SolCon User Guide

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

SolCon User Guide

Introduction

This document outlines how to use the SolCon tool to convert TOSCA YAML SOL001 files to JSON SOL006 files to be able to load merge into NCS in Rel3 with the SOL6 VNFD model.

It is important to note that the expectation for this tool is that it will do 80-90% of the work, and the last 10% be manually completed, as there are some things the converter cannot handle. See the Limitations section for more detail.

Installing the SolCon Converter

You can install the converter in any of the following ways:

1. Install the compiled binary on your system and run the converter.
2. Install and run the converter from the source code.

The following sections provide detailed descriptions. This documentation uses solcon-OSX-0.6 download version to provide examples. Be sure to replace `-OSX-0.6` in the commands listed in this documentation with your appropriate operating system and version for the converter tool.

Installing Compiled Binary

1. Download the most recent version of SolCon for your supported operating system from here.
   If you are not running OSX, Ubuntu, or CentOS see Installing and Running from Source.
2. Make sure you are able to run the executable file to confirm installation.
   `chmod +x solcon-OSX-0.6`
3. Ensure the input and configuration files are present on your machine.
   - Input files are TOSCA specified YAML files, such as `altiostar_vCU.yaml`.

- Configuration files are in TOML format.
  - TOSCA configuration for ESC VNFDs: config-esc.toml
  - SOL6 output configuration: config-sol6.toml

Note: `config-esc.toml` determines how the YAML VNFD is read and `config-sol6.toml` specifies how the JSON SOL6 VNFD is output.

These are default files and do not need to be modified.

4 Run the compiled program from the terminal:

```
./solcon-OSX-0.6 -f altiostar_vCU.yaml -o output_altiostar.json -c
config-esc.toml -s config-sol6.toml
```

The argument `-o output_altiostar.json` determines the location and name of the JSON file.

If the program fails to run, download and run the package from the source code. See Installing and Running from Source for more information.

5 Load merge the JSON file into NCS.

Instead of entering NCS, run the following terminal command.

```
ncs_load -lm -F o altiostar_vCU.json
```

- -lm stands for load merge
- -F is the format
- o is the flag for JSON format

If the file to be load merged is not properly formatted, or is missing some required fields, the ncs_load program displays an error. See Troubleshooting and Diagnostics for more information.

Installing and Running from Source

You can install and run SolCon from the source code if you are unable to install and run from the compiled binary, or if you wish to make changes on your own.

1 Download the source code from the repository. See Sources.

2 Download and install Python3 based on your operating system.

3 Once python3 is installed, run the following command to install the PyYAML and toml packages
`sh tools/setup-script.sh`

4 Run the following command to set up the `PYTHONPATH` variable

```
PYTHONPATH=python/nfvo_solcon_tosca
```

Running the command without a bash script requires that with every unique terminal instance, you must run the command from inside the repository directory. However, you can automate this command to set up the variable and run the program simultaneously. See Automating PYTHONPATH for more information.

5 Run the SolCon program.

```
python3 solcon.py -f altiostar_vCU.yaml -o output_altiostar.json -c
config-esc.toml -s config-sol6.toml
```

6 Load merge the JSON file into NCS.

Instead of entering NCS, run the following terminal command.

```
ncs_load -lm -F o altiostar_vCU.json
```

- -lm stands for load merge
- -F is the format

- o is the flag for JSON format

If the file to be load merged is not properly formatted, or is missing some required fields, the ncs_load program displays an error. See Troubleshooting and Diagnostics for more information.

Troubleshooting and Diagnostics

WARNINGs and ERRORs

SolCon makes liberal use of `WARNING` log messages, for example:

```
NFO - Starting Cisco TOSCA -> SOL6 converter.
WARNING - parameters not found in topology_template;node_templates;vnf;interfaces;Vnflcm;insta
WARNING - type not found in topology_template;node_templates;vBBU1_vol;properties;virtual_blo
```

These kinds of warnings are semi-expected, and they do not preclude the converter from generating an output file. Specifically they occur when a field is expected in the TOSCA file, but is not found. Unless it is an extremly important field, the program just prints the warning message and continues converting what it can.

