units- includes its internal decomposition in Virtual Network Function Components (VNFCs -also known as virtual deployment units, VDUs- usually virtual machine or container images), deployment flavours and references to the virtual links (internal or external).

Additionally, it also contain a series of configuration scripts that are required for life-cycle management and their arbitrary input parameters. Computation resources are listed and similarly to the NSD, monitoring, placement and scaling options can be specified.

3.3 Physical Network Function Descriptor

The Physical Network Function Descriptor (PNFD) specify how a physical equipment that implements a network function can be connected to a chain of other network functions

(virtualised or not). This metamodel is very similar to the VNFD and have similar purposes, but it is used to describe physical network functions and not virtual, so properties required for supporting virtualisation are suppressed.

The PNFD focus on the external interfaces required by the physical equipment and does not support placement and scaling options.

3.4 Virtual Link Descriptor

The Virtual Link Descriptor (VLD) contains the description of the virtual network links that compose the service, abstracting the interconnectivity between two different network functions (or VDUs). It also contains requirements about the network and stores information about the endpoints and type of connection.

3.5 VNF Forwarding Graph Descriptor

The VNF Forwarding Graph Descriptor (VNFFGD) describes the topology of the network service, by referencing network functions and virtual links. It contains the NS constituent VNFs, as well as their deployment in terms of network connectivity information.

The VNFFGD allows the specification of classification rules, so the data traffic can be steered through the rendered service path (RSP).

4 Data Structure and Modelling Language

For a proper organization of the information, MATILDA -in compliance with ETSI [1]- makes use of a tree data structure, which follows the standard XML approach[1] [7] for NSD and associated metamodels. In this data structure, each node has a name that identifies it, and can be seen as either a **leaf** node (when a value is provided) or a **container** node (when it contains one or more **children nodes**). Usually, leaf nodes are seen (and referred to as) properties of their parent nodes and must follow a specific computational **type**.

Since, a XML-inspired tree data structure can be arbitrarily flexible, YANG modules [8] are used to shape the format of this tree. These modules are the main subject of this deliverable and are available in the source code repository. Modules, in the context of YANG, are the primary organization unit that define the nodes that should be attached to the root node of the tree data structure. Inside each module, nodes can define its own children and properties.

Interestingly, YANG also enables the usage of **list** and **leaflist** nodes, using XML native features. These special nodes happen when containers and leaves (respectively) share the same name. YANG also allows referencing using XPath [9] expressions.

[1] Despite of following the XML approach, the models do not have to be necessarily serialized using the XML language.

5 Metamodels

The following subsections describe in detail the aforementioned tree data structure, as organized in the YANG modules. In most of the cases, the description provided is directly derived from the OSM Information Model [2] [10], as produced by the original authors, having been extracted via compilation. Modifications regarding MATILDA-specific business logic and decoupling from OSM implementation details were performed. The defined models are stored in the Gitlab repository [11]

5.1 Module: nsd

The following fields are defined:

Field	Type	Description
nsd-catalog		See: nsd:nsd-catalog

5.1.1 nsd:nsd-catalog

The following fields are defined:

Field	Type	Description
nsd	**list[1...N]**	See: nsd:nsd
schema-version	**string** default: **'v3.0'**	Schema version for the NSD. If unspecified, it assumes v3.0

nsd:nsd

The following fields are defined:

Field	Type	Description
category	**string[1...N]**	List of categories that can be used to group NS services providing similar features, e.g. **firewalling**, **VPN**, **lawful-inspection**, **ids/ips**.
connection-point	**list[1...N]**	List for external connection points. Each NS has one or more external connection points. As the name implies that external connection points are used for connecting the NS to other NS or to external networks. Each NS exposes these connection points to the orchestrator. The orchestrator can construct network service chains by connecting the connection points between different NS. See: nsd-base:connection-point
constituent-vnfd	**list[1...N]**	List of VNFDs that are part of this network service. See: nsd:constituent-vnfd
description	**string**	Description of the NSD.
id	**string** length: **(1, 63)**	Identifier for the NSD.

initial-service-primitive	**list[1...N]**	Initial set of service primitives for NSD. See: nsd-base:initial-service-primitive
input-parameter-xpath	**list[1...N]**	List of xpaths to parameters inside the NSD the can be customized during the instantiation. See: manotypes:input-parameter-xpath
ip-profiles	**list[1...N]**	List of IP Profiles. IP Profile describes the IP characteristics for the Virtual-Link See: manotypes:ip-profiles
key-pair	**list[1...N]**	Used to configure the list of public keys to be injected as part of ns instantiation See: nsd-base:key-pair
license		Software license applied to the network service See: nsd-base:license
logical-type	**string**	Different network services can be implemented in different ways, but perform basically a similar job with equivalent logical operations and similar connectivity. This field should be used to identify a high-level type that abstract the chosen implementation, allowing similar network services to be identified.
logo	**string**	File path for the vendor specific logo. For example icons/mylogo.png. The logo should be part of the network service
monitoring-param	**list[1...N]**	See: nsd:monitoring-param
name	**string**	NSD name.
parameter-pool	**list[1...N]**	Pool of parameter values which must be pulled from during configuration See: nsd-base:parameter-pool
placement-groups	**list[1...N]**	List of placement groups at NS level See: nsd:placement-groups
scaling-group-descriptor	**list[1...N]**	scaling group descriptor within this network service. The scaling group defines a group of VNFs, and the ratio of VNFs in the network service that is used as target for scaling action See: nsd-base:scaling-group-descriptor
service-primitive	**list[1...N]**	Network service level service primitives. See: nsd:service-primitive
short-name	**string**	Short name to appear as label in the UI
terminate-service-primitive	**list[1...N]**	Set of service primitives during termination for NSD. See: nsd-base:terminate-service-primitive
user	**list[1...N]**	List of users to be added through cloud-config See: nsd-base:user
vendor	**string**	Vendor of the NSD.

version	**string**	Version of the NSD
vld	**list[1...N]**	See: nsd:vld
vnf-dependency	**list[1...N]**	List of VNF dependencies. See: nsd:vnf-dependency
vnffgd	**list[1...N]**	List of VNF Forwarding Graph Descriptors (VNFFGD). See: nsd-base:vnffgd

nsd-base:connection-point

List for external connection points. Each NS has one or more external connection points. As the name implies that external connection points are used for connecting the NS to other NS or to external networks. Each NS exposes these connection points to the orchestrator. The orchestrator can construct network service chains by connecting the connection points between different NS.

The following fields are defined:

Field	Type	Description
name	**string**	Name of the NS connection point.
type	**connection-point-type**	Type of the connection point. • VPORT: Virtual Port • VNIC_ADDR: Virtual NIC Address • PNIC_ADDR: Physical NIC Address • PPORT: Physical Port.

nsd:constituent-vnfd

List of VNFDs that are part of this network service.

The following fields are defined:

Field	Type	Description
member-vnf-index	**uint64**	Identifier/index for the VNFD. This separate id is required to ensure that multiple VNFs can be part of single NS
start-by-default	**boolean** default: **'true'**	VNFD is started as part of the NS instantiation
vnfd-id-ref	**leafref**	Identifier for the VNFD. **'/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id'**

nsd-base:initial-service-primitive

Initial set of service primitives for NSD.

The following fields are defined:

Field	Type	Description
name	**string**	Name of the configuration primitive.
parameter	**list[1...N]**	See: manotypes:parameter

seq	**uint64**	Sequence number for the configuration primitive.
user-defined-script	**string**	A user defined script.

manotypes:parameter

The following fields are defined:

Field	Type	Description
name	**string**	Name of the configuration parameter.
value	**string**	Value of the configuration primitive.

manotypes:input-parameter-xpath

List of xpaths to parameters inside the NSD the can be customized during the instantiation.

