NON: Network Function Virtualisation Ontology Towards Semantic Service Implementation
Luis Cuellar Hoyos
MEng. Candidate

Prof. Dr. Christian Esteve Rothenberg
Advisor
Agenda

- Introduction
- Problem Review
- Goals
- Background
- NFV Ontology (NOn)
  - Implementation & PoC
- Semantic nFV Services (SnS)
  - Implementation & PoC
- Conclusions and Future Work
1. Introduction

A NFV Overview
To change the current appliance model of network functions, trying to remove the typical constraints.
NFV Principal Components

VNF: Software Defined Function Networks

NFVI: Provide the resources necessaries (Physical and virtual) to deploy a VNF

MANO: in charge of orchestrate and manage all the aspects related to the NS and VNF deployment
NFV High Level Architecture
2. Problem Review
The cost of Interoperability
NFV Implementation - Local Domain

- Different developers
- Different Technologies
- Different Communication Protocols
- Different terminology

How to gain interoperability?
Integration Using Manual Intervention
Manual Intervention

➔ Read descriptions, user manuals and metadata

➔ Interpret and understand

➔ Use capabilities and functionalities
NFV Implementation - Inter Domain

How to gain interoperability?
Integration Using Manual Intervention
Manual Intervention

Build new blocks to integrate NFV components
Middlewares
Some Component Integrations

➔ Parser: Change whole component. Interface remain same

➔ Parser: Transforms requests, from one side to another. Both interfaces

➔ Adding new software layers. Both interfaces can be used
Interoperability Cost

COST

INTEGRATION

High

Low

Manual

Automatic
Problem Summary

- NFV implementations are done without common knowledge
- NFV descriptors are done in an arbitrary way
- Inherited the problems of WS
- Manual intervention to integrate services
- NFV Service descriptions call to be done in a implicit way
- Higher integration costs
“Which kind of common understanding, semantics and communication mechanism can be used on NFV implementations to reduce manual intervention in order to obtain interoperability with an automatic integration process?”

Research Question
3. Goals

Our contribution
Specific Goals

➔ Develop a Network Function Virtualisation ontology NO on using as a base the ETSI Virtual Function Network Descriptor (VNFD).

➔ Create a semantic representation of the VNFD.

➔ Implement NFV interfaces following a semantic service approach.

➔ Automate NFV service integration by using an NFV Ontology and a semantic service implementation.

➔ Validate the concept of NFV semantic services by proof of concept implementations showcasing automatic service integration
4. Background

Related technologies
Rest Web Services

- Client-Server
- Stateless Interactions
- Self-descriptive messages
- Uniform Interface
- Named Resources
- Interconnected resource representations
- Layered components
→ Provide a common syntax for machine-understandable statements
→ Establish common vocabularies as in an ontology
→ Agree on a logical language
→ Use the language to exchange proofs
RDF - Resource Description Framework

RDF Composition:

- Resources – the things being described
- Properties – the relationships between things
- Classes – the buckets used to group the things

Statements are in form of triples:

<Subject> <Predicate> <Object>
A super set logic language of RDF and extends the data model

@prefix ppl:<http://example.org/people#>.
@prefix foaf:<http://xmlns.com/foaf/0.1/>.
{
    ppl:Cindy foaf:knows ppl:John.
}
=>
{
    ppl:John foaf:knows ppl:Cindy .
}.
Semantic Web Services

Web Semantic + Web Services = Semantic Web Services
RESTdesc describes hyperlinks

- REST API have hypermedia links
- Machines should browse the Web like humans
- What does it mean to follow a link?
  - Explain to machines in RDF
  - Use if/then construct