The converter will output an `ERROR` message when something critical has gone wrong. An output file will not be generated when an error occurs.

Log-level DEBUG

Run the SolCon converter program with the log-level set to debug

```
./solcon-OSX-0.6 (...) --log-level DEBUG
```

This helps to capture more log information than running the program in the info mode. Use the log information stored in the `logs/` folder to troubleshoot and diagnose any problems that you may encounter with the tool.

Ensure types match

Ensure the types in `config-esc.toml` match the types in the TOSCA VNFD.

For example, given the following definitions in a TOSCA file:

```
c1:
    type: cisco.nodes.nfv.Vdu.Compute

vnfd1-deployment-control-function-1-cf-boot:
    type: cisco.nodes.nfv.Vdu.VirtualBlockStorage

c1_nic0:
    type: cisco.nodes.nfv.VduCp
```

The `config-esc.toml` configuration file must look like this:

```
[provider_identifiers.cisco]
    vdu                     = ["type", "cisco.nodes.nfv.Vdu.Compute"]
    int_cpd                 = ["type", "cisco.nodes.nfv.VduCp"]
    int_cpd_mgmt            = ["type", "cisco.nodes.nfv.VduCp"]
    virtual_storage         = ["type", "cisco.nodes.nfv.Vdu.VirtualBlockStorage"]
```

There are more type definitions in the default files. If any of the types are not set correctly in the configuration, they will not be found by the converter and thus will be skipped.

Limitations

- Any data not present in the TOSCA file will not be able to be generated for the SOL6 model.
- Internal connection points and external connection points must have unique names.

Automating PYTHONPATH

Create a bash script as follows to set the PYTHONPATH and run the program simultaneously.

```
#!/usr/bin/env bash
export PYTHONPATH=python/nfvo_solcon_tosca

python3 solcon.py -f altiostar_vCU.yaml -o output_altiostar.json
    -c config-esc.toml -s config-sol6.toml
```

This sets the PYTHONPATH and runs the program at the same time.

CHAPTER 2

Appendix

Sources

If you are installing from the source code, download it from Here.

You can make modifications to the source code (check the license for specifics.)

