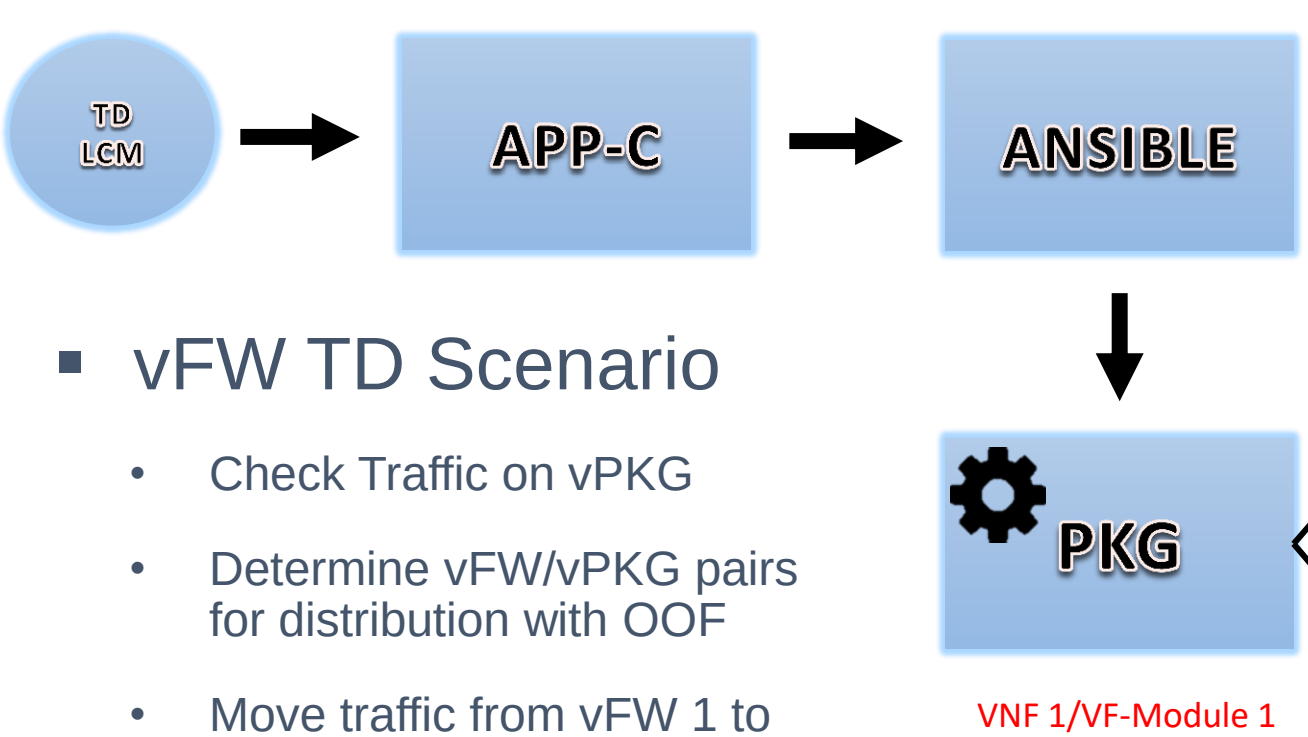
VNFC Level LCM Actions in ONAP

Problem statement and proposals for solution

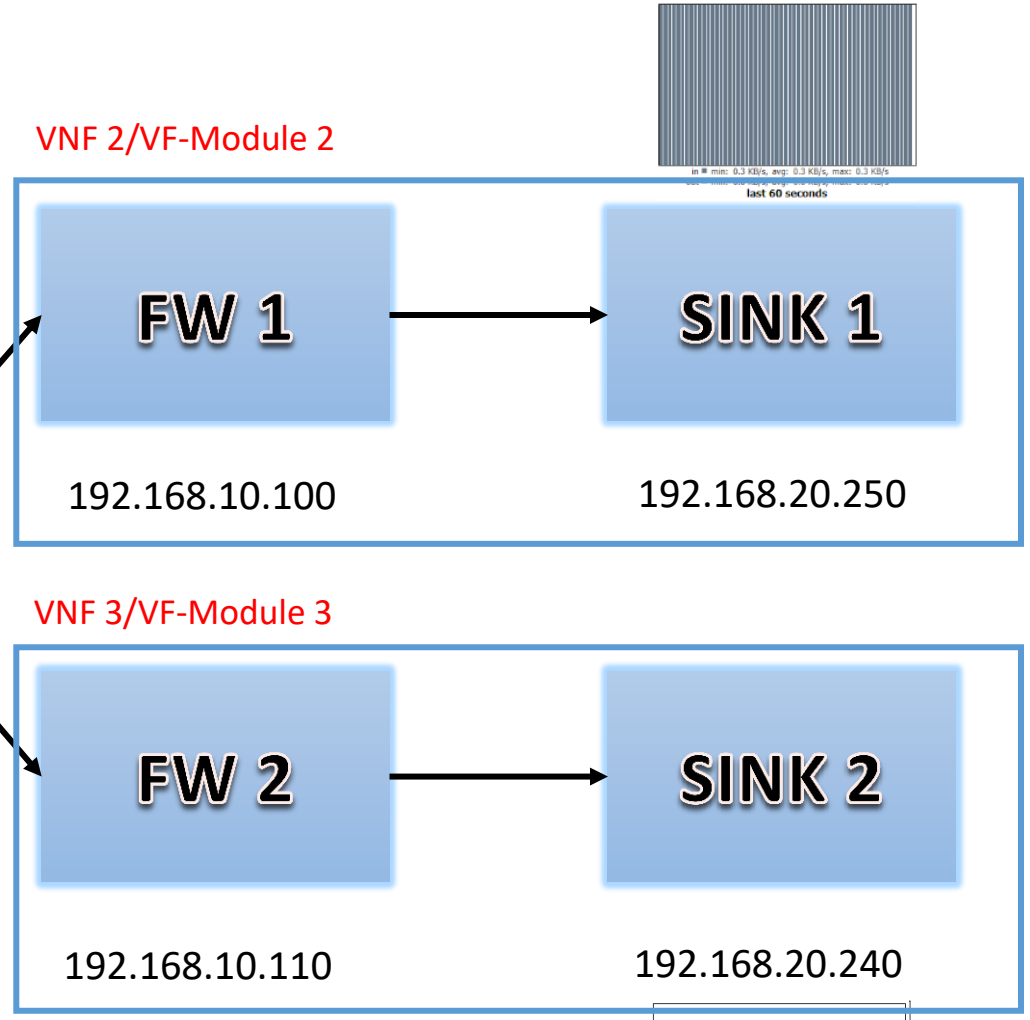
Łukasz Rajewski (Orange)
Yuriy Malakov (AT&T)
Scott Blandford (AT&T)

14.01.2020

vFW Traffic Distribution UC in El Alto – Impact of Service Design



- vFW TD Scenario
  - Check Traffic on vPKG
  - Determine vFW/vPKG pairs for distribution with OOF
  - Move traffic from vFW 1 to vFW 2 reconfiguring vPGN
  - Check Traffic on vFW 1
  - Check Traffic on vFW 2