The following fields are defined:

Field	Type	Description
default-value	**string**	Default Value for the Input Parameter
label	**string**	A descriptive string
xpath	**string**	An xpath that specifies the element in a descriptor.

manotypes:ip-profiles

List of IP Profiles. IP Profile describes the IP characteristics for the Virtual-Link

The following fields are defined:

Field	Type	Description
description	**string**	Description for IP profile
ip-profile-params		See: manotypes:ip-profile-params
name	**string**	Name of the IP-Profile

manotypes:ip-profile-params

The following fields are defined:

Field	Type	Description
dhcp-params		See: manotypes:dhcp-params
dns-server	**list[1...N]**	See: manotypes:dns-server
gateway-address	**ip-address**	IP Address of the default gateway associated with IP Profile
ip-version	**ip-version** default: **'ipv4'**	Possible values: **unknown**, **ipv4** and **ipv6**
security-group	**string**	Name of the security group
subnet-address	**ip-prefix**	Subnet IP prefix associated with IP Profile
subnet-prefix-pool	**string**	VIM Specific reference to pre-created subnet prefix

manotypes:dhcp-params

The following fields are defined:

Field	Type	Description
count	**uint32**	Size of the DHCP pool associated with DHCP domain
enabled	**boolean** default: **'true'**	This flag indicates if DHCP is enabled or not
start-address	**ip-address**	Start IP address of the IP-Address range associated with DHCP domain

manotypes:dns-server

The following fields are defined:

Field	Type	Description
address	**ip-address**	List of DNS Servers associated with IP Profile

nsd-base:key-pair

Used to configure the list of public keys to be injected as part of ns instantiation

The following fields are defined:

Field	Type	Description
key	**string**	Key associated with this key pair
name	**string**	Name of this key pair

nsd-base:license

Software license applied to the network service

The following fields are defined:

Field	Type	Description
notice	**string**	Text indicating the license under which the software used by the NS is released. Usually contains a short note and a reference (e.g. URL) to the complete document. For example, when using the open source license MPLv2, the author may consider including: This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at https://mozilla.org/MPL/2.0/.
short-id	**string**	Short identifier related to the license type, as listed in the tl;dr legal website. For example: • **agpl3** • **apache2**

	• **artistic-2.0** • **bsd3** • **cc0-1.0** • **custom** • **epl** • **freebsd** • **gpl-2.0** • **gpl-3.0** • **isc** • **lgpl-2.1** • **lgpl-3.0** • **mit** • **mpl-2.0** • **opensource** • **proprietary** • **public-domain** O.B.S.: **proprietary**, **opensource** and **custom** values allow a very flexible usage for this field, but should be complemented with an explicative text **notice**.

nsd:monitoring-param

The following fields are defined:

Field	Type	Description
aggregation-type	**aggregation-type**	Possible values: • AVERAGE • MINIMUM • MAXIMUM • COUNT • SUM
description	**string**	
group-tag	**string**	A tag to group monitoring parameters
id	**string**	
name	**string**	
numeric-constraints		See: manotypes:numeric-constraints
text-constraints		See: manotypes:text-constraints
units	**string**	Measured Counter Units (e.g., Packets, Kbps, Mbps, etc.)
value-decimal	**decimal64**	Current value for a decimal parameter
value-integer	**int64**	Current value for an integer parameter

value-string	**string**	Current value for a string parameter
value-type	**param-value-type** default: **'INT'**	The type of the parameter value: • INT • DECIMAL • STRING
vnfd-monitoring-param	**list[1...N]**	A list of VNFD monitoring params See: nsd:vnfd-monitoring-param
widget-type	**widget-type** default: **'COUNTER'**	Defines the UI Display variant of measured counters.

manotypes:numeric-constraints

The following fields are defined:

Field	Type	Description
max-value	**uint64**	Maximum value for the parameter
min-value	**uint64**	Minimum value for the parameter

manotypes:text-constraints

The following fields are defined:

Field	Type	Description
max-length	**uint8**	Maximum string length for the parameter
min-length	**uint8**	Minimum string length for the parameter

nsd:vnfd-monitoring-param

A list of VNFD monitoring params

The following fields are defined:

Field	Type	Description
member-vnf-index-ref	**leafref**	Mandatory reference to member-vnf within constituent-vnfds **'../../../constituent-vnfd/member-vnf-index'**
vnfd-id-ref	**leafref**	A reference to a VNFD. This is a leafref **'../../../constituent-vnfd[member-vnf-index = current()/../member-vnf-index-ref]/vnfd-id-ref'**
vnfd-monitoring-param-ref	**leafref**	A reference to the VNFD monitoring param **'/vnfd:vnfd-catalog/vnfd:vnfd[vnfd:id = current()/../vnfd-id-ref]/vnfd:monitoring-param/vnfd:id'**

nsd-base:parameter-pool

Pool of parameter values which must be pulled from during configuration

The following fields are defined:

Field	Type	Description

name	**string**	Name of the configuration value pool
range		Create a range of values to populate the pool with See: nsd-base:range

nsd-base:range

Create a range of values to populate the pool with

The following fields are defined:

Field	Type	Description
end-value	**uint32**	Generated pool values stop at this value
start-value	**uint32**	Generated pool values start at this value

nsd:placement-groups

List of placement groups at NS level

The following fields are defined:

Field	Type	Description
member-vnfd	**list[1...N]**	List of VNFDs that are part of this placement group See: nsd:member-vnfd
name	**string**	Place group construct to define the compute resource placement strategy in cloud environment
requirement	**string**	This is free text space used to describe the intent/rationale behind this placement group. This is for human consumption only
strategy	**enumeration** default: **'COLOCATION'**	Strategy associated with this placement group Following values are possible: • **COLOCATION**: Colocation strategy imply intent to share the physical infrastructure (hypervisor/network) among all members of this group. • **ISOLATION**: Isolation strategy imply intent to not share the physical infrastructure (hypervisor/network) among the members of this group.

nsd:member-vnfd

List of VNFDs that are part of this placement group

The following fields are defined:

Field	Type	Description
member-vnf-index-ref	**leafref**	Member VNF index of this member VNF **'../../../constituent-vnfd/member-vnf-index'**
vnfd-id-ref	**leafref**	Identifier for the VNFD. **'../../../constituent-vnfd[member-vnf-index = current()/../member-vnf-index-ref]/vnfd-id-ref'**

nsd-base:scaling-group-descriptor

scaling group descriptor within this network service. The scaling group defines a group of VNFs, and the ratio of VNFs in the network service that is used as target for scaling action

The following fields are defined:

Field	Type	Description
max-instance-count	**uint32** default: **'10'**	Maximum instances of this scaling group that are allowed in a single network service. The network service scaling will fail, when the number of service group instances exceed the max-instance-count specified.
min-instance-count	**uint32** default: **'0'**	Minimum instances of the scaling group which are allowed. These instances are created by default when the network service is instantiated.
name	**string**	Name of this scaling group.
scaling-config-action	**list[1...N]**	List of scaling config actions See: nsd-base:scaling-config-action
scaling-policy	**list[1...N]**	See: nsd-base:scaling-policy
vnfd-member	**list[1...N]**	List of VNFs in this scaling group See: nsd-base:vnfd-member

nsd-base:scaling-config-action

List of scaling config actions

The following fields are defined:

Field	Type	Description
ns-service-primitive-name-ref	**leafref**	Reference to the NS service primitive **'../../../service-primitive/name'**
trigger	**scaling-trigger**	scaling trigger

nsd-base:scaling-policy

The following fields are defined:

Field	Type	Description
cooldown-time	**uint32**	The duration after a scaling-in/scaling-out action has been triggered, for which there will be no further optional
enabled	**boolean** default: **'true'**	Specifies if the scaling policy can be applied
name	**string**	Name of the scaling policy

scale-in-operation-type	**scaling-criteria-operation** default: **'AND'**	Operation to be applied to check between scaling criteria to check if the scale in threshold condition has been met. Defaults to AND
scale-out-operation-type	**scaling-criteria-operation** default: **'OR'**	Operation to be applied to check between scaling criteria to check if the scale out threshold condition has been met. Defauls to OR
scaling-criteria	**list[1...N]**	list of conditions to be met for generating scaling requests See: nsd-base:scaling-criteria
scaling-type	**scaling-policy-type**	Type of scaling
threshold-time	**uint32**	The duration for which the criteria must hold true

nsd-base:scaling-criteria

list of conditions to be met for generating scaling requests

The following fields are defined:

Field	Type	Description
name	**string**	
ns-monitoring-param-ref	**leafref**	Reference to the NS level monitoring parameter that is aggregated **'../../../../monitoring-param/id'**
scale-in-threshold	**uint64**	Value below which scale-in requests are generated
scale-out-threshold	**uint64**	Value above which scale-out requests are generated

nsd-base:vnfd-member

List of VNFs in this scaling group

The following fields are defined:

Field	Type	Description
count	**uint32** default: **'1'**	count of this member VNF within this scaling group. The count allows to define the number of instances when a scaling action targets this scaling group
member-vnf-index-ref	**leafref**	member VNF index of this member VNF **'../../../constituent-vnfd/member-vnf-index'**

nsd:service-primitive

Network service level service primitives.