TOML Configuration Files

config-esc.toml

```
# These must match with 'provider-identifiers.{}'
# If the given provider for a VNFD matches with one of these, then it will automatically
# take the identifiers from this file
providers=["cisco"]

version = "0.1.0"

# All of the identifiers must be the same for all instances of that object,
# multiple IDs for a single type are not supported
[provider_identifiers.cisco]
    vdu                     = ["type", "cisco.nodes.nfv.Vdu.Compute"]
    int_cpd                 = ["type", "cisco.nodes.nfv.VduCp"]
    int_cpd_mgmt            = ["type", "cisco.nodes.nfv.VduCp"]
    instantiation_level     = ["type", "tosca.policies.nfv.VduInstantiationLevels"]
    scaling_aspects         = ["type", "tosca.policies.nfv.ScalingAspects"]
    scaling_aspects_deltas   = ["type", "tosca.policies.nfv.VduScalingAspectDeltas"]
    virtual_storage         = ["type", "cisco.nodes.nfv.Vdu.VirtualBlockStorage"]
    security_group          = ["type", "cisco.policies.nfv.SecurityGroupRule"]
    anti_affinity_rule      = ["type", "tosca.policies.nfv.AntiAffinityRule"]
    affinity_rule           = ["type", "tosca.policies.nfv.AffinityRule"]
    placement_group         = ["type", "tosca.groups.nfv.PlacementGroup"]

# Note: If there is a variable with "path_VAL", that means it will not be parsed for the path
# heirarchy, but will instead just be set with the value
# The structure of the TOSCA file, in paths
[tosca]
    topology_template       = "topology_template"
    node_templates          = ["topology_template", "node_templates"]
```

```
substitution_map        = ["topology_template", "substitution_mappings"]
substitution_req        = ["substitution_map", "requirements"]
policies                = ["topology_template", "policies"]
groups                  = ["topology_template", "groups"]
inputs                  = ["topology_template", "inputs"]
desc                    = "description"
input_key               = "get_input"

# ** VNF Metadata **
vnf                     = ["node_templates", "vnf"]
vnf_prop                = ["vnf", "properties"]
vnf_desc_id             = ["vnf_prop", "descriptor_id"]
vnf_desc_ver            = ["vnf_prop", "descriptor_version"]
vnf_provider            = ["vnf_prop", "provider"]
vnf_product_name        = ["vnf_prop", "product_name"]
vnf_software_ver        = ["vnf_prop", "software_version"]
vnf_product_info_name   = ["vnf_prop", "product_info_name"]
vnf_vnfm_info           = ["vnf_prop", "vnfm_info"]
vnf_conf_props          = ["vnf_prop", "configurable_properties"]
vnf_conf_autoheal       = ["vnf_conf_props", "is_autoheal_enabled"]
vnf_conf_autoscale      = ["vnf_conf_props", "is_autoscale_enabled"]
vnf_lcm_conf            = ["vnf_prop", "lcm_operations_configuration"]
vnf_lcm_heal            = ["vnf_lcm_conf", "heal"]
vnf_lcm_heal_item       = ["vnf_lcm_heal", "{}"]
# Additional configurable parameters
vnf_interfaces          = ["vnf", "interfaces"]
vnf_vnflcm              = ["vnf_interfaces", "Vnflcm"]
vnf_instantiate         = ["vnf_vnflcm", "instantiate"]
vnf_inst_inputs         = ["vnf_instantiate", "inputs"]
vnf_additional_param_list= ["vnf_inst_inputs", "additional_parameters"]
vnf_add_parameter       = ["vnf_additional_param_list", "parameters"]
vnf_add_param_elem      = ["vnf_add_parameter", "{}"]
# These are the variables that will be taken from parameters_list and put into the sol6 VNFD
ADD_PARAMS_VAL          = ["BOOTUP_TIME_SF", "BOOTUP_TIME_CF", "CHASSIS_KEY"]

# ** VDU **
vdu                     = ["node_templates", "{}"]
vdu_props               = ["vdu", "properties"]
vdu_name                = ["vdu_props", "name"]
vdu_boot                = ["vdu_props", "boot_order"]
vdu_desc                = ["vdu_props", "description"]
vdu_conf_props_base     = ["vdu_props", "configurable_properties"]
vdu_conf_props          = ["vdu_conf_props_base", "additional_vnfc_configurable_properties"]

vdu_vim_flavor          = ["vdu_conf_props", "vim_flavor"]
vdu_cap                 = ["vdu", "capabilities"]
vdu_cap_vc              = ["vdu_cap", "virtual_compute"]
vdu_cap_props           = ["vdu_cap_vc", "properties"]
vdu_virt_cpu            = ["vdu_cap_props", "virtual_cpu"]
vdu_virt_cpu_num        = ["vdu_virt_cpu", "num_virtual_cpu"]
vdu_virt_mem            = ["vdu_cap_props", "virtual_memory"]
vdu_virt_mem_size       = ["vdu_virt_mem", "virtual_mem_size"]

vdu_profile             = ["vdu_props", "vdu_profile"]
vdu_prof_inst_min       = ["vdu_profile", "min_number_of_instances"]
vdu_prof_inst_max       = ["vdu_profile", "max_number_of_instances"]

vdu_vendor              = ["vdu_props", "vendor_section"]
vdu_cisco_esc           = ["vdu_vendor", "cisco_esc"]