LCM Action Execution on APPC Example Today

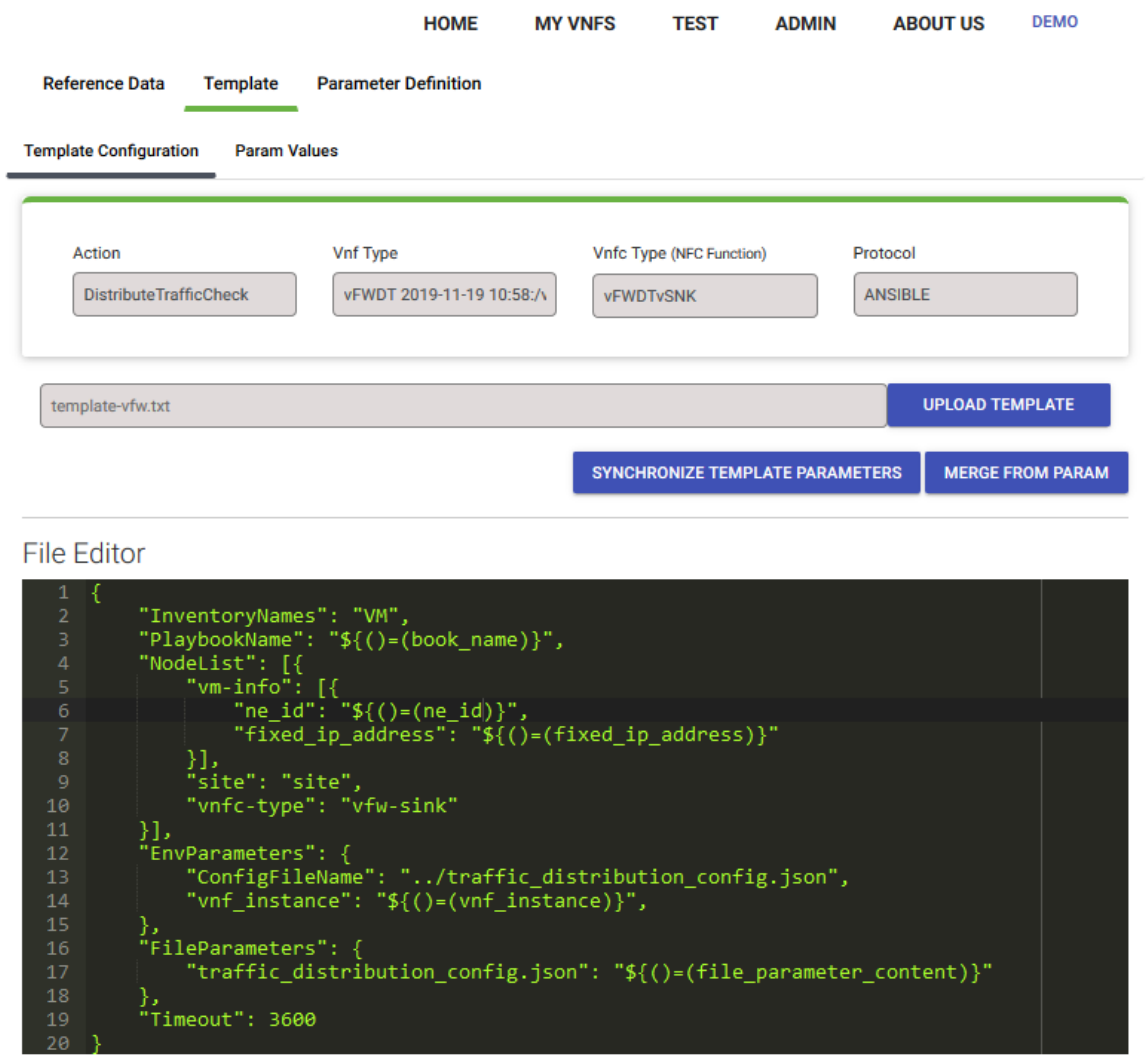
```
{
    "input": {
        "common-header": {
            "timestamp": "<TIMESTAMP>",
            "api-ver": "<API_VERSION>",
            "originator-id": "<SYSTEM_ID>",
            "request-id": "<REQUEST_ID>",
            "sub-request-id": "<SUBREQUEST_ID>",
            "flags": {
                "mode": "<EXCLUSIVE|NORMAL>",
                "force": "<TRUE|FALSE>",
                "ttl": "<TTL_VALUE>"
            }
        },
        "action": "<COMMAND_ACTION>",
        "action-identifiers": {
            "vnf-id": "<VNF_ID>",
            "vnfc-name": "<VNFC_NAME>",
            "vserver-id": "VSERVER_ID"
        },
        "payload": "{
            \"request-parameters\": {...},
            \"configuration-parameters\": {...}
        }"
    }
}
```

- **action** – LCM action to execute
- **action-identifiers** – identifies object to be modified – Today only vnf-id is accepted and vserver-id for OpenStack LCMs
- **request-parameters** [Optional] – request specific parameters i.e. like vf-module-id in ConfigScaleOut – interpreted by APPC
- **configuration-parameters** [Optional] – action specific parameters – merged with CDT template

We always want to execute LCM operation of concrete device(s) – VMs/PNFs

Determination how to access the device should be internal to the controller

CDT Template Definition Example for Ansible – vFWDT UC (1)



- book_name – name of Ansible playbook – may be a fixed value
- vnf_instance – should be a name of vnf instance
- NodeList – list of VM on which ansible playbook will be executed
  - ne_id – host name of VM – must be configured before in Ansible inventory file
  - fixed_ip_address – IP Address of VM – should be ONAP OAM address because Ansible must be able to reach this IP
  - Both can be taken from the AAI - from vnfc configuration

CDT Template Definition Example for Ansible – vFWDT UC (2)

Ansible Request Template in CDT

```
{
    "InventoryNames": "VM",
    "PlaybookName": "${()=(book_name)}",
    "NodeList": [{
        "vm-info": [{
            "ne_id": "${()=(ne_id)}",
            "fixed_ip_address": "${()=(fixed_ip_address)}"
        }],
        "site": "site",
        "vnfc-type": "vfw-sink"
    }],
    "EnvParameters": {
        "ConfigFileName": "../traffic_distribution_config.json",
        "vnf_instance": "vfwdt",
    },
    "FileParameters": {
        "traffic_distribution_config.json": "${()=(file_parameter_content)}"
    },
    "Timeout": 3600
}
```