The following fields are defined:

Field	Type	Description
name	**string**	Name of the service primitive.
parameter	**list[1...N]**	List of parameters for the service primitive.

		See: nsd:parameter
parameter-group	**list[1...N]**	Grouping of parameters which are logically grouped in UI See: manotypes:parameter-group
user-defined-script	**string**	A user defined script.
vnf-primitive-group	**list[1...N]**	List of service primitives grouped by VNF. See: nsd:vnf-primitive-group

nsd:parameter

List of parameters for the service primitive.

The following fields are defined:

Field	Type	Description
data-type	**parameter-data-type**	Data type associated with the name.
default-value	**string**	The default value for this field
hidden	**boolean** default: **'false'**	The value should be hidden by the UI. Only applies to parameters with default values.
mandatory	**boolean** default: **'false'**	Is this field mandatory
name	**string**	Name of the parameter.
parameter-pool	**string**	NSD parameter pool name to use for this parameter
read-only	**boolean** default: **'false'**	The value should be dimmed by the UI. Only applies to parameters with default values.

manotypes:parameter-group

Grouping of parameters which are logically grouped in UI

The following fields are defined:

Field	Type	Description
mandatory	**boolean** default: **'true'**	Is this parameter group mandatory
name	**string**	Name of the parameter group
parameter	**list[1...N]**	List of parameters for the service primitive. See: manotypes:parameter

nsd:vnf-primitive-group

List of service primitives grouped by VNF.

The following fields are defined:

Field	Type	Description

member-vnf-index-ref	**leafref**	Reference to member-vnf within constituent-vnfds **'../../../constituent-vnfd/member-vnf-index'**
primitive	**list[1...N]**	See: nsd:primitive
vnfd-id-ref	**leafref**	A reference to a VNFD. This is a leafref **'../../../constituent-vnfd[member-vnf-index = current()/../member-vnf-index-ref]/vnfd-id-ref'**
vnfd-name	**leafref**	Name of the VNFD **'/vnfd:vnfd-catalog/vnfd:vnfd[vnfd:id = current()/../vnfd-id-ref]/vnfd:name'**

nsd:primitive

The following fields are defined:

Field	Type	Description
index	**uint32**	Index of this primitive
name	**string**	Name of the primitive in the VNF primitive

nsd-base:terminate-service-primitive

Set of service primitives during termination for NSD.

The following fields are defined:

Field	Type	Description
name	**string**	Name of the configuration primitive.
parameter	**list[1...N]**	See: manotypes:parameter
seq	**uint64**	Sequence number for the configuration primitive.
user-defined-script	**string**	A user defined script.

nsd-base:user

List of users to be added through cloud-config

The following fields are defined:

Field	Type	Description
key-pair	**list[1...N]**	Used to configure the list of public keys to be injected as part of ns instantiation See: nsd-base:key-pair
name	**string**	Name of the user
user-info	**string**	The user name's real name

nsd:vld

The following fields are defined:

Field	Type	Description
description	**string**	Description of the VLD.

id	**string**	Identifier for the VLD.
leaf-bandwidth	**uint64**	For ELAN this is the bandwidth of branches.
mgmt-network	**boolean** default: **'false'**	Flag indicating whether this network is a VIM management network
name	**string**	Virtual Link Descriptor (VLD) name.
provider-network		Container for the provider network. See: manotypes:provider-network
root-bandwidth	**uint64**	For ELAN this is the aggregate bandwidth.
short-name	**string**	Short name to appear as label in the UI
type	**virtual-link-type**	Type of virtual link. Possible values: • ELAN: A multipoint service connecting a set of VNFs • ELINE: For a simple point to point connection between a VNF and the existing network. • ETREE: A multipoint service connecting one or more roots and a set of leaves, but preventing inter-leaf communication.
vendor	**string**	Provider of the VLD.
version	**string**	Version of the VLD
vnfd-connection-point-ref	**list[1...N]**	A list of references to connection points. See: nsd:vnfd-connection-point-ref

Deviation: `init-params`

Extra parameters for VLD instantiation

Depending on the value of **`init-params`**, **`vld`** may have additional fields.

When `init-params` **is** `'vim-network-profile'`

The following fields are defined:

Field	Type	Description
ip-profile-ref	**leafref**	Named reference to IP-profile object **'../../ip-profiles/name'**

When `init-params` **is** `'vim-network-ref'`

The following fields are defined:

Field	Type	Description
vim-network-name	**string**	Name of network in VIM account. This is used to indicate pre-provisioned network name in cloud account.

manotypes:provider-network

Container for the provider network.

The following fields are defined:

Field	Type	Description
physical-network	**string**	Name of the physical network on which the provider network is built.
segmentation_id	**uint32**	ID of segregated virtual networks

nsd:vnfd-connection-point-ref

A list of references to connection points.

The following fields are defined:

Field	Type	Description
ip-address	**ip-address**	IP address of the connection point
member-vnf-index-ref	**leafref**	Reference to member-vnf within constituent-vnfds **'../../../constituent-vnfd/member-vnf-index'**
vnfd-connection-point-ref	**leafref**	A reference to a connection point name **'/vnfd:vnfd-catalog/vnfd:vnfd[vnfd:id = current()/../vnfd-id-ref]/vnfd:connection-point/vnfd:name'**
vnfd-id-ref	**leafref**	A reference to a VNFD **'../../../constituent-vnfd[member-vnf-index = current()/../member-vnf-index-ref]/vnfd-id-ref'**

nsd:vnf-dependency

List of VNF dependencies.

The following fields are defined:

Field	Type	Description
vnf-depends-on-ref	**leafref**	Reference to VNF that sorce VNF depends. **'/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id'**
vnf-source-ref	**leafref**	**'/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id'**

nsd-base:vnffgd

List of VNF Forwarding Graph Descriptors (VNFFGD).

The following fields are defined:

Field	Type	Description
classifier	**list[1...N]**	List of classifier rules. See: nsd-base:classifier
description	**string**	Description of the VNFFGD.
id	**string**	Identifier for the VNFFGD.
name	**string**	VNFFGD name.
rsp	**list[1...N]**	List of Rendered Service Paths (RSP). See: nsd-base:rsp

short-name	**string**	Short name to appear as label in the UI
vendor	**string**	Provider of the VNFFGD.
version	**string**	Version of the VNFFGD

nsd-base:classifier

List of classifier rules.

The following fields are defined:

Field	Type	Description
id	**string**	Identifier for the classifier rule.
match-attributes	**list[1...N]**	List of match attributes. See: nsd-base:match-attributes
member-vnf-index-ref	**leafref**	Reference to member-vnf within constituent-vnfds **'../../../constituent-vnfd/member-vnf-index'**
name	**string**	Name of the classifier.
rsp-id-ref	**leafref**	A reference to the RSP. **'../../rsp/id'**
vnfd-connection-point-ref	**string**	A reference to a connection point name in a vnfd.
vnfd-id-ref	**leafref**	A reference to a vnfd. **'../../../constituent-vnfd[member-vnf-index = current()/../member-vnf-index-ref]/vnfd-id-ref'**

nsd-base:match-attributes

List of match attributes.

The following fields are defined:

Field	Type	Description
destination-ip-address	**ip-address**	Destination IP address.
destination-port	**port-number**	Destination port number.
id	**string**	Identifier for the classifier match attribute rule.
ip-proto	**uint8**	IP Protocol.
source-ip-address	**ip-address**	Source IP address.
source-port	**port-number**	Source port number.

nsd-base:rsp

List of Rendered Service Paths (RSP).

The following fields are defined:

Field	Type	Description
id	**string**	Identifier for the RSP.
name	**string**	RSP name.

vnfd-connection-point-ref	**list[1...N]**	A list of references to connection points. See: nsd-base:vnfd-connection-point-ref

nsd-base:vnfd-connection-point-ref

A list of references to connection points.