vdu_day0_list           = ["vdu_cisco_esc", "config_data"]
vdu_day0                = ["vdu_day0_list", "{}"]
vdu_day0_file           = ["vdu_day0", "file"]
```

```
vdu_day0_variables      = ["vdu_day0", "variables"]
vdu_day0_variable       = ["vdu_day0_variables", "{}"]

# ** Do not modify **
vdu_day0_custom_id      = ["vdu_day0", "custom_id"]
# ** End **

# ** Internal Connection Points **
int_cpd                 = ["node_templates", "{}"]
int_cpd_props           = ["int_cpd", "properties"]
int_cpd_req             = ["int_cpd", "requirements"]
int_cpd_virt_binding    = ["int_cpd_req", "virtual_binding"]
int_cpd_virt_link       = ["int_cpd_req", "virtual_link"]
int_cpd_layer_prot      = ["int_cpd_props", "layer_protocols"]
int_cpd_allowed_pair    = ["int_cpd_props", "allowed_address_pairs"]
int_cpd_ip_allowed_addr = ["int_cpd_allowed_pair", "ip_address"]
int_cpd_ip_addr         = ["int_cpd_props", "ip_address"]
int_cpd_vl_profile      = ["int_cpd_props", "vl_profile"]
int_cpd_virt_prot_data  = ["int_cpd_vl_profile", "virtual_link_protocol_data"]
int_cpd_l3_data         = ["int_cpd_virt_prot_data", "l3_protocol_data"]
int_cpd_cidr            = ["int_cpd_l3_data", "cidr"]
int_cpd_dhcp            = ["int_cpd_l3_data", "dhcp_enabled"]



virt_storage            = ["node_templates", "{}"]

virt_props              = ["virt_storage", "properties"]
virt_artifacts          = ["virt_storage", "artifacts"]
virt_vsb                = ["virt_props", "virtual_block_storage_data"]
virt_size               = ["virt_vsb", "size_of_storage"]
virt_storage_req        = ["virt_vsb", "vdu_storage_requirements"]
virt_type               = ["virt_storage_req", "type"]

sw_image_data           = ["virt_props", "sw_image_data"]
sw_name                 = ["sw_image_data", "name"]
sw_version              = ["sw_image_data", "version"]
sw_checksum             = ["sw_image_data", "checksum"]
sw_container_fmt        = ["sw_image_data", "container_format"]
sw_disk_fmt             = ["sw_image_data", "disk_format"]
sw_min_disk             = ["sw_image_data", "min_disk"]
sw_size                 = ["sw_image_data", "size"]
sw_image                = ["virt_artifacts", "sw_image"]
sw_image_file           = ["sw_image", "file"]

# ** Deployment Flavor **
df_id                   = ["vnf_prop", "flavour_id"]
df_desc                 = ["vnf_prop", "flavour_description"]


def_inst_level          = ["policies", "instantiation_levels"]
def_inst_key            = "default"
def_inst_prop           = ["def_inst_level", "properties"]
def_inst_p_levels       = ["def_inst_prop", "levels"]
def_inst_def            = ["def_inst_p_levels", "default"]
def_inst_desc           = ["def_inst_def", "description"] # Matches def_inst_key
inst_level              = ["policies", "{}"]
inst_level_targets      = ["inst_level", "targets"]
inst_level_props        = ["inst_level", "properties"]
inst_level_levels       = ["inst_level_props", "levels"]
inst_level_def          = ["inst_level_levels", "default"]
inst_level_num_instances = ["inst_level_def", "number_of_instances"]
```

```
    # ** Scaling Aspects **
    scaling_aspects         = ["policies", "{}"]
    scaling_props           = ["scaling_aspects", "properties"]
    scaling_aspect_item_list = ["scaling_props", "aspects"]
    scaling_aspect_item     = ["scaling_aspect_item_list", "{}"]
    scaling_aspect_name     = ["scaling_aspect_item", "name"]
    scaling_aspect_desc     = ["scaling_aspect_item", "description"]
    scaling_aspect_level    = ["scaling_aspect_item", "max_scale_level"]
    scaling_aspect_deltas   = ["scaling_aspect_item", "step_deltas"]

    # For use in the deltas definition block
    deltas_aspects          = ["policies", "{}"]
    deltas_props            = ["deltas_aspects", "properties"]
    deltas_list             = ["deltas_props", "deltas"]
    deltas_elem             = ["deltas_list", "{}"]
    deltas_num_instances    = ["deltas_elem", "number_of_instances"]
    deltas_targets          = ["deltas_aspects", "targets"]
    deltas_target           = ["deltas_targets", "{}"]

    # ** Security Groups **
    security_group          = ["policies", "{}"]
    security_group_name     = ["security_group", "group_name"]
    security_group_targets  = ["security_group", "targets"]

    # ** Affinity/Anti Groups **
    affinity_group          = ["policies", "{}"]
    affinity_group_props    = ["affinity_group", "properties"]
    affinity_group_scope    = ["affinity_group_props", "scope"]
    affinity_group_targets  = ["affinity_group", "targets"]

    placement_group         = ["groups", "{}"]
    placement_members       = ["placement_group", "members"]

[tosca.input_values]
    VIM_FLAVOR = "VIM_FLAVOR_INPUT"
```