Request merged with configuration-parameters

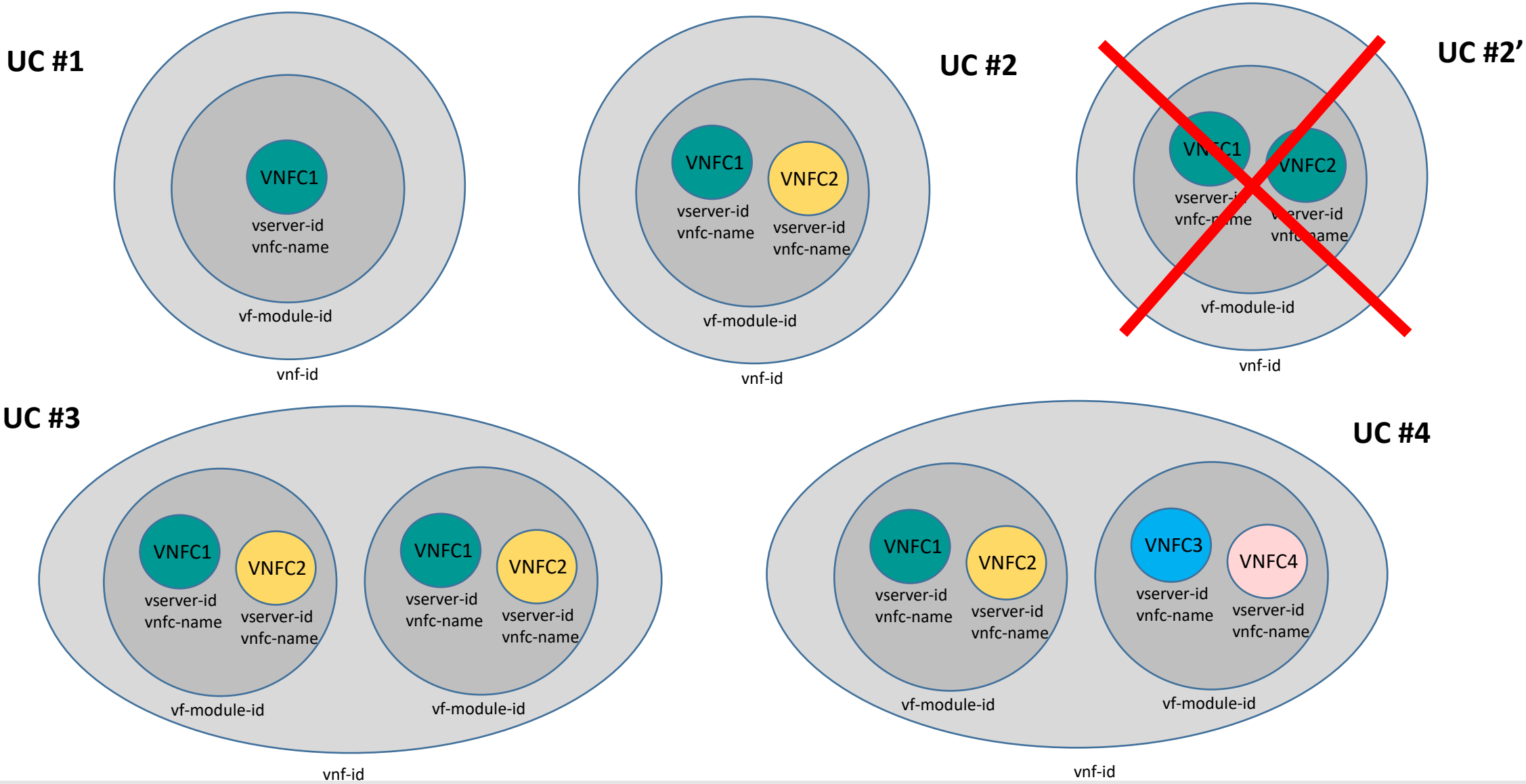
```
{
    "EnvParameters": {
        "ConfigFileName": "../traffic_distribution_config.json",
        "vnf_instance": "vfwdt"
    },
    "FileParameters": {
        "traffic_distribution_config.json": "{...}"
    },
    "InventoryNames": "VM",
    "NodeList": [
        {
            "site": "site",
            "vm-info": [
                {
                    "fixed_ip_address": "10.0.110.4",
                    "ne_id": "vfwl02vfw5190"
                }
            ],
            "vnfc-type": "vfw-sink"
        }
    ],
    "PlaybookName": "vfw-sink/latest/ansible/distributetrafficcheck/site.yml",
    "Timeout": "3600"
}
```

CDT Template Definition Example for Ansible – vFWDT UC (3)


```
{
    "input": {
        "action": "DistributeTraffic",
        "payload": "{
            \"configuration-parameters\": {
            \"fixed_ip_address\": \"10.0.110.4\",
            \"ne_id\": \"vfwl02vfwe22a\",
            \"file_parameter_content\": \"...\",
            \"book_name\": \"vfw-sink/latest/ansible/distributetrafficch
        "action-identifiers": {
            "vnf-id": "243ddlcl-88d7-4f3a-a44f-cce95d71897a"
        },
        "common-header": {
            "timestamp": "2019-11-26T09: 53: 30.244Z",
            "request-id": "7d9da48b-05bc-431d-a229-21cb0a14fd19",
            "originator-id": "vfw-dt-demo",
            "sub-request-id": "14bda5c7-acc7-4575-ac87-34f8b0a094bd",
            "flags": {
                "ttl": 36000,
                "force": "TRUE",
                "mode": "NORMAL"
            },
            "api-ver": "2.00"
        }
    }
}
```

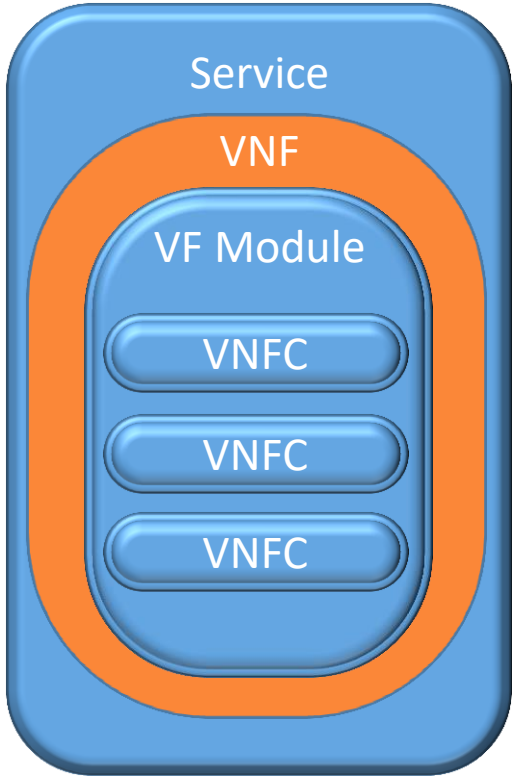

- Here concrete device is being identified by vnf-id + configuration-parameters
- In this case SO needs to find IP address, Hostname or receives this information from the workflow input parameters
- Ideally SO should identify concrete vf-module and vnfc-instance if there is more than one VM in VNF
- Identification may be much simpler if VNF would have its own VNFC controller
  - None of ONAP use cases (vLB/DNS, vFW) has it

Different types of VNF models in ONAP



Current SO Building Blocks Supported Use Cases

SO Building Block implementation:
The SO building blocks are a set of database-driven, configurable and generic process steps to be leveraged through several actions defined as 'Macro' flows. For each of the macro flows, there are a set of actions to the orchestration services and various type of resources orchestrated by ONAP.



<u>Supported Services & Resource types</u>
These resource types are essentially the ones defined in the model - through the SDC framework. SO orchestrates service, vnf and vfModule building block for assign, create configure and activate.

There is a lack of vnfc orchestration in ONAP that is required in order to support complex lifecycle management for various vnf use case.