![Diagram of hypermedia links](image-url)
Inference Engine

RULES -> Reasoning Process -> Knowledge Base

Inference
Open Project:
A common understanding of NFV
Non Roadmap

- Reading
- Abstracting
- Making Relationships
- Designing
- Building

MANO Components:
- NFVO_Component
- NFVM_Component
- VIM_Component

Descriptors:
- NSD
- VNFD
- VDU
- VLD

Component Property Object:
- NSD has VNFD
- VNFD has VDU
- VDU has VNFC
- VNFC has Connection Point
5.1 Low Level Elements
### Defining Model Elements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendor</td>
<td>Leaf</td>
<td>1</td>
<td>The vendor generating this VNFD.</td>
</tr>
<tr>
<td>vdu</td>
<td>Element</td>
<td>1...N</td>
<td>This describes a set of elements related to a particular VDU, see clause 6.3.1.2.</td>
</tr>
<tr>
<td>connection point</td>
<td>Element</td>
<td>1...N</td>
<td>This element describes an external interface exposed by this VNF enabling connection with a Virtual Link, see clause 6.3.1.4 (see note).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Slot Type</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnfd</td>
<td>Object</td>
<td>1</td>
<td>N/A</td>
<td>VNFD</td>
</tr>
<tr>
<td>vdu</td>
<td>Object</td>
<td>1...*</td>
<td>N/A</td>
<td>VNFD</td>
</tr>
<tr>
<td>vendor</td>
<td>Slot</td>
<td>1...*</td>
<td>String</td>
<td>VNFD</td>
</tr>
</tbody>
</table>
## Extending Model Elements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Leaf</td>
<td>1</td>
<td>A unique identifier of this VDU within the scope of the VNFD, including version functional description and other identification information. This will be used to refer to VDU when defining relationships between them.</td>
</tr>
<tr>
<td>vm image</td>
<td>Element</td>
<td>1...N</td>
<td>This provides a reference to a VM image</td>
</tr>
<tr>
<td>vnfc</td>
<td>Element</td>
<td>1...N</td>
<td>Defines minimum and maximum number of instances which can be created to support scale out/in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Slot Type</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnfd</td>
<td>Object</td>
<td>1</td>
<td>N/A</td>
<td>VNFD</td>
</tr>
<tr>
<td>vdu</td>
<td>Object</td>
<td>1...*</td>
<td>N/A</td>
<td>VNFD</td>
</tr>
<tr>
<td>vendor</td>
<td>Slot</td>
<td>1...*</td>
<td>String</td>
<td>VNFD</td>
</tr>
<tr>
<td>virtual_memory_resource_element</td>
<td>Slot</td>
<td>1</td>
<td>Integer</td>
<td>VDU</td>
</tr>
<tr>
<td>id</td>
<td>Slot</td>
<td>1</td>
<td>String</td>
<td>VDU</td>
</tr>
<tr>
<td>vm_image</td>
<td>Slot</td>
<td>0...1</td>
<td>anyURI</td>
<td>VDU</td>
</tr>
<tr>
<td>vnfc</td>
<td>Object</td>
<td>1...*</td>
<td>N/A</td>
<td>VDU</td>
</tr>
</tbody>
</table>
5.3 Relationships
## Defining Relationships

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Property</th>
<th>Cardinality</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNFD</td>
<td>id</td>
<td>1</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>descriptor_version</td>
<td>1</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>vnf_version</td>
<td>1</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>has_vdu</td>
<td>1...*</td>
<td>vdu</td>
</tr>
<tr>
<td></td>
<td>vendor</td>
<td>1</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>has_deployment_flavor</td>
<td>0...*</td>
<td>deployment_flavor</td>
</tr>
<tr>
<td></td>
<td>has_connection_point</td>
<td>1...*</td>
<td>connection_point</td>
</tr>
<tr>
<td>VDU</td>
<td>virtual_memory_resource_element</td>
<td>1</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>scale_in_out</td>
<td>0...1</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>computation_requirement</td>
<td>1</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>id</td>
<td>1</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>vm_image</td>
<td>0...1</td>
<td>anyURI</td>
</tr>
<tr>
<td></td>
<td>has_vnfc</td>
<td>1...*</td>
<td>vnfc</td>
</tr>
<tr>
<td>VNFC</td>
<td>id</td>
<td>1</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>has_connection_point</td>
<td>1...*</td>
<td>connection_point</td>
</tr>
</tbody>
</table>
5.2
High Level Elements
NFV Framework Elements

Root/Functional_Blocks

- Manager and Orchestrator (MANO)
  - NFV Orchestrator Component
  - VNF Manager Component
  - VIM Component