config-sol6.toml

```
# Sol6 Path configurations
[sol6]
    # **********
    # ** VNFD **
    # **********
    vnfd                        = "vnfd"
    vnfd_id                     = ["vnfd", "id"]
    vnfd_provider               = ["vnfd", "provider"]
    vnfd_product                = ["vnfd", "product-name"]
    vnfd_software_ver           = ["vnfd", "software-version"]
    vnfd_ver                    = ["vnfd", "version"]
    vnfd_info_name              = ["vnfd", "product-info-name"]
    vnfd_info_desc              = ["vnfd", "product-info-description"]
    vnfd_vnfm_info              = ["vnfd", "vnfm-info"]

    vnfd_config_props           = ["vnfd", "configurable-properties"]
    vnfd_config_autoheal        = ["vnfd_config_props", "is-auto-heal-enabled"]
    vnfd_config_autoscale       = ["vnfd_config_props", "is-auto-scalable-enabled"]
    vnfd_config_additional      = ["vnfd_config_props", "additional-configurable-property"]
    vnfd_config_add_elem        = ["vnfd_config_additional", "{}"]
    vnfd_config_add_key         = ["vnfd_config_add_elem", "key"]
    vnfd_config_add_value       = ["vnfd_config_add_elem", "value"]
```

```
PROTOCOLS_PREFIX_VAL        = "etsi-nfv-descriptors:"
VALID_PROTOCOLS_VAL         = ["ethernet", "ipv4", "ipv6", "mpls", "odu2", "pseudo-wire"]
VALID_DISK_FORMATS_VAL      = ["qcow2", "raw", "vmdk"]
VALID_CONTAINER_FORMATS_VAL = ["aki", "ami", "ari", "bare", "docker", "ova", "ovf"]
VALID_AFF_SCOPES_VAL        = ["nfvi-node", "nfvi-pop", "zone", "zone-group"]
VALID_STORAGE_TYPES_VAL     = ["ephemeral-storage", "root-storage", "swap-storage", "cisc

# ******************************
# ** Virtual Compute Descriptor **
# ******************************
vnfd_virt_compute_desc_base = ["vnfd", "virtual-compute-desc"]
vnfd_virt_compute_desc      = ["vnfd_virt_compute_desc_base", "{}"]
vnfd_vcd_id                 = ["vnfd_virt_compute_desc", "id"]
vnfd_vcd_flavor_name        = ["vnfd_virt_compute_desc", "cisco-etsi-nfvo-sol1-vnfd-exten
vnfd_virtual_cpu            = ["vnfd_virt_compute_desc", "virtual-cpu"]
vnfd_vcd_cpu_num            = ["vnfd_virtual_cpu", "num-virtual-cpu"]
vnfd_vcd_cpu_clock          = ["vnfd_virtual_cpu", "clock"]
vnfd_vcd_cpu_arch           = ["vnfd_virtual_cpu", "cpu-architecture"]
vnfd_vcd_cpu_oversub        = ["vnfd_virtual_cpu", "oversubscription-policy"]
vnfd_vcd_vdu_cpu_req        = ["vnfd_virtual_cpu", "vdu-cpu-requirements"]
vnfd_vcd_mem                = ["vnfd_virt_compute_desc", "virtual-memory"]
vnfd_vcd_mem_size           = ["vnfd_vcd_mem", "size"]

# ******************************
# ** Virtual Storage Descriptor **
# ******************************
vnfd_virt_storage_desc_base = ["vnfd", "virtual-storage-desc"]
vnfd_virt_storage_desc      = ["vnfd_virt_storage_desc_base", "{}"]
vnfd_virt_storage_id        = ["vnfd_virt_storage_desc", "id"]
vnfd_virt_storage_type      = ["vnfd_virt_storage_desc", "type-of-storage"]
VIRT_STORAGE_DEFAULT_VAL    = "root-storage"
vnfd_virt_storage_size      = ["vnfd_virt_storage_desc", "size-of-storage"]
vnfd_virt_storage_sw_image  = ["vnfd_virt_storage_desc", "sw-image-desc"]