The following fields are defined:

Field	Type	Description
member-vnf-index-ref	**leafref**	Reference to member-vnf within constituent-vnfds **'../../../../constituent-vnfd/member-vnf-index'**
order	**uint8**	A number that denotes the order of a VNF in a chain
vnfd-connection-point-ref	**string**	A reference to a connection point name in a vnfd.
vnfd-id-ref	**leafref**	A reference to a vnfd. **'../../../../constituent-vnfd[member-vnf-index = current()/../member-vnf-index-ref]/vnfd-id-ref'**

5.2 Module: vnfd

The following fields are defined:

Field	Type	Description
vnfd-catalog		Virtual Network Function Descriptor (VNFD). See: vnfd:vnfd-catalog

5.2.1 vnfd:vnfd-catalog

Virtual Network Function Descriptor (VNFD).

The following fields are defined:

Field	Type	Description
schema-version	**string** default: **'v3.0'**	Schema version for the VNFD. If unspecified, it assumes v3.0
vnfd	**list[1...N]**	See: vnfd:vnfd

vnfd:vnfd

The following fields are defined:

Field	Type	Description
category	**string[1...N]**	List of categories that can be used to group VNFs providing similar features, e.g. **security**, **cache**, **load-balancing**, ...
connection-point	**list[1...N]**	List for external connection points. Each VNF has one or more external connection points that connect the VNF to other VNFs or to external networks. Each VNF exposes connection points to the orchestrator, which can construct network services by connecting the

		connection points between different VNFs. The NFVO will use VLDs and VNFFGs at the network service level to construct network services. See: vnfd-base:connection-point
description	**string**	Description of the VNFD.
http-endpoint	**list[1...N]**	List of http endpoints to be used by monitoring params See: manotypes:http-endpoint
id	**string** length: **(1, 63)**	Identifier for the VNFD.
internal-vld	**list[1...N]**	List of Internal Virtual Link Descriptors (VLD). The internal VLD describes the basic topology of the connectivity such as E-LAN, E-Line, E-Tree. between internal VNF components of the system. See: vnfd-base:internal-vld
ip-profiles	**list[1...N]**	List of IP Profiles. IP Profile describes the IP characteristics for the Virtual-Link See: manotypes:ip-profiles
license		Software license applied to the VNF See: vnfd-base:license
logical-type	**string**	Different network functions can be implemented in different ways, but perform basically equivalent logical operations. This field should be used to identify a high-level type that abstract the chosen implementation, allowing similar VNFs to be identified. Examples: firewall, cdn, ids, sbc, cpe, reverse-proxy ...
logo	**string**	Vendor logo for the Virtual Network Function
mgmt-interface		Interface over which the VNF is managed. See: vnfd-base:mgmt-interface
monitoring-param	**list[1...N]**	List of monitoring parameters at the network service level See: manotypes:monitoring-param
name	**string**	VNFD name.
operational-status	**vnf-operational-status**	The operational status of the VNF • **init** : The VNF has just started. • **running** : The VNF is active in VM • **upgrading** : The VNF is being upgraded (EXPERIMENTAL) • **terminate** : The VNF is being terminated • **terminated** : The VNF is in the terminated state. • **failed** : The VNF instantiation failed.
placement-groups	**list[1...N]**	List of placement groups at VNF level See: vnfd-base:placement-groups

service-function-chain	**enumeration** default: **'UNAWARE'**	Type of node in Service Function Chaining Architecture
service-function-type	**string**	Type of Service Function. NOTE: This needs to map with Service Function Type in ODL to support VNFFG. Service Function Type is mandatory param in ODL SFC.
short-name	**string**	Short name to appear as label in the UI
vdu	**list[1...N]**	List of Virtual Deployment Units See: vnfd-base:vdu
vdu-dependency	**list[1...N]**	List of VDU dependencies. See: vnfd-base:vdu-dependency
vendor	**string**	Vendor of the VNFD.
version	**string**	Version of the VNFD
vnf-configuration		See: vnfd-base:vnf-configuration

vnfd-base:connection-point

List for external connection points. Each VNF has one or more external connection points that connect the VNF to other VNFs or to external networks. Each VNF exposes connection points to the orchestrator, which can construct network services by connecting the connection points between different VNFs. The NFVO will use VLDs and VNFFGs at the network service level to construct network services.

The following fields are defined:

Field	Type	Description
id	**string**	Identifier for the internal connection points
name	**string**	Name of the connection point
port-security-enabled	**boolean**	Enables the port security for the port
short-name	**string**	Short name to appear as label in the UI
type	**connection-point-type**	Type of the connection point.

manotypes:http-endpoint

List of http endpoints to be used by monitoring params

The following fields are defined:

Field	Type	Description
headers	**list[1...N]**	Custom HTTP headers to put on HTTP request See: manotypes:headers
https	**boolean**	Pick HTTPS instead of HTTP , Default is false

	default: `'false'`	
method	**http-method** default: **'GET'**	Method that the URI should perform. Default action is GET.
password	**string**	The HTTP basic auth password
path	**string**	The HTTP path on the management server
polling-interval-secs	**uint8** default: **'2'**	The HTTP polling interval in seconds
port	**port-number**	The HTTP port to connect to
username	**string**	The HTTP basic auth username

manotypes:headers

Custom HTTP headers to put on HTTP request

The following fields are defined:

Field	Type	Description
key	**string**	HTTP header key
value	**string**	HTTP header value

vnfd-base:internal-vld

List of Internal Virtual Link Descriptors (VLD). The internal VLD describes the basic topology of the connectivity such as E-LAN, E-Line, E-Tree. between internal VNF components of the system.

The following fields are defined:

Field	Type	Description
description	**string**	Text describing the VLD
id	**string**	Identifier for the VLD
internal-connection-point	**list[1...N]**	List of internal connection points in this VLD See: vnfd-base:internal-connection-point
leaf-bandwidth	**uint64**	For ELAN this is the bandwidth of branches.
name	**string**	Name of the internal VLD
provider-network		Container for the provider network. See: manotypes:provider-network
root-bandwidth	**uint64**	For ELAN this is the aggregate bandwidth.
short-name	**string**	Short name to appear as label in the UI
type	**virtual-link-type**	Type of virtual link. Possible values: • ELAN: A multipoint service connecting a set of VNFs • ELINE: For a simple point to point connection between a VNF and the existing network.

	• ETREE: A multipoint service connecting one or more roots and a set of leaves, but preventing inter-leaf communication.

Deviation: `init-params`

Extra parameters for VLD instantiation

Depending on the value of **`init-params`**, **`internal-vld`** may have additional fields.

When `init-params` **is** `'vim-network-profile'`

The following fields are defined:

Field	Type	Description
`ip-profile-ref`	**`string`**	Named reference to IP-profile object

When `init-params` **is** `'vim-network-ref'`

The following fields are defined:

Field	Type	Description
`vim-network-name`	**`string`**	Name of network in VIM account. This is used to indicate pre-provisioned network name in cloud account.

vnfd-base:internal-connection-point

List of internal connection points in this VLD

The following fields are defined:

Field	Type	Description
`id-ref`	**`leafref`**	reference to the internal connection point id **`'../../../vdu/internal-connection-point/id'`**
`ip-address`	**`ip-address`**	IP address of the internal connection point

vnfd-base:license

Software license applied to the VNF

The following fields are defined:

Field	Type	Description
`notice`	**`string`**	Text indicating the license under which the software used by the VNF is released. Usually contains a short note and a reference (e.g. URL) to the complete document. For example, when using the open source license MPLv2, the author may consider including: This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at https://mozilla.org/MPL/2.0/.

short-id	**string**	Short identifier related to the license type, as listed in the tl;dr legal website. For example: • **agpl3** • **apache2** • **artistic-2.0** • **bsd3** • **cc0-1.0** • **custom** • **epl** • **freebsd** • **gpl-2.0** • **gpl-3.0** • **isc** • **lgpl-2.1** • **lgpl-3.0** • **mit** • **mpl-2.0** • **opensource** • **proprietary** • **public-domain** O.B.S.: **proprietary**, **opensource** and **custom** values allow a very flexible usage for this field, but should be complemented with an explicative text **notice**.

vnfd-base:mgmt-interface

Interface over which the VNF is managed.