- NFV Infrastructure {INDL}
  - Hardware Components {Resources}
    - Compute Component
      - CPU {Node_Component}
      - Memory {Node_Component}
    - Network Component {SwitchingMatrix}
    - Storage Component {Node_Component}
  - Virtualisation Components
    - Virtual Compute Component {Virtual_Node}
    - Virtual Network Component {Virtual_Node}
    - Virtual Storage Component {Virtual_Node}

- Hyper-visor

- Descriptors
  - VNFD
  - Virtual Link Descriptor (VLD)
  - VDU

- VNF
5.4 Model
NFV Ontology
5.4 Implementation
NOn Main Classes

```
<owl:Class rdf:about = "#functional_blocks">
  <rdfs:subClassOf rdf:resource = " #nfv "/>
</owl:Class >

<owl:Class rdf:about = "#mano">
  <rdfs:subClassOf>
    <owl:Class rdf:ID = "functional_blocks"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about = "#descriptors">
  <rdfs:subClassOf>
    <owl:Class rdf:about = "#functional_blocks"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID = "#vnfd">
  <rdfs:subClassOf >
    <owl:Class rdf:about = "#descriptors"/>
  </rdfs:subClassOf >
</owl:Class >
```

```
### NOn Properties

<table>
<thead>
<tr>
<th>Object properties</th>
<th>Datatype Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>non:has_connection_point</td>
<td>non:computation_requirement</td>
</tr>
<tr>
<td>non:has_constituent_vdu</td>
<td>non:connection_points_references</td>
</tr>
<tr>
<td>non:has_deployment_flavour</td>
<td>non:connectivity_type</td>
</tr>
<tr>
<td>non:has_vdu</td>
<td>non:constituent_vnfic</td>
</tr>
<tr>
<td>non:has_virtual_link</td>
<td>non:constraint</td>
</tr>
<tr>
<td>non:has_vnfc</td>
<td>non:dependency</td>
</tr>
<tr>
<td></td>
<td>non:descriptor_version</td>
</tr>
<tr>
<td></td>
<td>non:id</td>
</tr>
<tr>
<td></td>
<td>non:lifecycle_event</td>
</tr>
<tr>
<td></td>
<td>non:number_of_instances</td>
</tr>
<tr>
<td></td>
<td>non:scale_in_out</td>
</tr>
<tr>
<td></td>
<td>non:type</td>
</tr>
<tr>
<td></td>
<td>non:vdu_reference</td>
</tr>
<tr>
<td></td>
<td>non:vendor</td>
</tr>
<tr>
<td></td>
<td>non:virtual_link_reference</td>
</tr>
<tr>
<td></td>
<td>non:virtual_memory_resource_element</td>
</tr>
<tr>
<td></td>
<td>non:virtual_network_bandwidth_resource</td>
</tr>
<tr>
<td></td>
<td>non:vm_image</td>
</tr>
<tr>
<td></td>
<td>non:vnf_version</td>
</tr>
</tbody>
</table>
Property Facets

Range and Domain

Cardinality
Properties Data Model

Object Properties

<owl:ObjectProperty rdf:about="#has_vdu">
    <rdfs:range rdf:resource="#vdu"/>
    <rdfs:domain rdf:resource="#vnfd"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#has_deployment_flavour">
    <rdfs:range rdf:resource="#deployment_flavour"/>
    <rdfs:domain rdf:resource="#vnfd"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#has_vnfc">
    <rdfs:range rdf:resource="#vnfc"/>
    <rdfs:domain rdf:resource="#vdu"/>
</owl:ObjectProperty>

Data Properties

<owl:DatatypeProperty rdf:about="#vnf_version">
    <rdfs:domain rdf:resource="#vnfd"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#dependency">
    <rdfs:domain rdf:resource="#vnfd"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#virtual_link_reference">
    <rdfs:domain rdf:resource="#connection_point"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="vnfd">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#descriptors"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="has_virtual_link"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="has_connection_point"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="vnf_version"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
5.5

Use Cases

Semantic Description Model
Use VNFD files from current NFV deployments

Parse the elements into a semantic file (Protégé).

Compare original file and the semantic one.
From Syntax to Semantics

NFV Descriptor

Semantic NFV Descriptor
Use Case: OpenBaton

**VNFD**