# **********************
# ** Deployment Flavor **
# **********************
deployment_flavor           = ["vnfd", "df"]
df_id                       = ["deployment_flavor", "id"]
df_desc                     = ["deployment_flavor", "description"]
df_inst_level_default       = ["deployment_flavor", "default-instantiation-level"]
df_vdu_profile_list         = ["deployment_flavor", "vdu-profile"]
df_vdu_profile              = ["df_vdu_profile_list", "{}"]
df_vdu_prof_id              = ["df_vdu_profile", "id"]
df_vdu_prof_inst_min        = ["df_vdu_profile", "min-number-of-instances"]
df_vdu_prof_inst_max        = ["df_vdu_profile", "max-number-of-instances"]
df_vdu_prof_aff_group_list  = ["df_vdu_profile", "affinity-or-anti-affinity-group"]
df_vdu_prof_aff_group       = ["df_vdu_prof_aff_group_list", "{}"]
df_vdu_prof_aff_group_id    = ["df_vdu_prof_aff_group", "id"]

# -- Instantiation Level
df_inst_level_base          = ["deployment_flavor", "instantiation-level"]
df_inst_level               = ["df_inst_level_base", "{}"]
df_inst_level_id            = ["df_inst_level", "id"]
df_inst_level_desc          = ["df_inst_level", "description"]
df_inst_level_vdu_level_lst = ["df_inst_level", "vdu-level"]
df_inst_level_vdu_level     = ["df_inst_level_vdu_level_lst", "{}"]
df_inst_level_vdu_vdu       = ["df_inst_level_vdu_level", "vdu-id"]
df_inst_level_vdu_num       = ["df_inst_level_vdu_level", "number-of-instances"]
# -- Scaling Info
df_inst_scaling_info_list   = ["df_inst_level", "scaling-info"]
```

```
df_inst_scaling_info        = ["df_inst_scaling_info_list", "{}"]
df_inst_scaling_aspect      = ["df_inst_scaling_info", "id"]
df_inst_scaling_level       = ["df_inst_scaling_info", "scale-level"]

df_scale_aspect_list        = ["deployment_flavor", "scaling-aspect"]
df_scale_aspect             = ["df_scale_aspect_list", "{}"]
df_scale_aspect_id          = ["df_scale_aspect", "id"]
df_scale_aspect_name        = ["df_scale_aspect", "name"]
df_scale_aspect_desc        = ["df_scale_aspect", "description"]
df_scale_aspect_max_level   = ["df_scale_aspect", "max-scale-level"]
df_scale_aspect_delta_det   = ["df_scale_aspect", "aspect-delta-details"]
df_scale_aspect_deltas_list = ["df_scale_aspect_delta_det", "deltas"]
df_scale_aspect_deltas      = ["df_scale_aspect_deltas_list", "{}"]
df_scale_aspect_deltas_id   = ["df_scale_aspect_deltas", "id"]
df_scale_aspect_vdu_delta_lst = ["df_scale_aspect_deltas", "vdu-delta"]
df_scale_aspect_vdu_delta   = ["df_scale_aspect_vdu_delta_lst", "{}"]
df_scale_aspect_vdu_id      = ["df_scale_aspect_vdu_delta", "id"]
df_scale_aspect_vdu_num     = ["df_scale_aspect_vdu_delta", "number-of-instances"]
df_scale_aspect_no_delta_VAL = "unknown"

df_affinity_group_list      = ["deployment_flavor", "affinity-or-anti-affinity-group"]
df_affinity_group           = ["df_affinity_group_list", "{}"]
df_affinity_id              = ["df_affinity_group", "id"]
df_affinity_type            = ["df_affinity_group", "type"]
df_affinity_scope           = ["df_affinity_group", "scope"]
affinity_VAL                = "affinity"
anti_affinity_VAL           = "anti-affinity"