The following fields are defined:

Field	Type	Description
dashboard-params		Parameters for the VNF dashboard See: vnfd-base:dashboard-params
port	**port-number**	Port for the management interface.

vnfd-base:dashboard-params

Parameters for the VNF dashboard

The following fields are defined:

Field	Type	Description
https	**boolean**	Pick HTTPS instead of HTTP , Default is false
path	**string**	The HTTP path for the dashboard
port	**port-number**	The HTTP port for the dashboard

Deviation: `endpoint-type`

Indicates the type of management endpoint.

Depending on the value of **endpoint-type**, **mgmt-interface** may have additional fields.

When endpoint-type **is** 'cp'

Use the ip address associated with this connection point. This cp is then considered as management.

The following fields are defined:

Field	Type	Description
cp	**leafref**	**'../../connection-point/name'**

When endpoint-type **is** 'ip'

Specifies the static IP address for managing the VNF.

The following fields are defined:

Field	Type	Description
ip-address	**ip-address**	The ip-address type represents an IP address and is IP version neutral. The format of the textual representations implies the IP version.

When endpoint-type **is** 'vdu-id'

Use the default management interface on this VDU.

The following fields are defined:

Field	Type	Description
vdu-id	**leafref**	**'../../vdu/id'**

manotypes:monitoring-param

List of monitoring parameters at the network service level

The following fields are defined:

Field	Type	Description
description	**string**	
group-tag	**string**	A tag to group monitoring parameters
http-endpoint-ref	**leafref**	**'../../http-endpoint/path'**
id	**string**	
json-query-method	**json-query-method** default: **'NAMEKEY'**	The method to extract a value from a JSON response • NAMEKEY - Use the name as the key for a non-nested value. • JSONPATH - Use jsonpath-rw implementation to extract a value. • OBJECTPATH - Use objectpath implementation to extract a value.

json-query-params		See: manotypes:json-query-params
name	**string**	Name identifying the monitoring parameter
numeric-constraints		See: manotypes:numeric-constraints
text-constraints		See: manotypes:text-constraints
units	**string**	Measured Counter Units (e.g., Packets, Kbps, Mbps, etc.)
value-decimal	**decimal64**	Current value for a decimal parameter
value-integer	**int64**	Current value for an integer parameter
value-string	**string**	Current value for a string parameter
value-type	**param-value-type** default: **'INT'**	The type of the parameter value: • INT • DECIMAL • STRING
widget-type	**widget-type** default: **'COUNTER'**	Defines the UI Display variant of measured counters.

manotypes:json-query-params

The following fields are defined:

Field	Type	Description
json-path	**string**	The jsonpath to use to extract value from JSON structure
object-path	**string**	The objectpath to use to extract value from JSON structure

vnfd-base:placement-groups

List of placement groups at VNF level

The following fields are defined:

Field	Type	Description
member-vdus	**list[1...N]**	List of VDUs that are part of this placement group See: vnfd-base:member-vdus
name	**string**	Place group construct to define the compute resource placement strategy in cloud environment
requirement	**string**	This is free text space used to describe the intent/rationale behind this placement group. This is for human consumption only
strategy	**enumeration** default: **'COLOCATION'**	Strategy associated with this placement group Following values are possible: • **COLOCATION**: Colocation strategy imply intent to share the physical infrastructure (hypervisor/network) among all members of this group.

	• **ISOLATION**: Isolation strategy imply intent to not share the physical infrastructure (hypervisor/network) among the members of this group.

vnfd-base:member-vdus

List of VDUs that are part of this placement group

The following fields are defined:

Field	Type	Description
member-vdu-ref	**leafref**	**'../../../vdu/id'**

vnfd-base:vdu

List of Virtual Deployment Units

The following fields are defined:

Field	Type	Description
alarm	**list[1...N]**	See: vnfd-base:alarm
alternative-images	**list[1...N]**	List of alternative images per VIM type. Different images can be used for specific types of VIMs instead of the default image. This allows deployments in sites where the image identifier in the VIM is given by the VIM provider and cannot be modified. If an alternative image is specified for a VIM type, it will prevail over the default image See: vnfd-base:alternative-images
count	**uint64**	Number of instances of VDU
description	**string**	Description of the VDU.
guest-epa		See: manotypes:guest-epa
host-epa		Specifies the host level EPA attributes. See: manotypes:host-epa
hypervisor-epa		See: manotypes:hypervisor-epa
id	**string**	Unique id for the VDU
image	**string**	Image name for the software image. If the image name is found within the VNF package it will be uploaded to all VIM accounts during onboarding process. Otherwise, the image must be added to the VIM account with the same name as entered here.
image-checksum	**string**	Image md5sum for the software image. The md5sum, if provided, along with the image name uniquely identifies an image uploaded to the CAL.
interface	**list[1...N]**	List of Interfaces (external and internal) for the VNF See: vnfd-base:interface

internal-connection-point	**list[1...N]**	List for internal connection points. Each VNFC has zero or more internal connection points. Internal connection points are used for connecting the VNF with components internal to the VNF. If a VNF has only one VNFC, it may not have any internal connection points. See: vnfd-base:internal-connection-point
mgmt-vpci	**string**	Specifies the virtual PCI address. Expressed in the following format **dddd:dd:dd.d**. For example: 0000:00:12.0. This information can be used to pass as metadata during the VM creation.
name	**string**	Unique name for the VDU
supplemental-boot-data		See: manotypes:supplemental-boot-data
vdu-configuration		See: vnfd-base:vdu-configuration
vm-flavor		See: manotypes:vm-flavor
volumes	**list[1...N]**	See: vnfd-base:volumes
vswitch-epa		See: manotypes:vswitch-epa

vnfd-base:alarm

The following fields are defined:

Field	Type	Description
actions		See: manotypes:actions
alarm-id	**string**	This field is reserved for the identifier assigned by the VIM provider
description	**string**	A description of this alarm
enabled	**boolean** default: **'true'**	This flag indicates whether the alarm has been enabled or disabled.
evaluations	**uint32**	Defines the length of time (seconds) in which metric data are collected in order to evaluate the chosen statistic.
metric	**alarm-metric-type**	The metric to be tracked by this alarm. Possible values: • CPU_UTILIZATION • MEMORY_UTILIZATION • STORAGE_UTILIZATION
name	**string**	A human readable string to identify the alarm
operation	**alarm-operation-type**	The relational operator used to define whether an alarm should be triggered in certain scenarios, such as if the metric statistic goes above or below a specified value. Possible values: • GE: greater than or equal

		• LE: less than or equal • GT: greater than • LT: less than • EQ: equal
period	**uint32**	The period defines the length of time (seconds) that the metric data are collected over in order to evaluate the chosen statistic.
repeat	**boolean** default: **'true'**	This flag indicates whether the alarm should be repeatedly emitted while the associated threshold has been crossed.
severity	**alarm-severity-type**	A measure of the importance or urgency of the alarm
statistic	**alarm-statistic-type**	The type of metric statistic that is tracked by this alarm. Possible values: • AVERAGE • MINIMUM • MAXIMUM • COUNT • SUM
value	**decimal64**	This value defines the threshold that, if crossed, will trigger the alarm.
vdur-id	**string**	The identifier of the VDUR that the alarm is associated with

manotypes:actions

The following fields are defined:

Field	Type	Description
alarm	**list[1...N]**	See: manotypes:alarm
insufficient-data	**list[1...N]**	See: manotypes:insufficient-data
ok	**list[1...N]**	See: manotypes:ok

manotypes:alarm

The following fields are defined:

Field	Type	Description
url	**string**	URL that should trigger the action

manotypes:insufficient-data

The following fields are defined:

Field	Type	Description
url	**string**	URL that should trigger the action

manotypes:ok

The following fields are defined:

Field	Type	Description
`url`	**`string`**	URL that should trigger the action

vnfd-base:alternative-images

List of alternative images per VIM type. Different images can be used for specific types of VIMs instead of the default image. This allows deployments in sites where the image identifier in the VIM is given by the VIM provider and cannot be modified. If an alternative image is specified for a VIM type, it will prevail over the default image

The following fields are defined:

Field	Type	Description
`image`	**`string`**	Image name for the software image. If the image name is found within the VNF package it will be uploaded to all VIM accounts during onboarding process. Otherwise, the image must be added to the VIM account with the same name as entered here.
`image-checksum`	**`string`**	Image md5sum for the software image. The md5sum, if provided, along with the image name uniquely identifies an image uploaded to the CAL.
`vim-type`	**`string`**	VIM type: openvim, openstack, vmware, aws, etc.

