Understanding Virtual Network Function Descriptors

Virtual Network Function Descriptor Overview

ESC supports a TOSCA-based Virtual Network Function Descriptor (VNFD) to describe the VNF properties. The VNFD conforms to the *GS NFV-SOL 001* specifications and standards specified by ETSI.

The VNFD file describes the instantiation parameters and operational behaviors of the VNFs. It contains KPIs, and other key requirements that can be used in the process of onboarding and managing the lifecycle of a VNF.

For VNF Lifecycle operations, see VNF Lifecycle Operations Using ETSI API.

Creating Extensions to the Virtual Network Function Descriptor

ESC implements extensions to the VNFD defined by Cisco to expose the more advanced concepts supported by ESC, but missing in the ETSI standards. These extensions are strongly typed in the Cisco types definition to describe the overridden data, node, and interface types.

VNF Configurable Properties

The VNF node type is always customized for each VNF. The Cisco extensions allow the ability to specify the recovery policy and time to wait for the VNF to recover before ESC considers any mitigating action.

For example:

```
vnf:
      type: cisco.VPC.1_0.1_0
      properties:
        descriptor_id: b98450dd-f532-4a42-8419-e3dc04327318
        descriptor_version: '3.8'
        provider: cisco
        product_name: VPC
        software_version: 1.0
        product_info_name: 'Virtual Packet Core (VPC); 32 vCPUs, 64Gb RAM, 66Gb vStorage'
        vnfm_info:
          - '9:Cisco Elastic Services Controller:v04.04.01'
```

```
    configurable_properties:
      is_autoscale_enabled: false
      is_autoheal_enabled: false
    lcm_operations_configuration:
      heal:
        recovery_action: REBOOT_THEN_REDEPLOY
        recovery_wait_time: 0
    flavour_id: default
    flavour_description: 'Default VNF Deployment Flavour'
```

Compute

The Cisco Compute node allows for many of the ESC features to be exposed via the extended ETSI data model. This includes the following:

- Overriding the automatically generated name for a VNFC on the VIM.
- VIM flavor (overriding the ETSI capabilities specified for a VNFC).
- Supplying ESC with an expected bootup time to prevent further actions being taken until this timer has expired.
- Providing Day-0 configuration blocks to execute/store on the VNFC once deployed.
- Specifying KPI parameters and associated rules to configure the monitoring agent.
- Intra-VM Group Placement rules.

For example:

```
vdu1:
  type: cisco.nodes.nfv.Vdu.Compute
  properties:
    name: Example VDU1
    description: Example VDU
    boot_order:
      - boot1-volume
    configurable_properties:
      additional_vnfc_configurable_properties:
        vim_flavor: Automation-Cirros-Flavor
        bootup_time: 1800
    name_override: my-vdu-1
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 1
      static_ip_address_pool:
        network: esc-net
        ip_address_range:
          start: { get_input: VDU1_NETWORK_START }
          end: { get_input: VDU1_NETWORK_END }
        ip_addresses: { get_input: VDU1_SCALE_IP_LIST }
    kpi_data:
      VM_ALIVE-1:
       event_name: 'VM_ALIVE-1'
        metric_value: 1
        metric_cond: 'GT'
        metric_type: 'UINT32'
        metric_occurrences_true: 1
        metric_occurrences_false: 30
        metric_collector:
          type: 'ICMPPing'
          nicid: 1
          poll_frequency: 10
          polling_unit: 'seconds'
```

```
        continuous_alarm: false
    admin_rules:
      VM_ALIVE-1:
        event_name: 'VM_ALIVE-1'
        action:
          - 'ALWAYS log'
          - 'FALSE recover autohealing'
          - 'TRUE esc_vm_alive_notification'
    placement_type: zone
    placement_target: nova
    placement_enforcement: strict
    vendor_section:
      cisco_esc:
        config_data:
          example.txt:
            file: ../Files/Scripts/example.txt
            variables:
              DOMAIN_NAME: { get_input: DOMAIN_NAME }
              NAME_SERVER: { get_input: NAME_SERVER }
              VIP_ADDR: { get_input: VIP_ADDR }
              VIP_PREFIX: { get_input: VIP_PREFIX }
  capabilities:
    virtual_compute:
      properties:
        virtual_cpu:
          num_virtual_cpu: 8
        virtual_memory:
          virtual_mem_size: 16
  requirements:
    - virtual_storage:  cdr1-volume
    - virtual_storage:  boot1-volume
```

Note You can supply a high number of input parameters, allowing the use of a single template for multiple deployments.

Connection Point

The Cisco extensions to the VduCp node type mainly allows for improved IP addressing capabilities and accessibility to the interface. The features added the connection point are as follows:

- Overriding the automatically generated name for a port on the VIM
- Static IP Addresses (and pools for scaling)
- Identification of whether the port is a management port (i.e. used for monitoring)
- Allowed Address Pairs
- Support for specific network card types and interface types, e.g. SR-IOV
- Whether port security is enabled

For example:

```
vdu1_nic0:
      type: cisco.nodes.nfv.VduCp
      properties:
        layer_protocols: [ ipv6 ]
        protocol:
          - associated_layer_protocol: ipv6
```

```
      trunk_mode: false
      order: 0
      nw_card_model: virtio
      iface_type: direct
      management: true
      name_override: my-vdu1-nic0
      ip_subnet:
        - ip_address: { get_input: VDU1_NIC0_IP }
      allowed_address_pairs:
        - ip_address: { get_input: VDU1_NIC0_AADR_PAIRS }
      port_security_enabled: false
    requirements:
      - virtual_binding: vdu1
```

Volume

The ESC supports VirtualBlockStorage volume type. Instead of the addition of extra properties, ESC gives the ability to override the volume specified in the VNFD and supplying its own persistent (deployed out-of-band) storage by identifying it with a UUID from the VIM.

For example:

```
boot1-volume:
  type: cisco.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    resource_id: { get_input: VDU1_BOOT_VOL_UUID }
    virtual_block_storage_data:
      size_of_storage: 4GB
      vdu_storage_requirements:
        vol_id: 1
        bus: ide
        type: LUKS
    sw_image_data:
      name: 'Automation_Cirros'
      version: '1.0'
      checksum: 9af30fce37a4c5c831e095745744d6d2
      container_format: bare
      disk_format: qcow2
      min_disk: 2 GB
      size: 2 GB
  artifacts:
    sw_image:
      type: tosca.artifacts.nfv.SwImage
      file: ../Files/Images/Automation-Cirros.qcow2
```

Security Group Rule

As per the handling of the volume above, ESC provides the ability to specify an out-of-band security group instead of configuring one in the VNFD. This is because the verbs used to describe the security group in the standards documentation are too simplistic for a very complicated configuration.

For example:

```
- NETWORK_ORCH_SEC_GRP_1:
    type: cisco.policies.nfv.SecurityGroupRule
    group_name: { get_input: VIM_NETWORK_ORCH_SEC_GRP_1 }
    targets: [ vdu1_nic0 ]
```