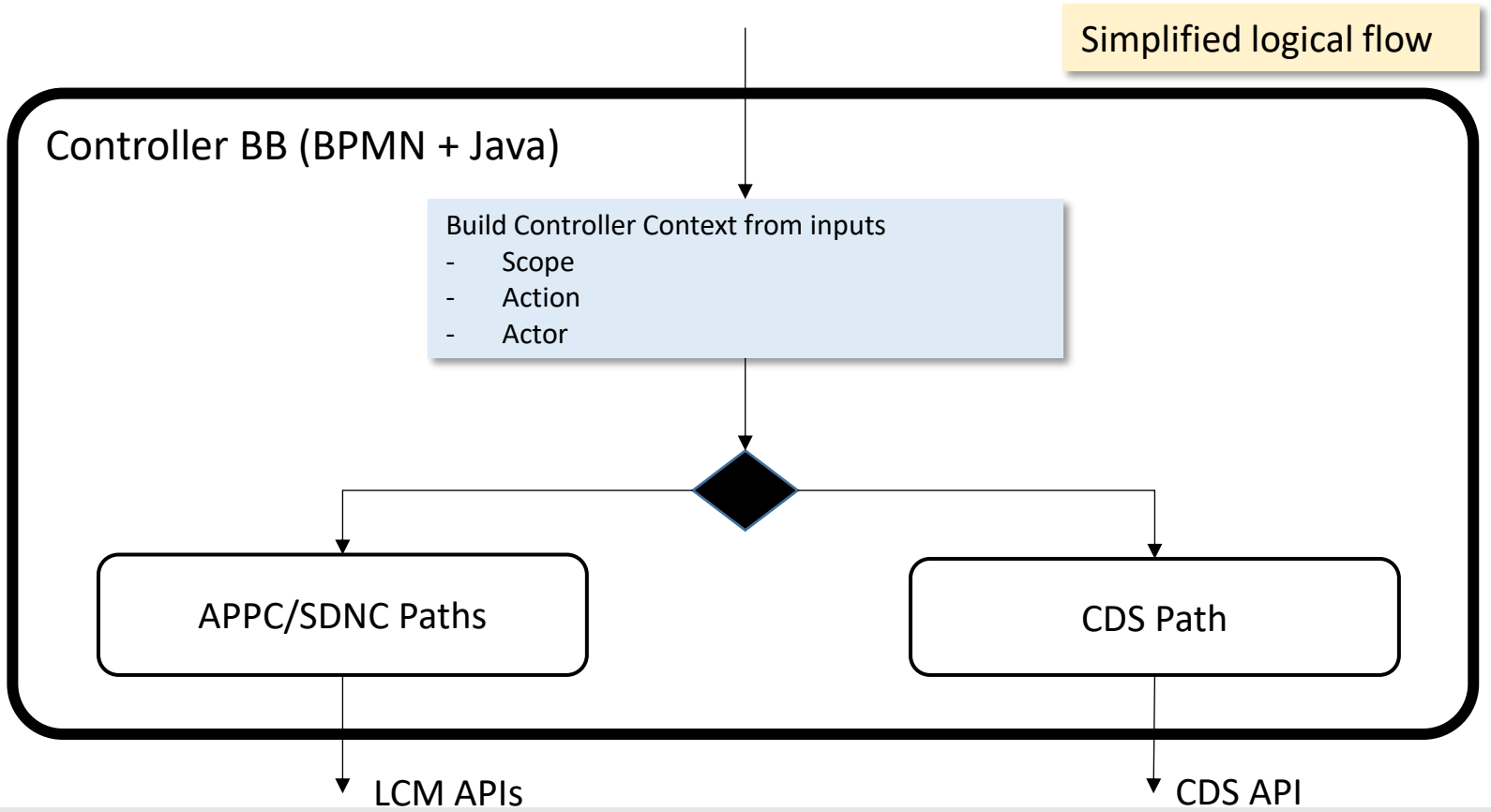
- ☑ Services
- ☑ VNF (Virtual Network Function)
- ☑ VF modules (i.e. a deployment unit, such as a HEAT stack)
- ✗ VNFC (virtual network function component)

SO MACRO Generic Building Block Orchestration

Id	Composite_Action	SEQ_NO	FLOW_NAME	FLOW_VERSION	LOOKUP_ID	SCOPE	ACTION
1	Service-Macro-Assign	1	ControllerExecutionBB	1	1	vnf	config-assign
2	Service-Macro-Assign	2	ControllerExecutionBB	1	1	vfModule	config-deploy

Scope
- service
- pnf
- vnf
- vfModule
- vnfc (to be added)

NOTE: Scope drives input action identifiers to controller LCM execution such as ssid, vnf-id, vf-module-id, vnfc-name, etc..



LCM/Action Execution on VNFC Level GAPs

- **How SO should determine the vf-module-id?**
  - Well known from the context in the instantiation time
  - Well known from the context in the ScaleOut time
  - There are situations when it should be resolved in a dynamic way
    - For ScaleIn (analysis of resources consumed, FIFO, FILO, etc..)
    - For Traffic Distribution – proportional distribution among modules, equal, only selected modules
- **HOW SO should determine VNFC Instance in vf-module for LCM operation**
  - No problem when there is only one VNFC instance in vf-module
  - When there is more than one:
    - For closed loop event allow OOF using policy or workflow input context to derive the VNFC LCM operation execution.

SO VNFC LCM Orchestration GAP

- SO should determine common identifiers based on scope for network function such as vnf-id, vf-module-id, vserver-id, vnfc-name
- SO should pass input params for which are passed to workflow context
- Orchestrator should not be responsible for determining provisioning configuration parameters for each action within a scope such as IP address, port, etc.. it should be internal to the controller
- Device communication parameters should be common and should not be model specific – it simplifies controller – the best source of this data is AAI and/or MDSAL i.e. for IP addresses in AAI we have
  - ipv4-oam-address – VNF level
  - ipaddress-v4-oam – PNF level
  - **ipaddress-v4-oam-vip - VNFC level – the best source for VNFC OAM**

Determination of key identifers dynamically with OOF

1. vf-module-id can come from the SO request input as well as vnfc-type
2. Today vf-module-id can be determined by OOF
   - OOF HAS (Homing, Allocation, Selection) since Dublin can resolve vf-module-id
   - Determined by standard HAS policies which in our case could be service, vnf, vnfc-type or action specific.
3. OOF not needed when
   - Only one vf-module with specified vnfc-type exists in VNF
   - vf-module-id comes from the workflow input or is determined by the workflow (i.e. as a result of ScaleOut operation)


```
{
    "requestInfo": {
        "transactionId": "e576c75e-7536-4145-a1c0-d60b65bb1bb8",
        "requestId": "de4f04e3-0a65-470b-9d07-8ea6c2fb3e10",
        "callbackUrl": "http://127.0.0.1:9000/osdfCallback/",
        "sourceId": "SO",
        "requestType": "create",
        "numSolutions": 100,
        "optimizers": ["placement"],
        "timeout": 1200
    },
    "placementInfo": {
        "requestParameters": {
            "chosenRegion": "RegionOne",
            "chosenCustomerId": "DemoCust_4fb7d3cf-5ddc-4d8c-8acf-70cc9174d18f"
        },
        "subscriberInfo": {
            "globalSubscriberId": "dbc2c763-6383-42d6-880a-b7d5c5bc84d9",
            "subscriberName": "oof-so-chm"
        },
        "placementDemands": [
            {
                "resourceModuleName": "vFWDTvFW",
                "serviceResourceId": "vFWDTvFW",
                "resourceModelInfo": {
                    "modelInvariantId": "6f3fd439-fd5f-4a2d-95bc-b6bf8787001a",
                    "modelVersionId": "202d2fd8-a045-4c9a-b767-2a1639c10291"
                },
                "excludedCandidates": [{
                        "identifierType": "vfmodule",
                        "identifiers": []}],
                "requiredCandidates": [{
                        "identifierType": "vfmodule",
                        "identifiers": []}]
            },
            {
                "resourceModuleName": "vFWDTvPGN",
                "serviceResourceId": "vFWDTvPGN",
                "unique": "false",
                "resourceModelInfo": {
                    "modelInvariantId": "3f356335-7b36-41ee-8f74-72d0a2ec3ebf",
                    "modelVersionId": "6bfe954e-bb00-4111-be3c-33eed9d20a8c"
                }
            }
        ]
    },
    "serviceInfo": {
        "serviceInstanceId": "2ad369d4-9056-4dc9-8e6d-df24f45e8729",
        "serviceName": "vFW_TD",
        "modelInfo": {
            "modelInvariantId": "TD-invariantId",
            "modelVersionId": "TD-versionId"
        }
    }
}
```