Deviation: `cloud-init-input`

Indicates how the contents of cloud-init script are provided. There are 2 choices - inline or in a file

Depending on the value of **`cloud-init-input`**, **`vdu`** may have additional fields.

When `cloud-init-input` **is** `'filename'`

The following fields are defined:

Field	Type	Description
`cloud-init-file`	**`string`**	Name of file with contents of cloud-init script in cloud-config format

When `cloud-init-input` **is** `'inline'`

The following fields are defined:

Field	Type	Description
`cloud-init`	**`string`**	Contents of cloud-init script, provided inline, in cloud-config format

manotypes:guest-epa

The following fields are defined:

Field	Type	Description

cpu-pinning-policy	**enumeration** default: **'ANY'**	CPU pinning policy describes association between virtual CPUs in guest and the physical CPUs in the host. • DEDICATED : Virtual CPUs are pinned to physical CPUs • SHARED : Multiple VMs may share the same physical CPUs. • ANY : Any policy is acceptable for the VM
cpu-thread-pinning-policy	**enumeration**	CPU thread pinning policy describes how to place the guest CPUs when the host supports hyper threads: • AVOID : Avoids placing a guest on a host with threads. • SEPARATE: Places vCPUs on separate cores, and avoids placing two vCPUs on two threads of same core. • ISOLATE : Places each vCPU on a different core, and places no vCPUs from a different guest on the same core. • PREFER : Attempts to place vCPUs on threads of the same core.
mempage-size	**enumeration**	Memory page allocation size. If a VM requires hugepages, it should choose LARGE or SIZE_2MB or SIZE_1GB. If the VM prefers hugepages it should choose PREFER_LARGE. • LARGE : Require hugepages (either 2MB or 1GB) • SMALL : Doesn't require hugepages • SIZE_2MB : Requires 2MB hugepages • SIZE_1GB : Requires 1GB hugepages • PREFER_LARGE : Application prefers hugepages
pcie-device	**list[1...N]**	List of pcie passthrough devices. See: manotypes:pcie-device
trusted-execution	**boolean**	This VM should be allocated from trusted pool

Deviation: `numa-policy`

Depending on the value of **numa-policy**, **guest-epa** may have additional fields.

When `numa-policy` **is** `'numa-aware'`

The following fields are defined:

Field	Type	Description
numa-node-policy		This policy defines NUMA topology of the guest. Specifically identifies if the guest should be run on a host with one NUMA node or multiple NUMA nodes. As an example a guest might need 8 VCPUs and 4 GB of memory. However, it might need the VCPUs and memory distributed across multiple NUMA nodes. In this scenario, NUMA node 1 could run with 6 VCPUs and 3GB, and NUMA node 2 could run with 2 VCPUs and 1GB.

	See: manotypes:numa-node-policy

manotypes:numa-node-policy

This policy defines NUMA topology of the guest. Specifically identifies if the guest should be run on a host with one NUMA node or multiple NUMA nodes. As an example a guest might need 8 VCPUs and 4 GB of memory. However, it might need the VCPUs and memory distributed across multiple NUMA nodes. In this scenario, NUMA node 1 could run with 6 VCPUs and 3GB, and NUMA node 2 could run with 2 VCPUs and 1GB.

The following fields are defined:

Field	Type	Description
mem-policy	**enumeration**	This policy specifies how the memory should be allocated in a multi-node scenario. • STRICT : The memory must be allocated strictly from the memory attached to the NUMA node. • PREFERRED : The memory should be allocated preferentially from the memory attached to the NUMA node
node	**list[1...N]**	See: manotypes:node
node-cnt	**uint16**	The number of NUMA nodes to expose to the VM.

manotypes:node

The following fields are defined:

Field	Type	Description
id	**uint64**	NUMA node identification. Typically it's 0 or 1
memory-mb	**uint64**	Memory size expressed in MB for this NUMA node.
vcpu	**list[1...N]**	List of VCPUs to allocate on this NUMA node. See: manotypes:vcpu

Deviation: `om-numa-type`

OpenMANO NUMA type selection

Depending on the value of **`om-numa-type`**, **`node`** may have additional fields.

When `om-numa-type` **is** `'cores'`

The following fields are defined:

Field	Type	Description
num-cores	**uint8**	Number of cores

When `om-numa-type` **is** `'paired-threads'`

The following fields are defined:

Field	Type	Description
paired-threads		See: manotypes:paired-threads

When `om-numa-type` **is** `'threads'`

The following fields are defined:

Field	Type	Description
`num-threads`	**`uint8`**	

manotypes:vcpu

List of VCPUs to allocate on this NUMA node.

The following fields are defined:

Field	Type	Description
`id`	**`uint64`**	List of VCPUs ids to allocate on this NUMA node

When `numa-policy` **is** `'numa-unaware'`

The following fields are defined:

Field	Type	Description
`numa-unaware`	**`empty`**	No policy is defined

manotypes:pcie-device

List of pcie passthrough devices.

The following fields are defined:

Field	Type	Description
`count`	**`uint64`**	Number of devices to attach to the VM.
`device-id`	**`string`**	Device identifier.

manotypes:host-epa

Specifies the host level EPA attributes.

The following fields are defined:

Field	Type	Description
`cpu-arch`	**`enumeration`**	Host CPU architecture. Possible values: • PREFER_X86 • REQUIRE_X86 • PREFER_X86_64 • REQUIRE_X86_64 • PREFER_I686 • REQUIRE_I686 • PREFER_IA64 • REQUIRE_IA64 • PREFER_ARMV7 • REQUIRE_ARMV7 • PREFER_ARMV8

		• REQUIRE_ARMV8
cpu-core-count	**uint64**	Number of cores on the host.
cpu-core-thread-count	**uint64**	Number of threads per cores on the host.
cpu-feature	**list[1...N]**	List of CPU features. See: manotypes:cpu-feature
cpu-model	**enumeration**	Host CPU model. Possible values: • PREFER_WESTMERE • REQUIRE_WESTMERE • PREFER_SANDYBRIDGE • REQUIRE_SANDYBRIDGE • PREFER_IVYBRIDGE • REQUIRE_IVYBRIDGE • PREFER_HASWELL • REQUIRE_HASWELL • PREFER_BROADWELL • REQUIRE_BROADWELL • PREFER_NEHALEM • REQUIRE_NEHALEM • PREFER_PENRYN • REQUIRE_PENRYN • PREFER_CONROE • REQUIRE_CONROE • PREFER_CORE2DUO • REQUIRE_CORE2DUO
cpu-socket-count	**uint64**	Number of sockets on the host.
cpu-vendor	**enumeration**	Host CPU Vendor. Possible values: • PREFER_INTEL • REQUIRE_INTEL • PREFER_AMD • REQUIRE_AMD
om-cpu-feature	**list[1...N]**	List of OpenMANO CPU features See: manotypes:om-cpu-feature
om-cpu-model-string	**string**	OpenMANO CPU model string

manotypes:cpu-feature

List of CPU features.

The following fields are defined:

Field	Type	Description
feature	**cpu-feature-type**	CPU feature. Possible values: **{REQUIRE|PREFER}_{X}**, where **X** is: • **AES**: CPU supports advanced instruction set for AES (Advanced Encryption Standard).