df_lcm_config               = ["deployment_flavor", "lcm-operations-configuration"]
df_lcm_heal_config          = ["df_lcm_config", "heal-vnf-op-config"]
df_heal_param_base          = ["df_lcm_heal_config", "parameter"]
df_heal_param               = ["df_heal_param_base", "{}"]
df_heal_param_key           = ["df_heal_param", "key"]
df_heal_param_value         = ["df_heal_param", "value"]

# ****************************
# ** Virtual/External Links **
# ****************************
virt_link_desc_base         = ["vnfd", "int-virtual-link-desc"]
virt_link_desc              = ["virt_link_desc_base", "{}"]
virt_link_desc_id           = ["virt_link_desc", "id"]
virt_link_desc_desc         = ["virt_link_desc", "description"]
virt_link_desc_conn         = ["virt_link_desc", "connectivity-type"]
virt_link_desc_protocol     = ["virt_link_desc_conn", "layer-protocol"]
virt_link_desc_flow         = ["virt_link_desc_conn", "flow-pattern"]
virt_link_desc_add_params   = ["virt_link_desc", "cisco-etsi-nfvo-sol1-vnfd-extensions:addit
virt_link_desc_cidr         = ["virt_link_desc_add_params", "cidr-variable"]
virt_link_desc_dhcp         = ["virt_link_desc_add_params", "dhcp-enabled-variable"]

ext_cpd_base                = ["vnfd", "ext-cpd"]
ext_cpd                     = ["ext_cpd_base", "{}"]
ext_cpd_id                  = ["ext_cpd", "id"]
ext_cpd_protocol            = ["ext_cpd", "layer-protocol"]
ext_cpd_virt_link           = ["ext_cpd", "int-virtual-link-desc"]
ext_cpd_role                = ["ext_cpd", "role"]
ext_cpd_vdu                 = ["ext_cpd", "int-cpd"]
ext_cpd_vdu_id              = ["ext_cpd_vdu", "vdu-id"]
ext_cpd_int_cpd_id          = ["ext_cpd_vdu", "cpd"]


#ext_cpd_int_cpd             = ["ext_cpd", "int-cpd"]
#ext_cpd_icp_vdu             = ["ext_cpd_int_cpd", "vdu"]
```

```
#ext_cpd_icp_cpd            = ["ext_cpd_int_cpd", "cpd"]


# *********
# ** VDU **
# *********
vdus                        = ["vnfd", "vdu"]
vdu                         = ["vdus", "{}"]
vdu_name                    = ["vdu", "name"]
vdu_desc                    = ["vdu", "description"]
vdu_id                      = ["vdu", "id"]
vdu_boot_order_list         = ["vdu", "boot-order"]
vdu_boot_order              = ["vdu_boot_order_list", "{}"]
vdu_boot_key                = ["vdu_boot_order", "key"]
vdu_boot_value              = ["vdu_boot_order", "value"]
vdu_vc_desc_list            = ["vdu", "virtual-compute-desc"]
vdu_vc_desc                 = ["vdu_vc_desc_list", "{}"]
vdu_vs_desc_list            = ["vdu", "virtual-storage-desc"]
vdu_vs_desc                 = ["vdu_vs_desc_list", "{}"]
vdu_sw_image_desc_list      = ["vdu", "sw-image-desc"]
vdu_sw_image_desc           = ["vdu_sw_image_desc_list", "{}"]

vdu_artifact                = ["vdu", "cisco-etsi-nfvo:artifact"]

# ******************************
# ** Internal Connection Points **
# ******************************
int_cpd_list                = ["vdu", "int-cpd"]
int_cpd                     = ["int_cpd_list", "{}"]
int_cpd_id                  = ["int_cpd", "id"]
int_cpd_layer_prot          = ["int_cpd", "layer-protocol"]
int_cpd_virt_link_desc      = ["int_cpd", "int-virtual-link-desc"]