VNFC support for CDS

SO CDS Action Execution Request Example (1)

```
{
    "commonHeader":{
        "subRequestId":"{generated_by_service orchestrator (SO)}",
        "requestId":"{req_id_from_DCAE}",
        "originatorId":"SERVICE ORCHESTRATOR (SO)"
    },
    "actionIdentifiers":{
        "mode":"sync",
        "blueprintName":"{blueprint_name_from_operational_service orchestrator (SO)_config}",
        "blueprintVersion":"{blueprint_version_from_operational_service orchestrator (SO)_config}",
        "actionName":"{blueprint_action_name_from_operational_service orchestrator (SO)_config}"
    },
    "payload":{
        "$actionName-request":{
            "resolution-key":"{generated_by_service orchestrator (SO)}",
            "$actionName-properties":{
                "$attribute_1":"",
                "$attribute_2":""
            }
        }
    }
}
```

- actionIdentifers - determined by SO and executed workflow
- $actionName-properties – inserted by SO base on action scope and workflow input
  - AAI enriched attributes i.e. identifiers of VNF, vf-module, OAM IP address
  - Close Loop event information
  - Static information
- Each action can also have dedicated step(s) for assignment of extra attributes with custom properties used i.e. in the Kotlin script implementing the action
  - Cross vf-module or vnfc dependency i.e. IP address from one vf-module used in reconfiguration of the other one

SO CDS Action Execution Request Example (2)

scope: pnf action: reconfigure-pnf


```
{
    "commonHeader":{
        "subRequestId":"14384b21-8224-4055-bb9b-0469397db801",
        "requestId":"d57709fb-bbec-491d-a2a6-8a25c8097ee8",
        "originatorId":"SERVICE ORCHESTRATOR (SO)"
    },
    "actionIdentifiers":{
        "mode":"sync",
        "blueprintName":"PNF-demo",
        "blueprintVersion":"1.0.0",
        "actionName":"reconfigure-pnf"
    },
    "payload":{
        "reconfigure-pnf-request":{
            "resolution-key":"8338b828-51ad-4e7c-ac8b-08d6978892e2",
            "reconfigure-pnf-properties":{
                "pnf.equip-vendor":"Vendor-A",
                "pnf.ipaddress-v4-oam":"10.10.10.10",
                "pnf.in-maint":false,
                "pnf.pnf-ipv4-address":"3.3.3.3",
                "pnf.resource-version":"1570746989505",
                "pnf.nf-role":"ToR DC101",
                "pnf.equip-type":"Router",
                "pnf.equip-model":"model-123456",
                "pnf.frame-id":"3",
                "pnf.pnf-name":"demo-pnf",
                "data": "peer-as=64577",
                "peer-group":"demo-peer-group",
                "neighbor-address":"4.4.4.4"
            }
        }
    }
}
```


scope: vnf action: config-deploy


```
{
    "commonHeader":{
        "subRequestId":"14384b21-8224-4055-bb9b-0469397db801",
        "requestId":"d57709fb-bbec-491d-a2a6-8a25c8097ee8",
        "originatorId":"SERVICE ORCHESTRATOR (SO)"
    },
    "actionIdentifiers":{
        "mode":"sync",
        "blueprintName":"vFW-CDS",
        "blueprintVersion":"1.0.0",
        "actionName":"config-deploy"
    },
    "payload":{
        "config-deploy-request":{
            "resolution-key":"6128eb53-0eac-4c79-855c-ff56a7b81141",
            "config-deploy-properties":{
                "service-instance.service-instance-id":"40004db6-c51f-45b0-aba
                "generic-vnf.vnf-id":"8d09e3bd-ae1d-4765-b26e-4a45f568a092",
                "data":{
                    "active-streams":"7"
                }
            }
        }
    }
}
```