```
{
    "vendor":"fokus",
    "version":"0.2",
    "name":"iperf-server",
    "type":"server",
    "endpoint":"generic",
    "virtual_link": [{"name":"private"}],
    "lifecycle_event": [{"event":"INSTANTIATE"}],
    "deployment_flavour": [{"flavour_key":"m1.small"}],
    "vdu": [{
        "vm_image": "ubuntu-14.04-server-cloudimg-amd64-disk1",
        "vimInstanceName": "vim-instance",
        "scale_in_out": 2
    }]
}
```

**Semantic VNFD**

```sparql
###  https://github.com/LCuellarH/NOn/blob/master/datamodel/non.owl#ob-vnfd-1
non:ob-vnfd-1 rdf:type owl:NamedIndividual ,
    non:vnfd ;
    non:descriptor_version "0.2"^^xsd:string ;
    non:vnf_version "0.2"^^xsd:string ;
    non:lifecycle_event "INSTANTIATE-install.sh-install-srv.sh"^^xsd:string ;
    non:vendor "fokus"^^xsd:string ;
    non:id "iperf-server"^^xsd:string ;
    non:has_vdu non:ob-vdu-1 ;
    non:has_virtual_link non:ob-vld-1 ;
    non:has_deployment_flavour non:os-m1-small .
```
Semantic nFV services

Reasoning NOn
SnS Roadmap

Create WS → Create Descriptions → Generic Client → PoC
Creating OpenBaton VNFD WebService

Params:
- vendor
- name
- vm_image
- virtual_link
- lifecycle
- dev_flavour
- scaleinout