	• **CAT**: Cache Allocation Technology (CAT) allows an Operating System, Hypervisor, or similar system management agent to specify the amount of L3 cache (currently the last-level cache in most server and client platforms) space an application can fill (as a hint to hardware functionality, certain features such as power management may override CAT settings). • **CMT**: Cache Monitoring Technology (CMT) allows an Operating System, Hypervisor, or similar system management agent to determine the usage of cache based on applications running on the platform. The implementation is directed at L3 cache monitoring (currently the last-level cache in most server and client platforms). • **DDIO**: Intel Data Direct I/O (DDIO) enables Ethernet server NICs and controllers talk directly to the processor cache without a detour via system memory. This enumeration specifies if the VM requires a DDIO capable host.

manotypes:om-cpu-feature

List of OpenMANO CPU features

The following fields are defined:

Field	Type	Description
feature	**string**	CPU feature

manotypes:hypervisor-epa

The following fields are defined:

Field	Type	Description
type	**enumeration**	Specifies the type of hypervisor. • KVM: KVM • XEN: XEN
version	**string**	

vnfd-base:interface

List of Interfaces (external and internal) for the VNF

The following fields are defined:

Field	Type	Description
mac-address	**string**	MAC address of the interface. Some VNFs require a specific MAC address to be configured in the interface. While this is not recommended at all in NFV environments, this parameter exists to allow those scenarios. This parameter will be likely deprecated in the future.
name	**string**	Name of the interface. Note that this name has only local significance to the VDU.

position	**uint32**	Explicit Position of the interface within the list
type	**interface-type** default: **'EXTERNAL'**	Type of the Interface (INTERNAL or EXTERNAL)
virtual-interface		Container for the virtual interface properties. See: vnfd-base:virtual-interface

Deviation: connection-point-type

Depending on the value of **connection-point-type**, **interface** may have additional fields.

When connection-point-type **is** 'external'

The following fields are defined:

Field	Type	Description
external-connection-point-ref	**leafref**	Leaf Ref to the particular external connection point **'../../../connection-point/name'**

When connection-point-type **is** 'internal'

The following fields are defined:

Field	Type	Description
internal-connection-point-ref	**leafref**	Leaf Ref to the particular internal connection point **'../../internal-connection-point/id'**

vnfd-base:virtual-interface

Container for the virtual interface properties.

The following fields are defined:

Field	Type	Description
bandwidth	**uint64**	Aggregate bandwidth of the NIC.
type	**enumeration** default: **'VIRTIO'**	Specifies the type of virtual interface between VM and host. • VIRTIO : Use the traditional VIRTIO interface. • PCI-PASSTHROUGH : Use PCI-PASSTHROUGH interface. • SR-IOV : Use SR-IOV interface. • E1000 : Emulate E1000 interface. • RTL8139 : Emulate RTL8139 interface. • PCNET : Emulate PCNET interface. • OM-MGMT : Deprecated! Use VIRTIO instead and set the VNF management interface at vnfd:mgmt-interface:cp
vpci	**string**	Specifies the virtual PCI address. Expressed in the following format dddd:dd:dd.d. For example 0000:00:12.0. This information can be used to pass as metadata during the VM creation.

manotypes:supplemental-boot-data

The following fields are defined:

Field	Type	Description
boot-data-drive	**boolean** default: **'false'**	Some VIMs implement additional drives to host config-files or meta-data
config-file	**list[1...N]**	List of configuration files to be written on an additional drive See: manotypes:config-file

manotypes:config-file

List of configuration files to be written on an additional drive

The following fields are defined:

Field	Type	Description
dest	**string**	Full path of the destination in the guest
source	**string**	Name of the configuration file

vnfd-base:vdu-configuration

The following fields are defined:

Field	Type	Description
config-primitive	**list[1...N]**	List of config primitives supported by the configuration agent for this VNF or VDU. See: manotypes:config-primitive
initial-config-primitive	**list[1...N]**	Initial set of configuration primitives. See: manotypes:initial-config-primitive

Deviation: `config-method`

Defines the configuration method for the VNF or VDU.

Depending on the value of **config-method**, **vdu-configuration** may have additional fields.

When `config-method` **is** `'juju'`

Configure the VNF or VDU through Juju.

The following fields are defined:

Field	Type	Description
juju		See: manotypes:juju

When `config-method` **is** `'script'`

Use custom script for configuring the VNF or VDU. This script is executed in the context of Orchestrator (The same system and environment as the Launchpad).

The following fields are defined:

Field	Type	Description

script	See: manotypes:script

manotypes:config-primitive

List of config primitives supported by the configuration agent for this VNF or VDU.

The following fields are defined:

Field	Type	Description
name	**string**	Name of the config primitive.
parameter	**list[1...N]**	List of parameters to the config primitive. See: manotypes:parameter
user-defined-script	**string**	A user defined script. If user defined script is defined, the script will be executed using bash

manotypes:initial-config-primitive

Initial set of configuration primitives.

The following fields are defined:

Field	Type	Description
seq	**uint64**	Sequence number for the configuration primitive.

Deviation: `primitive-type`

Depending on the value of **primitive-type**, **initial-config-primitive** may have additional fields.

When `primitive-type` **is** `'primitive-definition'`

The following fields are defined:

Field	Type	Description
name	**string**	Name of the configuration primitive.
parameter	**list[1...N]**	List of parameters to the configuration primitive. See: manotypes:parameter
user-defined-script	**string**	A user defined script.

manotypes:vm-flavor

The following fields are defined:

Field	Type	Description
memory-mb	**uint64**	Amount of memory in MB.
storage-gb	**uint64**	Amount of disk space in GB.
vcpu-count	**uint16**	Number of VCPUs for the VM.

vnfd-base:volumes

The following fields are defined:

Field	Type	Description

description	**string**	Description for Volume
device-bus	**enumeration**	Type of disk-bus on which this disk is exposed to guest (ide, usb, virtio, scsi)
device-type	**enumeration**	The type of device as exposed to guest (disk, cdroom, floppy, lun)
name	**string**	Name of the disk-volumes, e.g. vda, vdb etc
size	**uint64**	Size of disk in GB

Deviation: `volume-source`

Defines the source of the volume. Possible options are:

1. Ephemeral – Empty disk
2. Image – Refer to image to be used for volume
3. Volume – Reference of pre-existing volume to be used

Depending on the value of **`volume-source`**, **`volumes`** may have additional fields.

When `volume-source` **is** `'ephemeral'`

The following fields are defined:

Field	Type	Description
ephemeral	**empty**	

When `volume-source` **is** `'image'`

The following fields are defined:

Field	Type	Description
image	**string**	Image name for the software image. If the image name is found within the VNF package it will be uploaded to all VIM accounts during onboarding process. Otherwise, the image must be added to the VIM account with the same name as entered here.
image-checksum	**string**	Image md5sum for the software image. The md5sum, if provided, along with the image name uniquely identifies an image uploaded to the CAL.

manotypes:vswitch-epa

The following fields are defined:

Field	Type	Description
ovs-acceleration	**enumeration**	Specifies Open vSwitch acceleration mode. • MANDATORY: OVS acceleration is required • PREFERRED: OVS acceleration is preferred
ovs-offload	**enumeration**	Specifies Open vSwitch hardware offload mode. • MANDATORY: OVS offload is required • PREFERRED: OVS offload is preferred

vnfd-base:vdu-dependency

List of VDU dependencies.

The following fields are defined:

Field	Type	Description
vdu-depends-on-ref	**leafref**	Reference to the VDU on which the source VDU depends. **'../../vdu/id'**
vdu-source-ref	**leafref**	**'../../vdu/id'**

vnfd-base:vnf-configuration

The following fields are defined:

Field	Type	Description
config-primitive	**list[1...N]**	List of config primitives supported by the configuration agent for this VNF or VDU. See: manotypes:config-primitive
initial-config-primitive	**list[1...N]**	Initial set of configuration primitives. See: manotypes:initial-config-primitive

5.3 Module: pnfd

The following fields are defined:

Field	Type	Description
pnfd-catalog		See: pnfd:pnfd-catalog

5.3.1 pnfd:pnfd-catalog

The following fields are defined:

Field	Type	Description
pnfd	**list[1...N]**	See: pnfd:pnfd

pnfd:pnfd

The following fields are defined:

Field	Type	Description
category	**string[1...N]**	List of categories that can be used to group PNFs providing similar features, e.g. **routing**, **access-point**.
connection-point	**list[1...N]**	List for external connection points. Each PNF has one or more external connection points. See: pnfd:connection-point
description	**string**	Description of the PNFD.
id	**uuid**	Identifier for the PNFD.

	pattern: `['[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}']`	
logical-type	**string**	Different network functions can be implemented in different ways and by different vendors, but perform basically equivalent logical operations. This field should be used to identify a high-level type that abstract the chosen implementation, allowing similar PNFs to be identified. Examples: firewall, ids, sbc, cpe …
name	**string**	PNFD name.
short-name	**string**	Short name to appear as label in the UI
vendor	**string**	Vendor of the PNFD.
version	**string**	Version of the PNFD

pnfd:connection-point

List for external connection points. Each PNF has one or more external connection points.