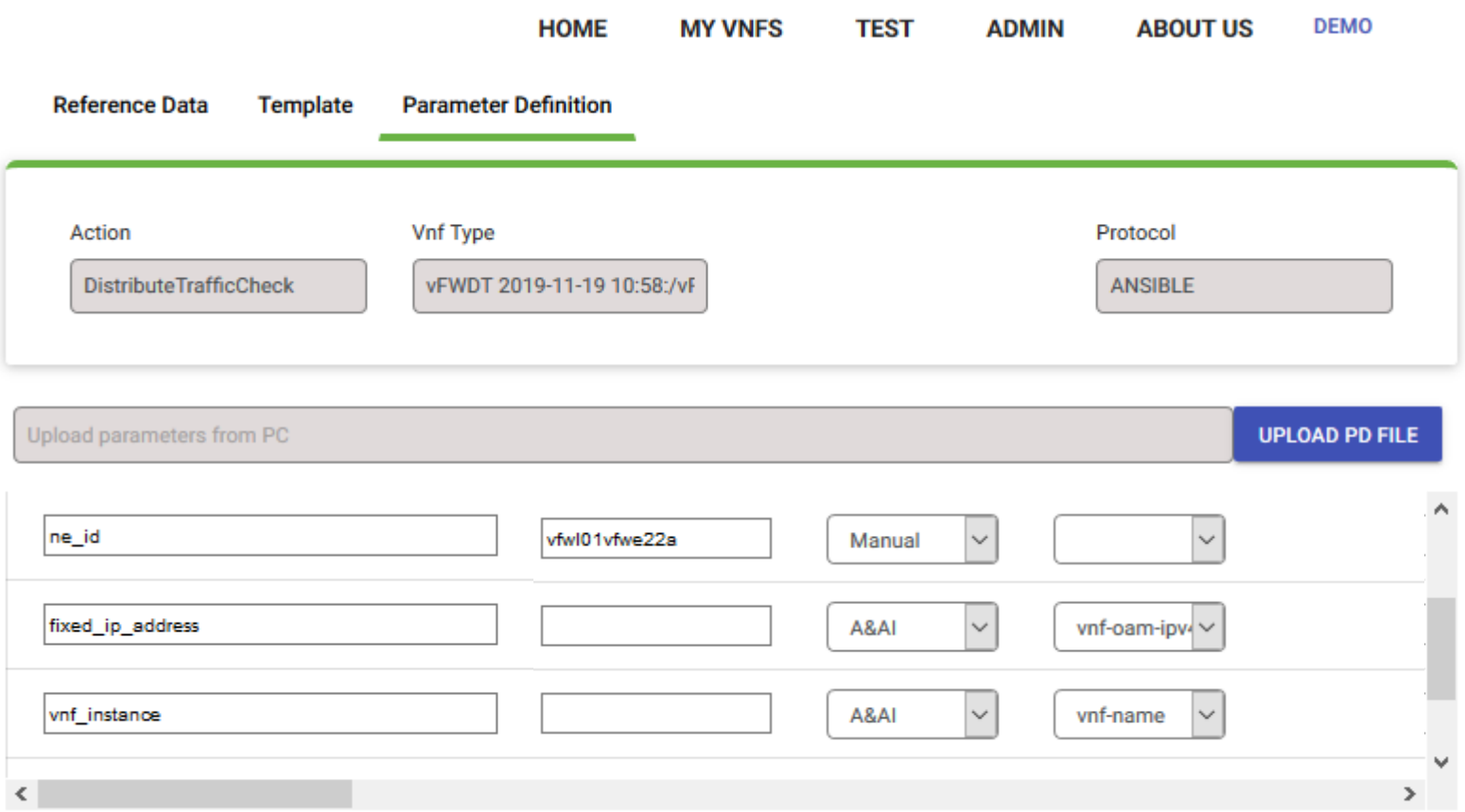

For vnfc support just new scope must be introduced on SO side which will generate missing identifiers in the action properties part of the payload. Rest is implementation of action on blueprint side

VNFC support for APPC

APPC – Existing AAI integration for VNFC recognition

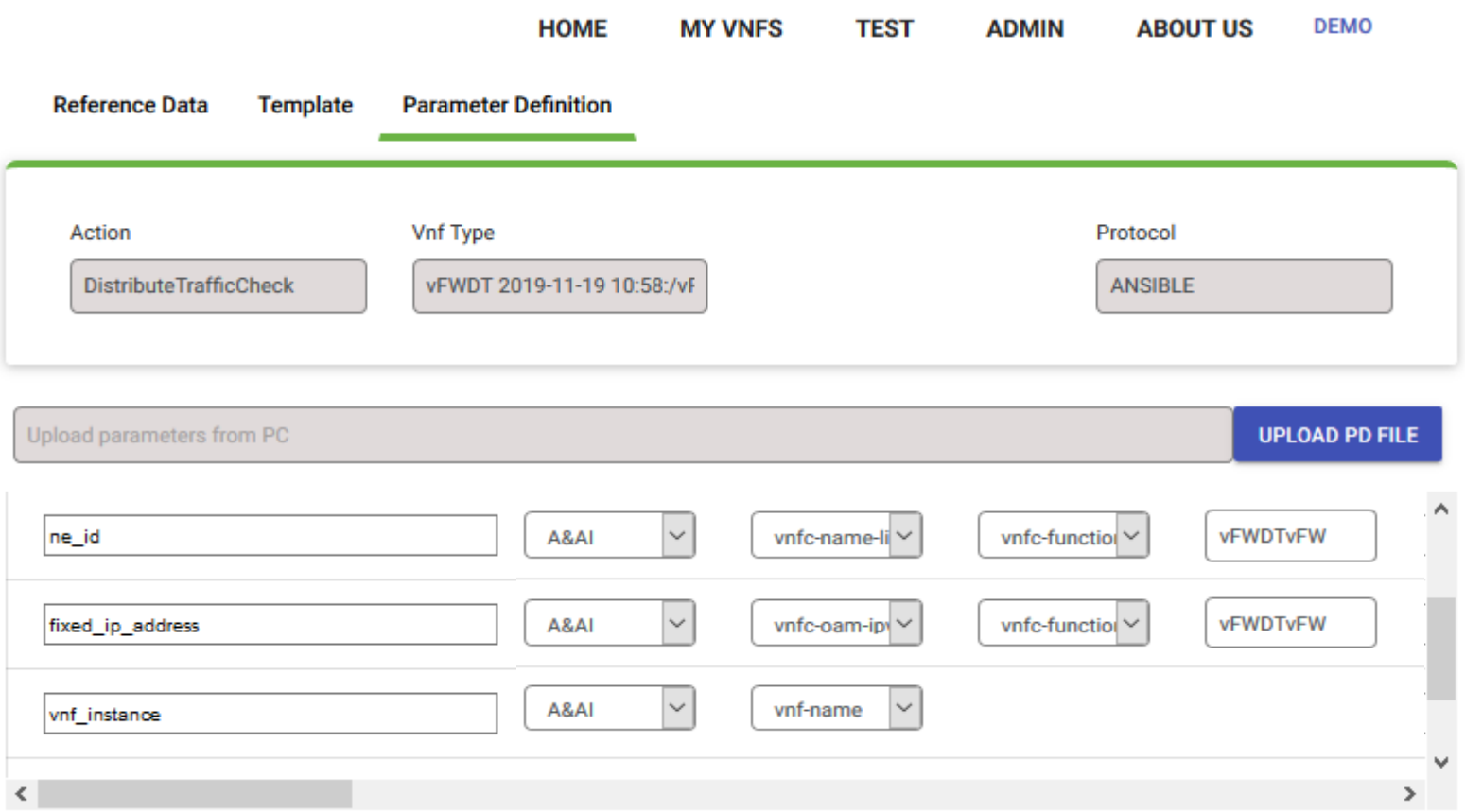
2019-11-20T14:18:05,498 | AAIResourceNode | Populating Final Context
2019-11-20T14:18:05,499 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vnfc-count Value = 1
2019-11-20T14:18:05,499 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vnfc-name Value = vfwl01pgne22a
2019-11-20T14:18:05,499 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].cloud-region-id Value = RegionOne
2019-11-20T14:18:05,499 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vserver-selflink Value = http://192.168.186.11:8774/v2.1/1f6284684eb44ae79a0a5677e
2019-11-20T14:18:05,500 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].tenant-id Value = 1f6284684eb44ae79a0a5677e12da7eb
2019-11-20T14:18:05,500 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].group-notation Value = null
2019-11-20T14:18:05,500 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].cloud-owner Value = CloudOwner
2019-11-20T14:18:05,500 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vserver-name Value = vfwl01pgne22a
2019-11-20T14:18:05,500 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vnfc-ipaddress-v4-oam-vip Value = null
2019-11-20T14:18:05,500 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vf-module-id Value = b8c1d740-164d-4ca4-9314-a7406041cde9
2019-11-20T14:18:05,500 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vnfc-type Value = vFWDTvPKG
2019-11-20T14:18:05,501 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vnfc-function-code Value = vFWDTvPKG
2019-11-20T14:18:05,501 | AAIResourceNode | Populating Context Key = tmp.vnfInfo.vm[0].vserver-id Value = 8ed45a28-ae08-4f75-824c-d8db78af1c2c
2019-11-20T14:18:05,501 | AAIResourceNode | VNFCNAME 0vfwl01pgne22a
2019-11-20T14:18:05,501 | AAIResourceNode | VMCOUNT IN GETALLVSERVERS 1
2019-11-20T14:18:05,501 | AAIResourceNode | VMSWITHNOVNFCSCOUNT IN GETALLVSERVERS 0
2019-11-20T14:18:05,502 | AAIResourceNode | VMSWITHNOVNFCSCOUNTFOR VFMODULE IN GETALLVSERVERS 0
2019-11-20T14:18:05,502 | AAIResourceNode | VMCOUNT FOR VFMODULE IN GETALLVSERVERS 0