```
"vendor":"fokus",
"version":"0.2",
"name":"iperf-server",
"type":"server",
"endpoint":"generic",
"virtual_link": [{"name":"private"}],
"lifecycle_event": [{"event":"INSTANTIATE"}],
"deployment_flavour": [{"flavour_key": "m1.small"}],
"vdu": [{
  "vm_image": "ubuntu-14.04-server-cloudimg-amd64-disk1",
  "vimInstanceName": "vim-instance",
  "scale_in_out": 2
}]
```
## Adding Semantics NFV Services

<table>
<thead>
<tr>
<th>RESTdesc</th>
<th>Service Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unified VNFD WS</td>
</tr>
<tr>
<td></td>
<td>OpenBaton VNFD WS</td>
</tr>
<tr>
<td></td>
<td>VNFD WS</td>
</tr>
</tbody>
</table>
{  
  #Pre-conditions  
  ?vnfd a non:vnfd;  
    non:id ?vnfd_id;  
    non:descriptor_version ?ver_des;  
    non:has_vdu ?vdu .  
  ...
  ?vdu a non:vdu;  
    non:vm_image ?vm_image  
  ...
  ?vl a non:vld;  
    non:connectivity_type ?vl_type.  
}

=>

{  
  #Process  
  _:request http:methodName "GET";
    http:MessageHeader "Content-Type:application/json";  
    http:req[http:body json:openbaton_vnfd].  

  #Post-conditions  
  ?vnf non:has_vnfd ?non_vnfd.  
}..
Generic Client

Adapting itself to consume workflows without using a predefined context or background
Consuming Semantic Services

Generic Client → VNFD → Service Descriptor → Inference Engine

- Header: URI
- Response: Method
- Params: Param

Rest Request:
- Method: GET
- URI: ...vnf/vnfd?vendor=fokus...
- Content-Type: application/json
SnS Workflow Inference

Service Descriptions
Semantic VNFD
Goal

Logic Workflow

Goal
Service n
Precondition: n-1
Postcondition: Goal

Service n-1
Precondition: 1
Postcondition: n-1

Service 1
Precondition: VNFD
Postcondition: 1

Inference Engine

Rules
Ontology
Consuming a Goal-Based Workflow

Semantic VNFD

Goal: Metadata

Generic Client

RESTdesc OpenBaton VNFD WS

RESTdesc OpenBaton Metadata WS

RESTdesc OpenBaton WS

Rules

Ontology

Reasoner

VNF Deployment Workflow
Consume R1

Consume R2

OpenBaton VNFD WS

OpenBaton Metadata WS

OpenBaton WS
Use Case I

VNFD Benchmarking
Consuming Multiple NFV Services

1. Get OpenBaton VNFD
   - Open Baton WS Client
   - Open Baton WS Server

2. Get OpenMano VNFD
   - Open Baton WS Client
   - OpenMano WS Server

3. Get NFV VNFD
   - Open Baton WS Client
   - NFV WS Server
Scenario

Inference Engine

Rules

Semantics

Service Descriptions

Semantic VNFD

OpenMano VNFD Generator WS

OpenBaton VNFD Generator WS

NFV VNFD Generator WS

Consume OpenBaton

Generic Client
Use Case II

Deploying a VNF Semantic Service
OpenBaton VNF Deployment Process

1. VNFD
2. Metadata
3. Process
4. TAR

Network Functions

Virtualisation Infrastructure

cloud network
Scenario

**OpenBaton VNFD WS**
- REST desc

**OpenBaton Metadata WS**
- REST desc

**Semantic VNFD**

**Goal: Deploy VNF**

**Cloud**

**Generic Client**

**OpenBaton Service Description**

**NFVI**

**NFVO**

**VNFM**

**VIM openstack™**

**MANO**

**Inference Engine**

**Semantic VNFD**

**Goal: Deploy VNF**

**Cloud**

**Generic Client**

**OpenBaton Service Description**
Sequence Diagram

1: SVFND & Goal
2: Inputs
3: Workflow
4: Process Workflow
5: GET VNFD
6: JSON VNFD
7: GET Metadata
8: YAML Metadata
9: Create VNF Package
10: POST Upload
11: OpenBaton Package ID
7. Conclusions and Future Work
Conclusions and Limitations

Interoperability gaps were identified and attempted to be removed by the implementation of a common NFV data model known NOn.

Projects based on the information models defined by the ETSI exhibit better chance inter-working in multi-domain scenarios without requiring manual intervention.

NOn opened the door to create Semantic nFV Services. The implementation of REST interfaces, explicit service descriptions and the use of inference engines.

A Generic Client was capable of self adapting to consume dynamic REST Web Service workflows without the need of humans in the loop.

Open source NFV projects are currently not following ETSI specifications in the syntax definition.

Current NFV implementations do not use semantic technologies in their developments, implementation of semantic technologies were scoped just to component interfaces.
Conclusions and Limitations

The implementation of NOn and SnS are an initial step towards automatic service integration.

The full potential was not explored in the sense of leveraging of semantic technologies to build full components.

The current state of NOn does not fulfill all the needs of a complete data model capable of describing all possible VNFD files.
Contributions and Related Work

- A structured knowledge representation and a common language on the NFV domain, Non.
- A Virtual Network Functions Descriptor with a semantic approach.
- The concept of semantic services for NFV implementations
- A Generic Client capable of read, interpret and consume dynamic workflows.
Future Work

MANO

VNFM

NFVO

Inference Engine

Rules

Ontology

Semantic Service, VNF and Infrastructure Descriptions

VNFs

NFVI

VIM

MANO

SnS

SnS

SnS

SnS
Thanks!

Any questions?

You can find me at:
@lcuellarh
luiscuellarh@gmail.com
## Related Work - Description Models

<table>
<thead>
<tr>
<th>Network Modeling Language</th>