int_cpd_role                = ["int_cpd", "role"]
int_cpd_interface_id        = ["int_cpd", "cisco-etsi-nfvo:interface-id"]
int_cpd_management          = ["int_cpd", "cisco-etsi-nfvo:management"]
int_cpd_management_VAL      = "[null]"
int_cpd_additional_params   = ["int_cpd", "cisco-etsi-nfvo-sol1-vnfd-extensions:additiona
int_cpd_allowed_addr        = ["int_cpd_additional_params", "allowed-address-variable"]
int_cpd_ip_addr             = ["int_cpd_additional_params", "ip-address-variable"]
int_cpd_security            = ["int_cpd_additional_params", "security-group-variable"]

KEY_VIRT_LINK_MGMT_VAL      = "VIM_NETWORK_MANAGEMENT-VL"
KEY_VIRT_LINK_MGMT_PROT_VAL = "etsi-nfv-descriptors:ipv4"
KEY_VIRT_LINK_ORCH_VAL      = "VIM_NETWORK_ORCHESTRATION-VL"
KEY_VIRT_LINK_ORCH_PROT_VAL = "etsi-nfv-descriptors:ipv4"
KEY_EXT_CP_MGMT_VAL         = "VIM_NETWORK_MANAGEMENT"
KEY_EXT_CP_MGMT_PROT_VAL    = "etsi-nfv-descriptors:ipv4"
KEY_EXT_CP_ORCH_VAL         = "VIM_NETWORK_ORCHESTRATION"
KEY_EXT_CP_ORCH_PROT_VAL    = "etsi-nfv-descriptors:ipv4"

# ******************************
# ** Software Image Descriptor **
# ******************************
sw_img_desc_base            = ["vnfd", "sw-image-desc"]
sw_img_desc                 = ["sw_img_desc_base", "{}"]
sw_id                       = ["sw_img_desc", "id"]
sw_name                     = ["sw_img_desc", "name"]
sw_image_name_var           = ["sw_img_desc", "cisco-etsi-nfvo-sol1-vnfd-extensions:image
sw_version                  = ["sw_img_desc", "version"]
sw_checksum                 = ["sw_img_desc", "checksum"]
sw_checksum_hash            = ["sw_checksum", "hash"]
sw_checksum_algorithm       = ["sw_checksum", "algorithm"]
```

```
sw_checksum_algorithm_VAL    = "sha-256"
sw_container_format         = ["sw_img_desc", "container-format"]
sw_disk_format              = ["sw_img_desc", "disk-format"]
sw_min_disk                 = ["sw_img_desc", "min-disk"]
sw_min_ram                  = ["sw_img_desc", "min-ram"]
sw_size                     = ["sw_img_desc", "size"]
sw_image                    = ["sw_img_desc", "image"]
sw_operating_sys            = ["sw_img_desc", "operating-system"]
sw_supp_virt_environ        = ["sw_img_desc", "supported-virtualization-environment"]

# **************
# ** Artifact **
# **************
artifact_base               = ["vnfd", "cisco-etsi-nfvo:artifact"]
artifact                    = ["artifact_base", "{}"]
artifact_id                 = ["artifact", "id"]
artifact_dest               = ["artifact", "destination-name"]
artifact_url                = ["artifact", "url"]
artifact_variable_list      = ["artifact", "variable"]
artifact_variable           = ["artifact_variable_list", "{}"]
artifact_variable_id        = ["artifact_variable", "id"]
artifact_variable_desc      = ["artifact_variable", "description"]
artifact_checksum           = ["artifact", "checksum"]
artifact_hash               = ["artifact_checksum", "hash"]
artifact_algorithm          = ["artifact_checksum", "algorithm"]
artifact_hash_DUMMY_VAL     = "9af30fce37a4c5c831e095745744d6d2"
artifact_algorithm_DUMMY_VAL = "etsi-nfv-descriptors:sha-256"
```