CDT Template Definition for VNFC - Option 1 - Available



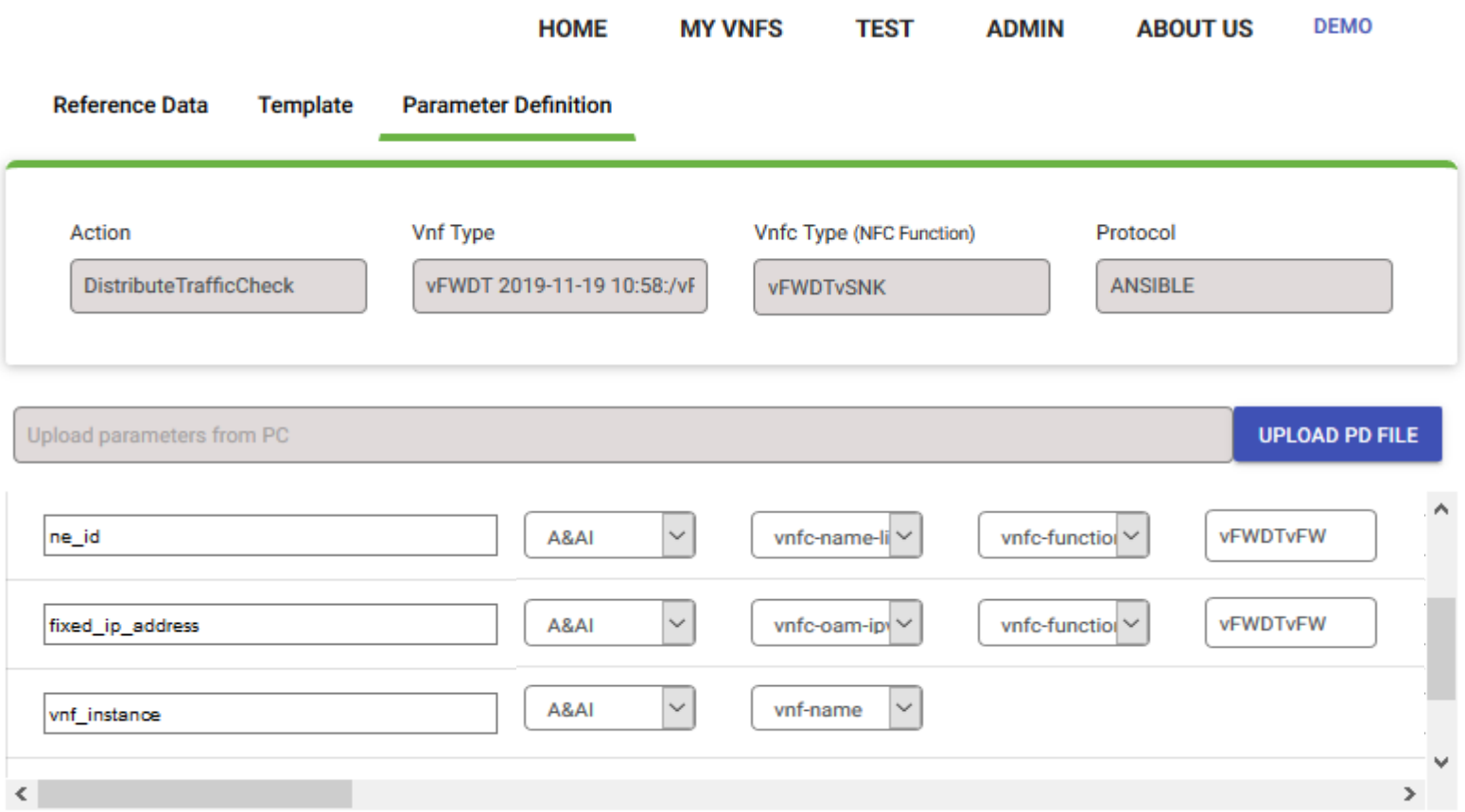
- ne_id – fixed value
- fixed_ip_address – APPC resolves from AAI -> generic-VNF -> ipv4-oam-address
- action-identifiers: vnf-id
- Can be applied for:
  - UC #1
  - UC #2 when only one VNFC can be reconfigurable
  - UC #2, 3, 4 when one VNFC acts a role of VNFC controller in VNF – ne_id ipv4-oam-address must point this controller
  - UC #2, 3, 4 when VNF has external VNFC controller (VNF Manager) – ne_id ipv4-oam-address must point this controller

CDT Template Definition for VNFC - Option 2 - Available



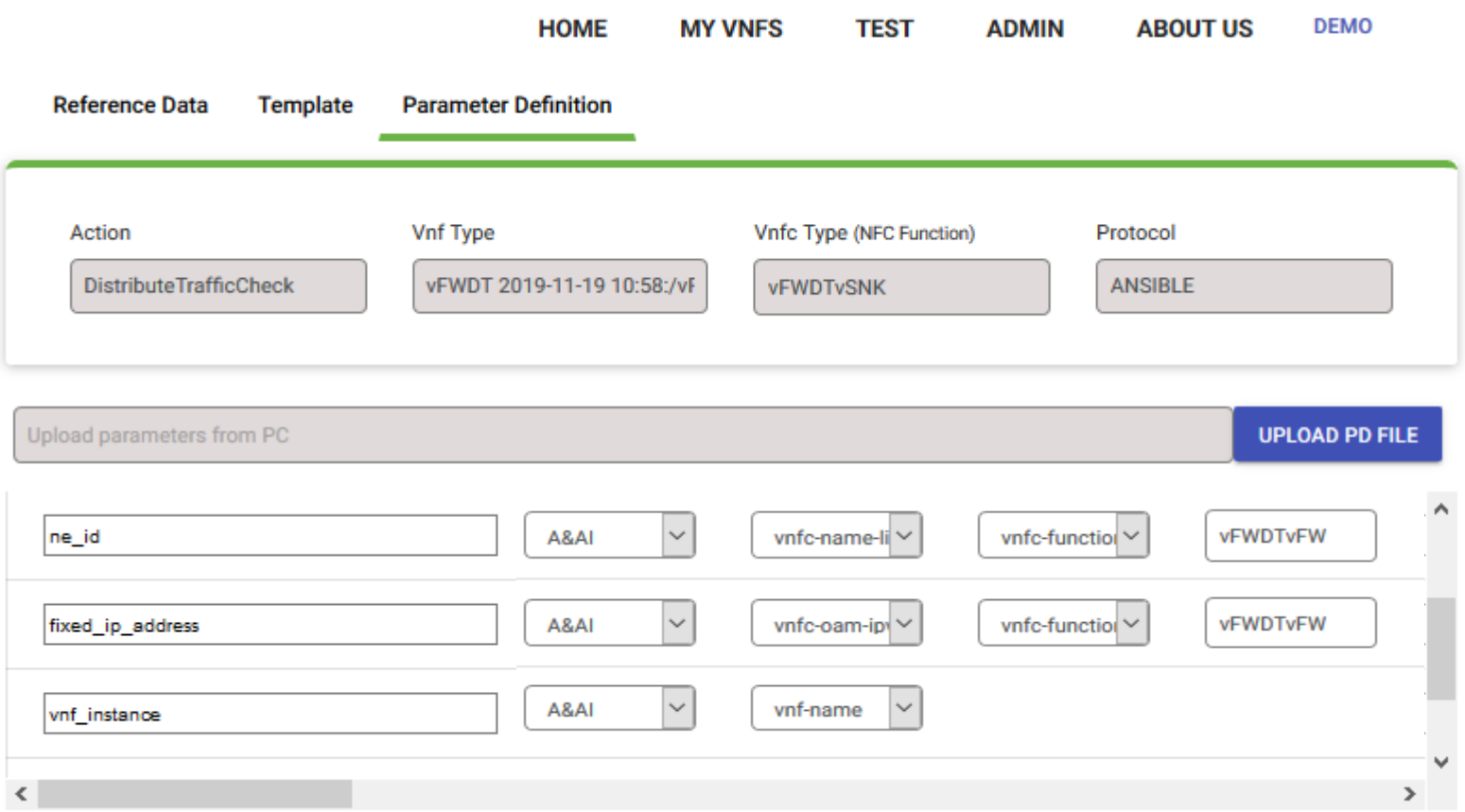
- Requires vnfc info in AAI
- ne_id – APPC resolves from AAI -> VNFC -> vnfc-name
- fixed_ip_address – APPC resolves from AAT -> VNFC -> vnfc-ipaddress-v4-oam-vip
- action-identifiers: vnf-id
- VNFC type selected by nfc-function type
- Can be applied for:
  - UC #1
  - UC #2 – when each action type executed on different vnfc-type
  - UC #4 like UC #2 + only one instance of VNFC type in VNF