<th>Share Ontology</th>
<th>Semantic Description</th>
<th>Web Semantic Approach</th>
<th>Web Service Interoperability</th>
<th>Autonomous Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure Network Description Language</td>
<td>Share Ontology</td>
<td>Semantic Description</td>
<td>N/A</td>
<td>N/A</td>
<td>Can be use</td>
</tr>
<tr>
<td>Resource Information Service</td>
<td>Stand Alone Ontology</td>
<td>-</td>
<td>Manual Intervention External Plugin</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table:

- **Network Modeling Language**
  - Share Ontology
  - Semantic Description
  - Web Semantic Approach: Semantic Description
  - Web Service Interoperability: N/A
  - Autonomous Interoperability: Can be use

- **Infrastructure Network Description Language**
  - Share Ontology
  - Semantic Description
  - Web Semantic Approach: Semantic Description
  - Web Service Interoperability: N/A
  - Autonomous Interoperability: Can be use

- **Resource Information Service**
  - Stand Alone Ontology
  - Web Semantic Approach: -
  - Web Service Interoperability: Manual Intervention External Plugin
  - Autonomous Interoperability: -
## Related Work - Interoperability

<table>
<thead>
<tr>
<th></th>
<th>Semantic Description Defined by</th>
<th>Web Semantic Approach</th>
<th>Web Service Interoperability</th>
<th>Autonomous Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDN Rest API</strong></td>
<td>Developers</td>
<td>-</td>
<td>Manual Intervention REST Protocol</td>
<td>-</td>
</tr>
<tr>
<td><strong>ML2</strong></td>
<td>Developers Metadata</td>
<td>-</td>
<td>Manual Intervention External Plugin</td>
<td>-</td>
</tr>
<tr>
<td><strong>OpenStack</strong></td>
<td>Developers Metadata</td>
<td>-</td>
<td>Manual Intervention External Plugin</td>
<td>-</td>
</tr>
<tr>
<td><strong>RESTdesc</strong></td>
<td>Share Ontologies</td>
<td>Semantic Services</td>
<td>REST Protocol</td>
<td>Service Functionality Discovery</td>
</tr>
</tbody>
</table>

- SDN Rest API: Developers
- ML2: Developers Metadata
- OpenStack: Developers Metadata
- RESTdesc: Share Ontologies
## Related Work - NFV Implementations

<table>
<thead>
<tr>
<th></th>
<th>Semantic Description Defined by</th>
<th>Web Semantic Approach</th>
<th>Web Service Interoperability</th>
<th>Autonomous Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T-NOVA</strong></td>
<td>Developers Metadata</td>
<td>-</td>
<td>Manual Intervention External Plugin</td>
<td>-</td>
</tr>
</tbody>
</table>
Achieving Our Goal!

Semantics

Manual

Automatic Integration

High

INTEGRATION

COST

Low

Manual

Automatic

$
NFV High Level Architecture
We Wonder…..

Why if a component developed in a domain A, such NFVI can not just be “Plug and Play” in a domain B and use it by other components, such as MANO?

Why a developed VNF can not have a descriptor that multiple NFV domains can interpret and use?

How different NFV implementations following the same ETSI specifications could have the same capabilities independently of the development technologies by sharing common terminology to define the NFV elements?
OpenBaton

- All elements belonging to ETSI specifications were able to be mapped, proprietary elements do not
- Some elements can be mapped using other components of the ontology
- Follows ETSI specifications in the syntax
- Easy to map

OpenMano

- Few elements belonging to ETSI specifications were able to be mapped, proprietary elements do not
- Some elements can be mapped using a manual intervention over the semantic file
- Does not follow ETSI specifications in the syntax
- Difficult to Map
Use Case I: First Concept

MANO A (OpenBaton)

PoP

MANO Y (Vendor)

PoP

MANO X (OpenMano)

PoP

VNFD
## Use Case II: Results

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Goal</th>
<th>Expected Result</th>
<th>Expected vs Obtained Results</th>
<th>Modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario I</td>
<td>Consume OpenBaton SnS</td>
<td>OpenBaton SnS Consumed</td>
<td>Service not Consumed: VIM component missed</td>
<td>VIM component was added manually</td>
</tr>
<tr>
<td>Scenario II</td>
<td>Consume OpenMano SnS</td>
<td>OpenMano SnS Consumed</td>
<td>Service not Consumed: several components missed</td>
<td>Other components of NOn were added manually, components outside of NOn were deleted</td>
</tr>
<tr>
<td>Scenario III</td>
<td>Consume any SnS</td>
<td>SnS Not Consumed</td>
<td>OpenMano service was consumed</td>
<td>VIM component was added manually</td>
</tr>
</tbody>
</table>
Use Case III

OpenBaton - Unify Integration Proposal
Scenario

Inference Engine

OpenBaton VNFD WS

OpenBaton Metadata WS

Semantic VNFD

Goal: Deploy VNF

OpenBaton Service Description

OpenBaton

NFVI

VIM

MANO

NFVO

VNFM

LAB

86
Minicons Free Vector Icons Pack by Webalys is published under a Creative Commons Attribution license and Free for both personal and commercial use. You can copy, adapt, remix, distribute or transmit it. If you use this set on your presentation remember to keep the "Credits" slide or provide a mention of this "Minicons Free Vector Icons Pack" and a link back to this page: [http://www.webalys.com/minicons](http://www.webalys.com/minicons)