The following fields are defined:

Field	Type	Description
cp-type	**connection-point-type**	Type of the connection point.
id	**uint64**	Identifier for the external connection points

5.4 Module: vnffgd

The following fields are defined:

Field	Type	Description
vnffgd-catalog		See: vnffgd:vnffgd-catalog

5.4.1 vnffgd:vnffgd-catalog

The following fields are defined:

Field	Type	Description
vnffgd	**list[1...N]**	List of VNF Forwarding Graph Descriptors (VNFFGD). See: vnffgd:vnffgd

vnffgd:vnffgd↓

List of VNF Forwarding Graph Descriptors (VNFFGD).

The following fields are defined:

Field	Type	Description

classifier	**list[1...N]**	List of classifier rules. See: vnffgd:classifier
description	**string**	Description of the VNFFGD.
id	**uuid** pattern: **['[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}']**	Identifier for the VNFFGD.
name	**string**	VNF Forwarding Graph Descriptor name.
rsp	**list[1...N]**	List of Rendered Service Paths (RSP). See: vnffgd:rsp
short-name	**string**	Short name to appear as label in the UI
vendor	**string**	Provider of the VNFFGD.
version	**string**	Version of the VNFFGD

vnffgd:classifier

List of classifier rules.

The following fields are defined:

Field	Type	Description
Id	**string**	Identifier for the classifier rule.
match-attributes	**list[1...N]**	List of match attributes. See: vnffgd:match-attributes
Name	**string**	Name of the classifier.
rsp-id-ref	**leafref**	A reference to the RSP. **'../../rsp/id'**
vnfd-connection-point-name-ref	**leafref**	A reference to a connection point name in a vnfd. **'/vnfd:vnfd-catalog/vnfd:vnfd[vnfd:id = current()/../vnfd-id-ref]/vnfd:connection-point/vnfd:name'**
vnfd-id-ref	**leafref**	A reference to a vnfd. **'/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id'**

vnffgd:match-attributes

List of match attributes.

The following fields are defined:

Field	Type	Description
destination-ip-address	**ip-address**	Destination IP address.
destination-port	**port-number**	Destination port number.
Id	**string**	Identifier for the classifier match attribute rule.

ip-proto	**uint8**	IP Protocol.
source-ip-address	**ip-address**	Source IP address.
source-port	**port-number**	Source port number.

vnffgd:rsp

List of Rendered Service Paths (RSP).

The following fields are defined:

Field	Type	Description
Id	**string**	Identifier for the RSP.
Name	**string**	RSP name.
vnfd-connection-point-ref	**list[1...N]**	A list of references to connection points. See: vnffgd:vnfd-connection-point-ref

vnffgd:vnfd-connection-point-ref

A list of references to connection points.

The following fields are defined:

Field	Type	Description
Order	**uint8**	A number that denotes the order of a VNF in a chain
vnfd-connection-point-name-ref	**leafref**	A reference to a connection point name in a vnfd. **'/vnfd:vnfd-catalog/vnfd:vnfd[vnfd:id = current()/../vnfd-id-ref]/vnfd:connection-point/vnfd:name'**
vnfd-id-ref	**leafref**	A reference to a vnfd. **'/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id'**

5.5 Module: vld

The following fields are defined:

Field	Type	Description
vld-catalog		See: vld:vld-catalog

5.5.1 vld:vld-catalog

The following fields are defined:

Field	Type	Description
vld	**list[1...N]**	See: vld:vld

vld:vld

The following fields are defined:

Field	Type	Description
description	**string**	Description of the VLD.

id	**uuid** pattern: **['[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}']**	Identifier for the VLD.
leaf-bandwidth	**uint64**	For ELAN this is the bandwidth of branches.
name	**string**	Virtual Link Descriptor (VLD) name.
provider-network		Container for the provider network. See: manotypes:provider-network
root-bandwidth	**uint64**	For ELAN this is the aggregate bandwidth.
short-name	**string**	Short name for VLD for UI
type	**virtual-link-type**	
vendor	**string**	Provider of the VLD.
version	**string**	Version of the VLD
vnfd-connection-point-ref	**list[1...N]**	A list of references to connection points. See: vld:vnfd-connection-point-ref

vld:vnfd-connection-point-ref

A list of references to connection points.

The following fields are defined:

Field	Type	Description
member-vnf-index-ref	**uint64**	A reference to the consituent-vnfd id in nsd
vnfd-connection-point-ref	**leafref**	A reference to a connection point name in a vnfd **'/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:connection-point/vnfd:name'**
vnfd-ref	**leafref**	A reference to a vnfd **'/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id'**

6 Conclusions

Following the reference architecture presented in MATILDA D1.1 [3], this deliverable presented a set of metamodels that will be used within MATILDA to stablish network connectivity for the different parts of the 5G-ready Application Graph and to link computational nodes in a multi-side datacentre environment.

Models derived from the Network Service Descriptor will be used within the MATILDA framework in the north bound application programmable interface of the infrastructure provider administrative zone, so structured information can be exchanged between MATILDA Slice Broker and the telecommunication company OSS/BSS. Models derived from the other presented metamodels (VNFDs, PNFDs, VLDs and VNFFGDs) are referenced by the NSDs and provide the required information for using network functions (virtualised or not) and creating rich network topologies between them, following network requirements that are able to capture to business specificities.

The same metamodels will form the basis for the MATILDA ecosystem and community surrounding the Marketplace, by allowing developers to publish pieces of software that respect a unified interface and can be easily adopted by telecommunication companies and service providers that target different vertical industries.

Finally, despite of adopting the OSM approach [2], network services created with the framework will be reusable outside of MATILDA, since the descriptors can be easily translated to other formats, being virtually compatible with most of the existing solutions for VNF orchestration, thanks to the increasingly adoption of the ETSI standards they are based on.

The contents presented here will be complemented by similar documents created in MATILDA's first work package WP1 that describe related metamodels, specially the Slice Intent metamodel in D1.4 [5]. Moreover, the outputs of this deliverable will serve as input for the implementation work that will be performed in the following work packages, 2, 3 and 4.

7 References

[1] ETSI, 'GS NFV-MAN 001 - V1.1.1 - Network Functions Virtualisation (NFV); Management and Orchestration', 2014 [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf
[2] OSM, 'OSM Information Model', 2017 [Online]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Information_Model
[3] MATILDA, 'D1.1 - Framework and Reference Architecture', 2017 [Online]. Available: https://private.matilda-5g.eu/documents/PublicDownload/119
[4] MATILDA, 'D1.2 - Chainable Application Component and 5G-Ready Application Graph Metamodel', 2017 [Online]. Available: https://matilda-5g.eu/index.php/outcomes
[5] MATILDA, 'D1.4 - Slice Intent Metamodel', 2017 [Online]. Available: https://matilda-5g.eu/index.php/outcomes
[6] C. Buerger *et al.*, 'OSM RELEASE THREE - A Technical Overview', 2017 [Online]. Available: https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseTHREE-FINAL.pdf
[7] W3C, 'Extensible Markup Language (XML) 1.0 (Fifth Edition)' [Online]. Available: https://www.w3.org/TR/xml/. [Accessed: 26-Feb-2018]
[8] IETF, 'YANG -A Data Modeling Language for the Network Configuration Protocol (NETCONF)', 2010 [Online]. Available: http://www.rfc-editor.org/info/rfc6020.
[9] W3C, 'XML Path Language (XPath)' [Online]. Available: https://www.w3.org/TR/xpath/. [Accessed: 26-Feb-2018]
[10] OSM, 'OSM Information Model Repository'. 2017 [Online]. Available: https://osm.etsi.org/gitweb/?p=osm/IM.git;a=tree
[11] MATILDA, 'Source Code Repository for Metamodels'. 2018 [Online]. Available: https://gitlab.com/matilda-project/matilda-metamodels