CDT Template Definition for VNFC - Option 3 - Available



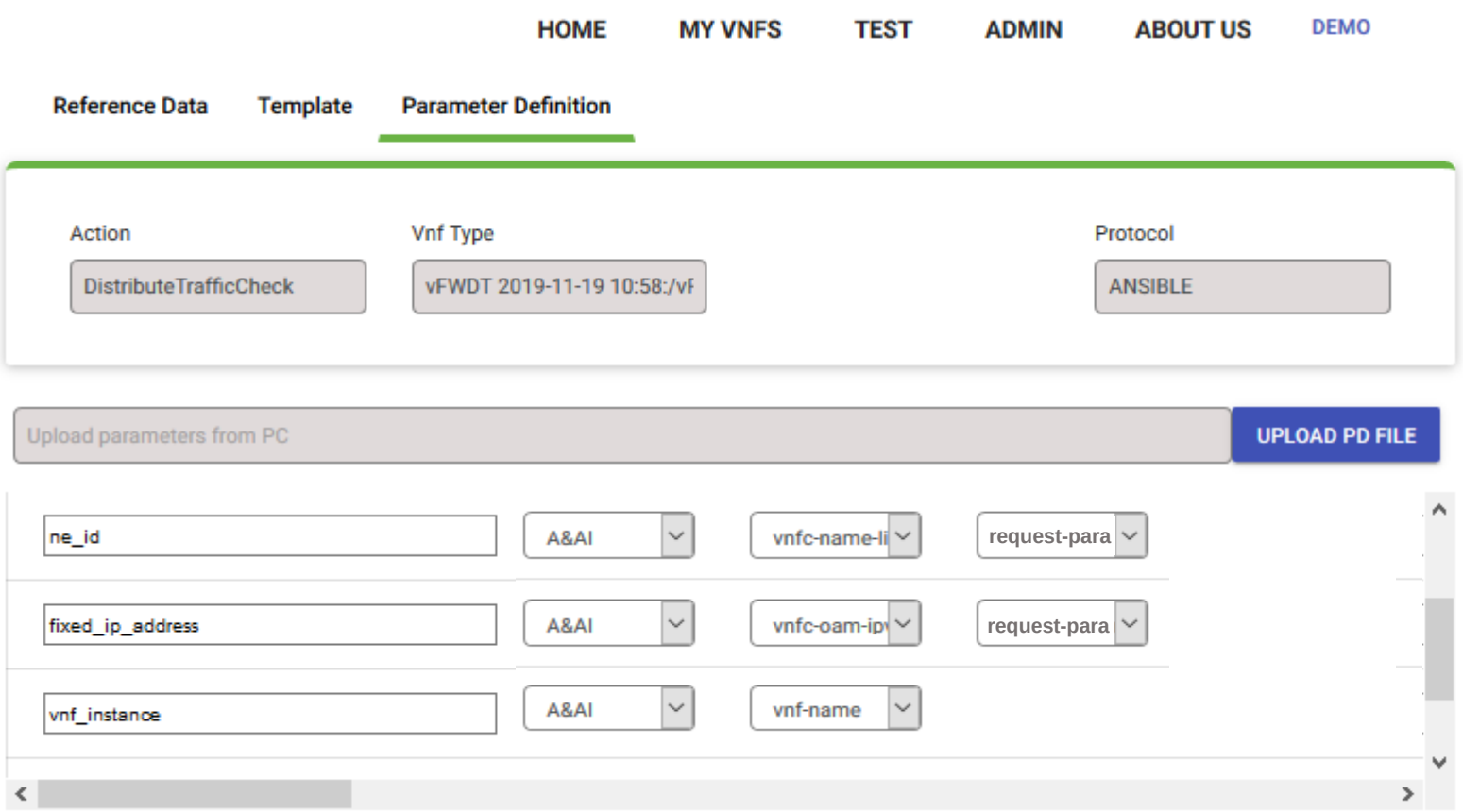
- Requires vnfc info in AAI
- ne_id – APPC resolves from AAI -> VNFC -> vnfc-name
- fixed_ip_address – APPC resolves from AAI -> VNFC -> vnfc-ipaddress-v4-oam-vip
- action-identifiers: vnf-id
- VNFC type selected by vnfc-function type
- Requires APPC to have many templates for one VNF selected by vnfc type – today we cannot
- Can be applied for:
  - UC #1
  - UC #2
  - UC #4 only one instance of VNFC type in VNF

CDT Template Definition for VNFC - Option 4 - Proposal



- Requires vnfc info in AAI
- ne_id – APPC resolves from AAI -> VNFC -> vnfc-name
- fixed_ip_address – APPC resolves from AAI -> VNFC -> vnfc-ipaddress-v4-oam-vip
- action-identifiers: vnf-id, vnfc-name
- VNFC type selected by nfc-function type
- Requires existing (but disabled) mechanisms of VNFC support to be enabled in APPC
- SO finds VNFC instance (vnfc-name)
- Can be applied for:
  - UC #1
  - UC #2
  - UC #3
  - UC #4

CDT Template Definition for VNFC - Option 5 - Proposal



- Requires vnfc info in AAI
- ne_id – APPC resolves from AAI -> VNFC -> vnfc-name
- fixed_ip_address – APPC resolves from AAI -> VNFC -> vnfc-ipaddress-v4-oam-vip
- action-identifiers: vnf-id
- filtering by request-parameters: vf-module-id, vnfc-type
- SO determines vf-module-id and vnfc-type. Most likely vnf-type can come from the input
- Can be applied for:
  - UC #1
  - UC #2
  - UC #3
  - UC #4

The best option that allows to configure concrete VM for vnf, vf-module and vnfc scope

Summary